



Junos[®] OS

CLI User Guide



Modified: 2017-02-14

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Junos[®] OS CLI User Guide

Copyright © 2017, Juniper Networks, Inc.
All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <http://www.juniper.net/support/eula.html>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

	About the Documentation	xv
	Documentation and Release Notes	xv
	Supported Platforms	xv
	Using the Examples in This Manual	xv
	Merging a Full Example	xvi
	Merging a Snippet	xvi
	Documentation Conventions	xvii
	Documentation Feedback	xix
	Requesting Technical Support	xix
	Self-Help Online Tools and Resources	xix
	Opening a Case with JTAC	xx
Chapter 1	Overview	21
	Introducing the Junos OS Command-Line Interface	21
	Key Features of the CLI	22
	Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies	23
	Junos OS CLI Command Modes	23
	CLI Command Hierarchy	24
	Configuration Statement Hierarchy	24
	Moving Among Hierarchy Levels	25
	Other Tools to Configure and Monitor Devices Running Junos OS	26
	Commands and Configuration Statements for Junos-FIPS	26
Chapter 2	Getting Started: A Quick Tour of the CLI	29
	Getting Started with the Junos OS Command-Line Interface	29
	Switching Between Junos OS CLI Operational and Configuration Modes	31
	Configuring a User Account on a Device Running Junos OS	32
	Using the CLI Editor in Configuration Mode	34
	Checking the Status of a Device Running Junos OS	36
	Example: Configuring a Routing Protocol	38
	Shortcut	39
	Longer Configuration	39
	Making Changes to a Routing Protocol Configuration	41
	Rolling Back Junos OS Configuration Changes	44
Chapter 3	Getting Online Help	47
	Getting Online Help from the Junos OS Command-Line Interface	47
	Getting Help About Commands	47
	Getting Help About a String in a Statement or Command	48
	Getting Help About Configuration Statements	49

Chapter 4

Getting Help About System Log Messages	49
Junos OS CLI Online Help Features	50
Help for Omitted Statements	50
Using CLI Command Completion	50
Using Command Completion in Configuration Mode	51
Displaying Tips About CLI Commands	51
Examples: Using Command Completion in Configuration Mode	51
Examples: Using the Junos OS CLI Command Completion	53
Displaying the Junos OS CLI Command and Word History	54
Using Configuration Statements to Configure a Device	55
Understanding Junos OS CLI Configuration Mode	56
Configuration Mode Commands	57
Configuration Statements and Identifiers	58
Configuration Statement Hierarchy	60
Entering and Exiting the Junos OS CLI Configuration Mode	62
Notational Conventions Used in Junos OS Configuration Hierarchies	64
Forms of the configure Command	65
Using the configure exclusive Command	67
Using the configure Command	68
Modifying the Junos OS Configuration	68
Adding Junos OS Configuration Statements and Identifiers	69
Deleting a Statement from a Junos OS Configuration	70
Example: Deleting a Statement from the Junos OS Configuration	71
Copying a Junos OS Statement in the Configuration	73
Example: Copying a Statement in the Junos Configuration	73
Issuing Relative Junos OS Configuration Mode Commands	75
Renaming an Identifier in a Junos OS Configuration	76
Examples: Re-Using Configuration	76
Inserting a New Identifier in a Junos OS Configuration	81
Example: Inserting a New Identifier in a Junos Configuration	81
Example: Using the Wildcard Command with the Range Option	85
Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration	89
Example: Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration	90
Adding Comments in a Junos OS Configuration	92
Adding Comments in the CLI	92
Adding Comments in a File	93
Example: Including Comments in a Junos OS Configuration by Using the CLI	94
Updating the configure private Configuration	96
Displaying the Current Junos OS Configuration	97
Example: Displaying the Current Junos OS Configuration	98
Displaying Additional Information About the Junos OS Configuration	99
Displaying set Commands from the Junos OS Configuration	101
Example: Displaying set Commands from the Configuration	102
Example: Displaying Required set Commands at the Current Hierarchy Level	102

	Example: Displaying set Commands with the match Option	103
	Displaying Users Currently Editing the Junos OS Configuration	104
	Verifying a Junos OS Configuration	104
Chapter 5	Committing a Junos OS Configuration	107
	Junos OS Commit Model for Router or Switch Configuration	107
	Committing a Junos OS Configuration	108
	Committing a Junos OS Configuration and Exiting Configuration Mode	110
	Commit Operation When Multiple Users Configure the Software	111
	Activating a Junos OS Configuration but Requiring Confirmation	112
	Scheduling a Junos OS Commit Operation	113
	Monitoring the Junos OS Commit Process	114
	Adding a Comment to Describe the Committed Configuration	115
	Backing Up the Committed Configuration on the Alternate Boot Drive	116
	Junos OS Batch Commits Overview	116
	Aggregation and Error Handling	117
	Example: Configuring Batch Commit Server Properties	117
Chapter 6	Managing Configurations	125
	Understanding How the Junos OS Configuration Is Stored	125
	Comparing Configuration Changes with a Prior Version	126
	Understanding the show compare display xml Command Output	128
	Adding a Statement (create Operation)	129
	Deleting a Statement (delete Operation)	129
	Changing a Statement (delete and create Operations)	130
	Changing Metadata (inactive Attribute and Operation)	131
	Adding an Annotation (comment Tag and create Operation)	132
	Changing an Annotation (comment Tag, and delete and create Operations)	132
	Adding a Statement Inside a Container (create Operation, and insert and key Attributes)	133
	Changing the Order Inside a Container (merge Operation, and insert and key Attributes)	134
	Returning to the Most Recently Committed Junos OS Configuration	134
	Returning to a Previously Committed Junos OS Configuration	135
	Returning to a Configuration Prior to the One Most Recently Committed . .	135
	Displaying Previous Configurations	135
	Comparing Configuration Changes with a Prior Version	136
	Creating and Returning to a Rescue Configuration	138
	Saving a Configuration to a File	139
	Saving a Configuration to a File	140
	Additional Details About Specifying Junos OS Statements and Identifiers	141
	Specifying Statements	141
	Performing CLI Type Checking	143
	Loading a Configuration from a File or the Terminal	144
	Examples: Loading a Configuration from a File	147
	Creating and Returning to a Rescue Configuration	149
	Compressing the Current Configuration File	149
	Example: Protecting the Junos OS Configuration from Modification or Deletion	151

	Synchronizing Routing Engines	158
	Configuring Multiple Routing Engines to Synchronize Committed Configurations Automatically	161
Chapter 7	Using Operational Commands to Monitor a Device	163
	Overview of Junos OS CLI Operational Mode Commands	163
	CLI Command Categories	163
	Commonly Used Operational Mode Commands	165
	Junos OS Operational Mode Commands That Combine Other Commands	166
	Understanding the Brief, Detail, Extensive, and Terse Options of Junos OS Operational Commands	167
	Controlling the Scope of an Operational Mode Command	168
	Operational Mode Commands on a TX Matrix Router or TX Matrix Plus Router	169
	Examples of Routing Matrix Command Options	169
	Monitoring Who Uses the Junos OS CLI	171
	Interface Naming Conventions Used in the Junos OS Operational Commands	172
	Physical Part of an Interface Name	172
	Logical Part of an Interface Name	172
	Channel Identifier Part of an Interface Name	173
	Viewing Files and Directories on a Device Running Junos OS	173
	Directories on the Router or Switch	173
	Listing Files and Directories	174
	Specifying Filenames and URLs	176
	Displaying Junos OS Information	177
	Managing Programs and Processes Using Junos OS Operational Mode Commands	179
	Showing Software Processes	180
	Restarting the Junos OS Process	181
	Stopping Junos OS	182
	Rebooting Junos OS	183
	Using the Junos OS CLI Comment Character # for Operational Mode Commands	184
	Example: Using Comments in Junos OS Operational Mode Commands	184
Chapter 8	Filtering Command Output	187
	Using the Pipe () Symbol to Filter Junos OS Command Output	187
	Using Regular Expressions with the Pipe () Symbol to Filter Junos OS Command Output	188
	Filtering Operational Mode Command Output in a QFabric System	189
	Pipe () Filter Functions in the Junos OS Command-Line Interface	190
	Comparing Configurations and Displaying the Differences in Text	190
	Comparing Configurations and Displaying the Differences in XML	192
	Counting the Number of Lines of Output	192
	Displaying Output in XML Tag Format	192
	Displaying Output in JSON Format	193
	Displaying the Configuration with YANG Translation Scripts Applied	193
	Displaying the RPC Tags for a Command	195
	Ignoring Output That Does Not Match a Regular Expression	195

	Displaying Output from the First Match of a Regular Expression	195
	Retaining Output After the Last Screen	196
	Displaying Output Beginning with the Last Entries	196
	Displaying Output That Matches a Regular Expression	196
	Preventing Output from Being Paginated	197
	Sending Command Output to Other Users	197
	Resolving IP Addresses	197
	Saving Output to a File	198
	Appending Output to a File	198
	Displaying Output on Screen and Writing to a File	198
	Trimming Output by Specifying the Starting Column	199
	Refreshing the Output of a Command	199
Chapter 9	Using Shortcuts, Wildcards, and Regular Expressions in the CLI	201
	Using Keyboard Sequences to Move Around and Edit the Junos OS CLI	201
	Using Wildcard Characters in Interface Names	203
	Common Regular Expressions to Use with the replace Command	204
	Using Global Replace in the Junos OS Configuration	205
	Example: Using Global Replace in a Junos OS Configuration—Using the \n Back Reference	206
	Example: Using Global Replace in a Junos OS Configuration—Replacing an Interface Name	208
	Example: Using Global Replace in a Junos OS Configuration—Using the upto Option	210
	Using Regular Expressions to Delete Related Items from a Junos OS cConfiguration	211
Chapter 10	Using Configuration Groups to Quickly Configure Devices	215
	Understanding Junos OS Configuration Groups	216
	Configuration Groups Overview	216
	Inheritance Model	216
	Configuring Configuration Groups	216
	Creating the Junos OS Configuration Group	217
	Applying the Junos OS Configuration Group	219
	Example: Configuring and Applying Junos OS Configuration Groups	220
	Example: Creating and Applying Configuration Groups on a TX Matrix Router . .	221
	Disabling Inheritance of a Junos OS Configuration Group	222
	Using Wildcards with Configuration Groups	224
	Example: Configuring Sets of Statements with Configuration Groups	227
	Example: Configuring Interfaces Using Junos OS Configuration Groups	228
	Example: Configuring a Consistent IP Address for the Management Interface . .	230
	Example: Configuring Peer Entities	232
	Establishing Regional Configurations	234
	Configuring Wildcard Configuration Group Names	235
	Example: Referencing the Preset Statement From the Junos OS defaults Group	236
	Example: Viewing Default Statements That Have Been Applied to the Configuration	237
	Using Conditions to Apply Configuration Groups Overview	238
	Example: Configuring Conditions for Applying Configuration Groups	238

	Improving Commit Time When Using Configuration Groups	240
	Example: Improving Commit Time When Using Configuration Groups	241
	Using Junos OS Defaults Groups	242
	Setting Up Routing Engine Configuration Groups	244
Chapter 11	Controlling the CLI Environment	247
	Controlling the Junos OS CLI Environment	247
	Setting the Terminal Type	248
	Setting the CLI Prompt	248
	Setting the CLI Directory	248
	Setting the CLI Timestamp	248
	Setting the Idle Timeout	248
	Setting the CLI to Prompt After a Software Upgrade	248
	Setting Command Completion	249
	Displaying CLI Settings	249
	Setting the Junos OS CLI Screen Length and Width	249
	Setting the Screen Length	250
	Setting the Screen Width	250
	Example: Controlling the CLI Environment	250
	Example: Enabling Configuration Breadcrumbs	256
Chapter 12	Junos OS Configuration Statements and Commands	259
	apply-groups	260
	apply-groups-except	261
	activate	262
	annotate	263
	commit	264
	commit-interval (Batch Commits)	269
	configuration-breadcrumbs	270
	copy	271
	days-to-keep-error-logs (Batch Commits)	271
	deactivate	272
	delete	273
	edit	274
	exit	275
	export-format	276
	groups	277
	help	279
	insert	280
	load	281
	maximum-aggregate-pool (Batch Commits)	283
	maximum-entries (Batch Commits)	283
	no-hidden-commands	284
	protect	285
	quit	286
	rename	287
	replace	288
	rollback	289
	run	290
	save	291

	server (Batch Commits)	292
	set	293
	show	294
	show configuration	295
	show display inheritance	298
	show display omit	299
	show display set	300
	show display set relative	301
	show groups junos-defaults	302
	status	303
	top	304
	traceoptions (Batch Commits)	305
	unprotect	306
	up	307
	update	308
	when	309
	wildcard delete	311
Chapter 13	Junos OS CLI Environment Commands	313
	set cli complete-on-space	314
	set cli directory	315
	set cli idle-timeout	316
	set cli prompt	317
	set cli restart-on-upgrade	318
	set cli screen-length	319
	set cli screen-width	320
	set cli terminal	321
	set cli timestamp	322
	set date	323
	show cli	324
	show cli	326
	show cli authorization	327
	show cli directory	328
	show cli history	329
Chapter 14	Junos OS CLI Operational Mode Commands	331
	configure	332
	file	334
	help	335
	(pipe)	336
	request	339
	request system commit server pause	341
	request system commit server queue cleanup	342
	request system commit server start	343
	restart	344
	set	355
	show system commit server queue	356
	show system commit server status	360

List of Figures

Chapter 1	Overview	21
	Figure 1: Monitoring and Configuring Routers	22
	Figure 2: Committing a Configuration	24
	Figure 3: Configuration Statement Hierarchy Example	25
Chapter 4	Using Configuration Statements to Configure a Device	55
	Figure 4: Configuration Mode Hierarchy of Statements	61
Chapter 5	Committing a Junos OS Configuration	107
	Figure 5: Confirm a Configuration	113
Chapter 6	Managing Configurations	125
	Figure 6: Overriding the Current Configuration	147
	Figure 7: Using the replace Option	147
	Figure 8: Using the merge Option	147
	Figure 9: Using a Patch File	148
	Figure 10: Using the set Option	148
Chapter 7	Using Operational Commands to Monitor a Device	163
	Figure 11: Commands That Combine Other Commands	167
	Figure 12: Command Output Options	168
	Figure 13: Restarting a Process	182
Chapter 9	Using Shortcuts, Wildcards, and Regular Expressions in the CLI	201
	Figure 14: Replacement by Object	210

List of Tables

	About the Documentation	xv
	Table 1: Notice Icons	xvii
	Table 2: Text and Syntax Conventions	xviii
Chapter 1	Overview	21
	Table 3: CLI Configuration Mode Navigation Commands	25
Chapter 4	Using Configuration Statements to Configure a Device	55
	Table 4: Summary of Configuration Mode Commands	57
	Table 5: Configuration Mode Top-Level Statements	59
	Table 6: Forms of the configure Command	66
Chapter 6	Managing Configurations	125
	Table 7: CLI Configuration Input Types	143
Chapter 7	Using Operational Commands to Monitor a Device	163
	Table 8: Commonly Used Operational Mode Commands	165
	Table 9: Directories on the Router	174
	Table 10: show system process extensive Command Output Fields	181
Chapter 8	Filtering Command Output	187
	Table 11: Common Regular Expression Operators in Operational Mode Commands	188
Chapter 9	Using Shortcuts, Wildcards, and Regular Expressions in the CLI	201
	Table 12: CLI Keyboard Sequences	202
	Table 13: Wildcard Characters for Specifying Interface Names	203
	Table 14: Common Regular Expressions to Use with the replace Command	204
	Table 15: Replacement Examples	205
Chapter 13	Junos OS CLI Environment Commands	313
	Table 16: show cli Output Fields	324

About the Documentation

- [Documentation and Release Notes on page xv](#)
- [Supported Platforms on page xv](#)
- [Using the Examples in This Manual on page xv](#)
- [Documentation Conventions on page xvii](#)
- [Documentation Feedback on page xix](#)
- [Requesting Technical Support on page xix](#)

Documentation and Release Notes

To obtain the most current version of all Juniper Networks® technical documentation, see the product documentation page on the Juniper Networks website at <http://www.juniper.net/techpubs/>.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at <http://www.juniper.net/books>.

Supported Platforms

For the features described in this document, the following platforms are supported:

- [ACX Series](#)
- [M Series](#)
- [MX Series](#)
- [T Series](#)
- [PTX Series](#)

Using the Examples in This Manual

If you want to use the examples in this manual, you can use the **load merge** or the **load merge relative** command. These commands cause the software to merge the incoming

configuration into the current candidate configuration. The example does not become active until you commit the candidate configuration.

If the example configuration contains the top level of the hierarchy (or multiple hierarchies), the example is a *full example*. In this case, use the **load merge** command.

If the example configuration does not start at the top level of the hierarchy, the example is a *snippet*. In this case, use the **load merge relative** command. These procedures are described in the following sections.

Merging a Full Example

To merge a full example, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration example into a text file, save the file with a name, and copy the file to a directory on your routing platform.

For example, copy the following configuration to a file and name the file **ex-script.conf**. Copy the **ex-script.conf** file to the **/var/tmp** directory on your routing platform.

```
system {
  scripts {
    commit {
      file ex-script.xsl;
    }
  }
}
interfaces {
  fxp0 {
    disable;
    unit 0 {
      family inet {
        address 10.0.0.1/24;
      }
    }
  }
}
```

2. Merge the contents of the file into your routing platform configuration by issuing the **load merge** configuration mode command:

```
[edit]
user@host# load merge /var/tmp/ex-script.conf
load complete
```

Merging a Snippet

To merge a snippet, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration snippet into a text file, save the file with a name, and copy the file to a directory on your routing platform.

For example, copy the following snippet to a file and name the file **ex-script-snippet.conf**. Copy the **ex-script-snippet.conf** file to the **/var/tmp** directory on your routing platform.


```
commit {
  file ex-script-snippet.xml; }
```

2. Move to the hierarchy level that is relevant for this snippet by issuing the following configuration mode command:

```
[edit]
user@host# edit system scripts
[edit system scripts]
```

3. Merge the contents of the file into your routing platform configuration by issuing the **load merge relative** configuration mode command:

```
[edit system scripts]
user@host# load merge relative /var/tmp/ex-script-snippet.conf
load complete
```

For more information about the **load** command, see [CLI Explorer](#).

Documentation Conventions

Table 1 on page xvii defines notice icons used in this guide.

Table 1: Notice Icons

Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.
	Tip	Indicates helpful information.
	Best practice	Alerts you to a recommended use or implementation.

Table 2 on page xviii defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the configure command: user@host> configure
Fixed-width text like this	Represents output that appears on the terminal screen.	user@host> show chassis alarms No alarms currently active
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces or emphasizes important new terms. Identifies guide names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS CLI User Guide</i> RFC 1997, <i>BGP Communities Attribute</i>
<i>Italic text like this</i>	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# set system domain-name <i>domain-name</i>
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none"> To configure a stub area, include the stub statement at the [edit protocols ospf area area-id] hierarchy level. The console port is labeled CONSOLE.
< > (angle brackets)	Encloses optional keywords or variables.	stub <default-metric <i>metric</i> >;
(pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	broadcast multicast (<i>string1</i> <i>string2</i> <i>string3</i>)
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	rsvp { # Required for dynamic MPLS only
[] (square brackets)	Encloses a variable for which you can substitute one or more values.	community name members [<i>community-ids</i>]
Indentation and braces ({ })	Identifies a level in the configuration hierarchy.	[edit] routing-options { static { route default { nexthop <i>address</i> ; retain; } } }
;(semicolon)	Identifies a leaf statement at a configuration hierarchy level.	

GUI Conventions

Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
Bold text like this	Represents graphical user interface (GUI) items you click or select.	<ul style="list-style-type: none"> In the Logical Interfaces box, select All Interfaces. To cancel the configuration, click Cancel.
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select Protocols>Ospf .

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can provide feedback by using either of the following methods:

- Online feedback rating system—On any page of the Juniper Networks TechLibrary site at <http://www.juniper.net/techpubs/index.html>, simply click the stars to rate the content, and use the pop-up form to provide us with information about your experience. Alternately, you can use the online feedback form at <http://www.juniper.net/techpubs/feedback/>.
- E-mail—Send your comments to techpubs-comments@juniper.net. Include the document or topic name, URL or page number, and software version (if applicable).

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or Partner Support Service support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at <http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf>.
- Product warranties—For product warranty information, visit <http://www.juniper.net/support/warranty/>.
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: <http://www.juniper.net/customers/support/>
- Search for known bugs: <http://www2.juniper.net/kb/>
- Find product documentation: <http://www.juniper.net/techpubs/>
- Find solutions and answer questions using our Knowledge Base: <http://kb.juniper.net/>
- Download the latest versions of software and review release notes:
<http://www.juniper.net/customers/csc/software/>
- Search technical bulletins for relevant hardware and software notifications:
<http://kb.juniper.net/InfoCenter/>
- Join and participate in the Juniper Networks Community Forum:
<http://www.juniper.net/company/communities/>
- Open a case online in the CSC Case Management tool: <http://www.juniper.net/cm/>

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: <https://tools.juniper.net/SerialNumberEntitlementSearch/>

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at <http://www.juniper.net/cm/>.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see <http://www.juniper.net/support/requesting-support.html>.

CHAPTER 1

Overview

- [Introducing the Junos OS Command-Line Interface on page 21](#)
- [Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies on page 23](#)
- [Other Tools to Configure and Monitor Devices Running Junos OS on page 26](#)
- [Commands and Configuration Statements for Junos-FIPS on page 26](#)

Introducing the Junos OS Command-Line Interface

The Junos[®] operating system (Junos OS) command-line interface (CLI) is the software interface you use to access a device running Junos OS—whether from the console or through a network connection.

The Junos OS CLI is a Juniper Networks-specific command shell that runs on top of a FreeBSD UNIX-based operating system kernel. By leveraging industry-standard tools and utilities, the CLI provides a powerful set of commands that you can use to monitor and configure devices running Junos OS (see [Figure 1 on page 22](#)).

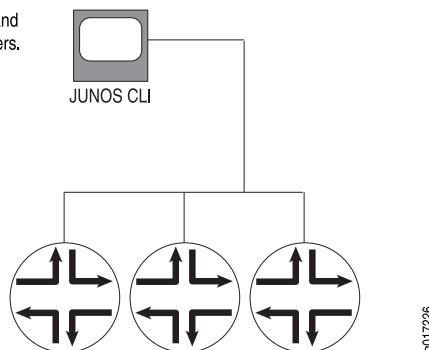
The Junos OS CLI has two modes:

- **Operational mode**—This mode displays the current status of the device. In operational mode, you enter commands to monitor and troubleshoot the Junos OS, devices, and network connectivity.
- **Configuration mode**—This mode enables you to configure the device. A configuration is stored as a hierarchy of configuration statements. In this mode, you enter statements to configure all properties of the device, including interfaces, general routing information, routing protocols, user access, and several system and hardware properties.

When you enter configuration mode, you are actually viewing and changing a file called the *candidate configuration*. The candidate configuration file enables you to make configuration changes without causing operational changes to the current operating configuration, called the *active configuration*. The router or switch does not implement the changes you added to the candidate configuration file until you commit them, which activates the configuration on the device. Candidate configurations enable you to alter your configuration without causing potential damage to your current network operations.

Figure 1: Monitoring and Configuring Routers

Use the JUNOS CLI to monitor and configure Juniper Networks routers.



Key Features of the CLI

The Junos OS CLI commands and statements follow a hierarchal organization and have a regular syntax. The Junos OS CLI provides the following features to simplify CLI use:

- Consistent command names—Commands that provide the same type of function have the same name, regardless of the portion of the software on which they are operating. For example, all **show** commands display software information and statistics, and all **clear** commands erase various types of system information.
- Lists and short descriptions of available commands—Information about available commands is provided at each level of the CLI command hierarchy. If you type a question mark (?) at any level, you see a list of the available commands along with a short description of each command. This means that if you already are familiar with the Junos OS or with other routing software, you can use many of the CLI commands without referring to the documentation.
- Command completion—Command completion for command names (keywords) and for command options is available at each level of the hierarchy. To complete a command or option that you have partially typed, press the Tab key or the Spacebar. If the partially typed letters begin a string that uniquely identifies a command, the complete command name appears. Otherwise, a beep indicates that you have entered an ambiguous command, and the possible completions are displayed. Completion also applies to other strings, such as filenames, interface names, usernames, and configuration statements.

If you have typed the mandatory arguments for executing a command in the operational or configuration mode the CLI displays **<[Enter]>** as one of the choices when you type a question mark (?). This indicates that you have entered the mandatory arguments and can execute the command at that level without specifying any further options. Likewise, the CLI also displays **<[Enter]>** when you have reached a specific hierarchy level in the configuration mode and do not have to enter any more mandatory arguments or statements.

- Industry-standard technology—With FreeBSD UNIX as the kernel, a variety of UNIX utilities are available on the Junos OS CLI. For example, you can:
 - Use regular expression matching to locate and replace values and identifiers in a configuration, filter command output, or examine log file entries.

- Use Emacs-based key sequences to move around on a command line and scroll through the recently executed commands and command output.
- Store and archive Junos OS device files on a UNIX-based file system.
 - Use standard UNIX conventions to specify filenames and paths.
- Exit from the CLI environment and create a UNIX C shell or Bourne shell to navigate the file system, manage router processes, and so on.

Related Documentation

- [Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies on page 23](#)
- [Getting Started with the Junos OS Command-Line Interface on page 29](#)
- [Other Tools to Configure and Monitor Devices Running Junos OS on page 26](#)
- [Commands and Configuration Statements for Junos-FIPS on page 26](#)

Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies

The Junos OS command-line interface (CLI) commands and statements are organized under two command modes and various hierarchies. The following sections provide you an overview of the Junos OS CLI command modes and commands and statements hierarchies:

- [Junos OS CLI Command Modes on page 23](#)
- [CLI Command Hierarchy on page 24](#)
- [Configuration Statement Hierarchy on page 24](#)
- [Moving Among Hierarchy Levels on page 25](#)

Junos OS CLI Command Modes

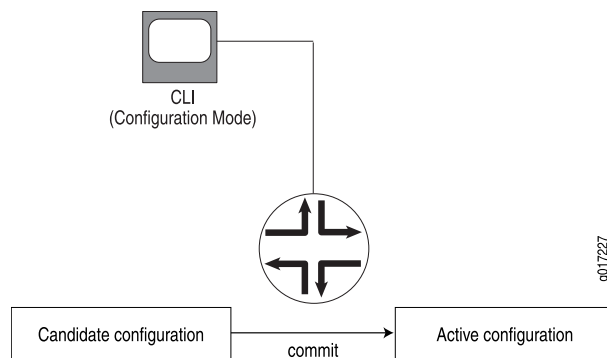
The Junos OS CLI has two modes:

- **Operational mode**—This mode displays the current status of the device. In operational mode, you enter commands to monitor and troubleshoot the Junos OS, devices, and network connectivity. To enter the operational mode, type the **CLI** command. The character “>” identifies operational mode. For example, `user@router>`
- **Configuration mode**—A configuration for a device running on Junos OS is stored as a hierarchy of statements. In configuration mode, you enter these statements to define all properties of the Junos OS, including interfaces, general routing information, routing protocols, user access, and several system and hardware properties. You enter the configuration mode by issuing the **configure** command from the operational mode. The character “#” identifies configuration mode. For example, `user@router#`

When you enter configuration mode, you are actually viewing and changing a file called the *candidate configuration*. The candidate configuration file enables you to make configuration changes without causing operational changes to the current operating configuration, called the *active configuration*. The router or switch does not implement

the changes you added to the candidate configuration file until you commit them, which activates the configuration on the router or switch (see [Figure 2 on page 24](#)). Candidate configurations enable you to alter your configuration without causing potential damage to your current network operations.

Figure 2: Committing a Configuration



CLI Command Hierarchy

CLI commands are organized in a hierarchy. Commands that perform a similar function are grouped together under the same level of the hierarchy. For example, all commands that display information about the system and the system software are grouped under the **show system** command, and all commands that display information about the routing table are grouped under the **show route** command.

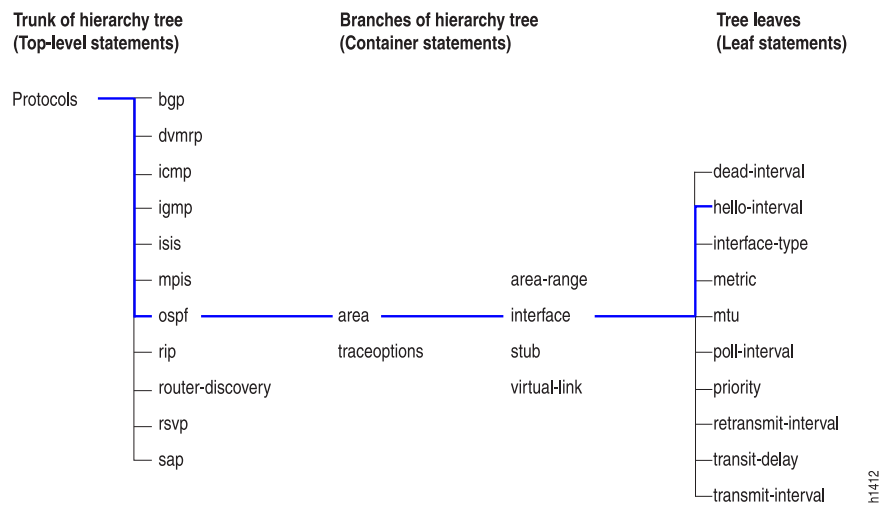
To execute a command, you enter the full command name, starting at the top level of the hierarchy. For example, to display a brief view of the routes in the routing table, use the command **show route brief**.

Configuration Statement Hierarchy

The configuration statement hierarchy has two types of statements: *container statements*, which are statements that contain other statements, and *leaf statements*, which do not contain other statements. All of the container and leaf statements together form the *configuration hierarchy*.

[Figure 3 on page 25](#) illustrates a part of the hierarchy tree. The **protocols** statement is a top-level statement at the trunk of the configuration tree. The **ospf**, **area**, and **interface** statements are all subordinate container statements of a higher statement (they are branches of the hierarchy tree), and the **hello-interval** statement is a leaf on the tree.

Figure 3: Configuration Statement Hierarchy Example



Moving Among Hierarchy Levels

You can use the CLI commands in [Table 3 on page 25](#) to navigate the levels of the configuration statement hierarchy.

Table 3: CLI Configuration Mode Navigation Commands

Command	Description
edit <i>hierarchy-level</i>	Moves to an existing configuration statement hierarchy or creates a hierarchy and moves to that level.
exit	Moves up the hierarchy to the previous level where you were working. This command is, in effect, the opposite of the edit command. Alternatively, you can use the quit command. The exit and quit commands are interchangeable.
up	Moves up the hierarchy one level at a time.
top	Moves directly to the top level of the hierarchy.

Related Documentation

- [Introducing the Junos OS Command-Line Interface on page 21](#)
- [Getting Started with the Junos OS Command-Line Interface on page 29](#)

Other Tools to Configure and Monitor Devices Running Junos OS

Apart from the command-line interface, Junos OS also supports the following applications, scripts, and utilities that enable you to configure and monitor devices running Junos OS:

- J-Web graphical user interface (GUI)—Allows you to monitor, configure, troubleshoot, and manage the router on a client by means of a Web browser with Hypertext Transfer Protocol (HTTP) or HTTP over Secure Sockets Layer (HTTPS) enabled. For more information, see the *J-Web Interface User Guide*.
- Junos XML management protocol—Application programmers can use the Junos XML management protocol to monitor and configure Juniper Networks routers. Juniper Networks provides a Perl module with the API to help you more quickly and easily develop custom Perl scripts for configuring and monitoring routers. For more information, see the *Junos XML Management Protocol Developer Guide*.
- NETCONF Application Programming Interface (API)—Application programmers can also use the NETCONF XML management protocol to monitor and configure Juniper Networks routers. For more information, see the *NETCONF XML Management Protocol Developer Guide*.
- Junos OS commit scripts and self-diagnosis features—You can define scripts to enforce custom configuration rules, use commit script macros to provide simplified aliases for frequently used configuration statements, and configure diagnostic event policies and actions associated with each policy. For more information, see the *Automation Scripting Feature Guide*.
- Management Information Bases (MIBs)—You can use enterprise-specific and standard MIBs to retrieve information about the hardware and software components on a Juniper Networks router. For more information about MIBs, see the *Network Management Administration Guide*.

Related Documentation

- [Introducing the Junos OS Command-Line Interface on page 21](#)
- [Getting Started with the Junos OS Command-Line Interface on page 29](#)
- [Commands and Configuration Statements for Junos-FIPS on page 26](#)

Commands and Configuration Statements for Junos-FIPS

Junos-FIPS enables you to configure a network of Juniper Networks routers in a Federal Information Processing Standards (FIPS) 140-2 environment.

The Junos-FIPS software environment requires the installation of FIPS software by a crypto officer. In Junos-FIPS, some Junos OS commands and statements have restrictions and some additional configuration statements are available. For more information, see the following resources:

- *Common Criteria and FIPS Certifications*—Provides links to guidelines for configuring devices running Junos OS so that the secure environment is in compliance with the

requirements of public sector certifications such as Common Criteria (CC) and FIPS certification.

- [Compliance Advisor](#)—A Web application that provides regulatory compliance information about Common Criteria, FIPS, Homologation, ROHS2, and USGv6 for Juniper Networks products.

**Related
Documentation**

- *IPsec Requirements for Junos-FIPS*
- *Configuring IPsec for Enabling Internal Communications Between Routing Engines for Junos OS in FIPS Mode*

CHAPTER 2

Getting Started: A Quick Tour of the CLI

- [Getting Started with the Junos OS Command-Line Interface on page 29](#)
- [Switching Between Junos OS CLI Operational and Configuration Modes on page 31](#)
- [Configuring a User Account on a Device Running Junos OS on page 32](#)
- [Using the CLI Editor in Configuration Mode on page 34](#)
- [Checking the Status of a Device Running Junos OS on page 36](#)
- [Example: Configuring a Routing Protocol on page 38](#)
- [Rolling Back Junos OS Configuration Changes on page 44](#)

Getting Started with the Junos OS Command-Line Interface

As an introduction to the Junos OS command-line interface (CLI), this topic provides instructions for simple steps you take after installing Junos OS on the device. It shows you how to start the CLI, view the command hierarchy, and make small configuration changes. The related topics listed at the end of this topic provide you more detailed information about using the CLI.



NOTE:

- The instructions and examples in this topic are based on sample M Series and T Series routers. You can use them as a guideline for entering commands on your devices running Junos OS.
- Before you begin, make sure your device hardware is set up and Junos OS is installed. You must have a direct console connection to the device or network access using SSH or Telnet. If your device is not set up, follow the installation instructions provided with the device before proceeding.

To log in to a router and start the CLI:

1. Log in as **root**.

The root login account has superuser privileges, with access to all commands and statements.

2. Start the CLI:

```
root# cli
```

```
root@>
```

The > command prompt shows you are in operational mode. Later, when you enter configuration mode, the prompt will change to #.



NOTE: If you are using the root account for the first time on the device, remember that the device ships with no password required for root, but the first time you commit a configuration with Junos OS Release 7.6 or later, you must set a root password. Root access is not allowed over a telnet session. To enable root access over an SSH connection, you must configure the `system services ssh root-login allow` statement.

The CLI includes several ways to get help about commands. This section shows some examples of how to get help:

1. Type `?` to show the top-level commands available in operational mode.

```
root@> ?
Possible completions:
clear          Clear information in the system
configure      Manipulate software configuration information
diagnose       Invoke diagnose script
file           Perform file operations
help           Provide help information
monitor        Show real-time debugging information
mtrace         Trace multicast path from source to receiver
ping           Ping remote target
quit           Exit the management session
request        Make system-level requests
restart         Restart software process
set            Set CLI properties, date/time, craft interface message
show           Show system information
ssh            Start secure shell on another host
start          Start shell
telnet         Telnet to another host
test           Perform diagnostic debugging
traceroute     Trace route to remote host
```

2. Type `file ?` to show all possible completions for the `file` command.

```
root@> file ?
Possible completions:
<[Enter]>      Execute this command
archive        Archives files from the system
checksum       Calculate file checksum
compare        Compare files
copy           Copy files (local or remote)
delete         Delete files from the system
list           List file information
rename         Rename files
show           Show file contents
source-address Local address to use in originating the connection
|             Pipe through a command
```

3. Type `file archive ?` to show all possible completions for the `file archive` command.

```
root@> file archive ?
```

Possible completions:

compress	Compresses the archived file using GNU gzip (.tgz)
destination	Name of created archive (URL, local, remote, or floppy)
source	Path of directory to archive

**Related
Documentation**

- [Getting Online Help from the Junos OS Command-Line Interface on page 47](#)
- [Switching Between Junos OS CLI Operational and Configuration Modes on page 31](#)
- [Checking the Status of a Device Running Junos OS on page 36](#)
- [Configuring a User Account on a Device Running Junos OS on page 32](#)
- [Example: Configuring a Routing Protocol on page 38](#)
- [Examples: Using the Junos OS CLI Command Completion on page 53](#)

Switching Between Junos OS CLI Operational and Configuration Modes

When you monitor and configure a device running Junos OS, you may need to switch between operational mode and configuration mode. When you change to configuration mode, the command prompt also changes. The operational mode prompt is a right angle bracket (>) and the configuration mode prompt is a pound sign (#).

To switch between operational mode and configuration mode:

1. When you log in to the router and type the **cli** command, you are automatically in operational mode:

```
--- JUNOS 9.2B1.8 built 2008-05-09 23:41:29 UTC
% cli
user@host>
```

2. To enter configuration mode, type the **configure** command or the **edit** command from the CLI operation mode. For example:

```
user@host> configure
Entering configuration mode
```

```
[edit]
user@host#
```

The CLI prompt changes from **user@host>** to **user@host#** and a banner appears to indicate the hierarchy level.

3. You can return to operational mode in one of the following ways:

- To commit the configuration and exit:

```
[edit]
user@host# commit and-quit
commit complete
Exiting configuration mode
user@host>
```

- To exit without committing:

```
[edit]
```

```
user@host# exit
Exiting configuration mode
user@host>
```

When you exit configuration mode, the CLI prompt changes from **user@host#** to **user@host>** and the banner no longer appears. You can enter or exit configuration mode as many times as you wish without committing your changes.

4. To display the output of an operational mode command, such as **show**, while in configuration mode, issue the **run** configuration mode command and then specify the operational mode command:

```
[edit]
user@host# run operational-mode-command
```

For example, to display the currently set priority value of the Virtual Router Redundancy Protocol (VRRP) primary router while you are modifying the VRRP configuration for a backup router:

```
[edit interfaces xe-4/2/0 unit 0 family inet vrrp-group 27]
user@host# show
virtual-address [ 192.168.1.15 ];
[edit interfaces xe-4/2/0 unit 0 family inet vrrp-group 27]
user@host# run show vrrp detail
Physical interface: xe-5/2/0, Unit: 0, Address: 192.168.29.10/24
Interface state: up, Group: 10, State: backup
Priority: 190, Advertisement interval: 3, Authentication type: simple
Preempt: yes, VIP count: 1, VIP: 192.168.29.55
Dead timer: 8.326, Master priority: 201, Master router: 192.168.29.254
[edit interfaces xe-4/2/0 unit 0 family inet vrrp-group 27]
user@host# set priority ...
```

Related Documentation

- [Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies on page 23](#)
- [Getting Online Help from the Junos OS Command-Line Interface on page 47](#)
- [Configuring a User Account on a Device Running Junos OS on page 32](#)

Configuring a User Account on a Device Running Junos OS

This topic describes how to log on to a device running Junos OS using a root account and configure a new user account. You can configure an account for your own use or create a test account.

To configure a new user account on the device:

1. Log in as root and enter configuration mode:

```
root@host> configure
[edit]
root@host#
```

The prompt in brackets (**[edit]**), also known as a *banner*, shows that you are in configuration edit mode at the top of the hierarchy.

2. Change to the **[edit system login]** section of the configuration:

```
[edit]
root@host# edit system login
[edit system login]
root@host#
```

The prompt in brackets changes to **[edit system login]** to show that you are at a new level in the hierarchy.

3. Now add a new user account:

```
[edit system login]
root@host# edit user nchen
```

This example adds an account **nchen** (for Nathan Chen).



NOTE: In Junos OS Release 12.2 and later, user account names can contain a period (.) in the name. For example, you can have a user account named **nathan.chen**. However, the username cannot begin or end with a period.

4. Configure a full name for the account. If the name includes spaces, enclose the entire name in quotation marks (" "):

```
[edit system login user nchen]
root@host# set full-name "Nathan Chen"
```

5. Configure an account class. The account class sets the user access privileges for the account:

```
[edit system login user nchen]
root@host# set class super-user
```

6. Configure an authentication method and password for the account:

```
[edit system login user nchen]
root@host# set authentication plain-text-password
New password:
Retype new password:
```

When the new password prompt appears, enter a clear-text password that the system can encrypt, and then confirm the new password.

7. Commit the configuration:

```
[edit system login user nchen]
root@host# commit
commit complete
```

Configuration changes are not activated until you commit the configuration. If the commit is successful, a **commit complete** message appears.

8. Return to the top level of the configuration, and then exit:

```
[edit system login user nchen]
root@host# top
[edit]
root@host# exit
Exiting configuration mode
```

9. Log out of the device:

```
root@host> exit
% logout Connection closed.
```

10. To test your changes, log back in with the user account and password you just configured:

```
login: nchen
Password: password
--- Junos 8.3-R1.1 built 2005-12-15 22:42:19 UTC
nchen@host>
```

When you log in, you should see the new username at the command prompt.

You have successfully used the CLI to view the device status and perform a simple configuration change. See the related topics listed in this section for more information about the Junos OS CLI features.



NOTE: For complete information about the commands to issue to configure your device, including examples, see the Junos OS configuration guides.

Related Documentation

- [Getting Started with the Junos OS Command-Line Interface on page 29](#)
- [Getting Online Help from the Junos OS Command-Line Interface on page 47](#)
- [Displaying the Junos OS CLI Command and Word History on page 54](#)
- [Example: Configuring a Routing Protocol on page 38](#)

Using the CLI Editor in Configuration Mode

This topic describes some of the basic commands that you must use to enter configuration mode in the command-line interface (CLI) editor, navigate through the configuration hierarchy, get help, and commit or revert the changes that you make during the configuration session.

Task	Command/Statement	Example
Edit Your Configuration		
Enter configuration mode.	<code>configure</code>	<code>user@host> configure</code>
When you first log in to the device, the device is in operational mode. You must explicitly enter configuration mode. When you do, the CLI prompt changes from <code>user@host></code> to <code>user@host#</code> and the hierarchy level appears in square brackets.		<code>[edit]</code> <code>user@host#</code>

Task	Command/Statement	Example
<p>Create a statement hierarchy.</p> <p>You can use the edit command to simultaneously create a hierarchy and move to that new level in the hierarchy. You cannot use the edit command to change the value of identifiers.</p>	<code>edit hierarchy-level value</code>	<pre>[edit] user@host# edit security zones security-zone myzone [edit security zones security-zone myzone] user@host#</pre>
<p>Create a statement hierarchy and set identifier values.</p> <p>The set command is similar to edit except that your current level in the hierarchy does not change.</p>	<code>set hierarchy-level value</code>	<pre>[edit] user@host# set security zones security-zone myzone [edit] user@host#</pre>
Navigate the Hierarchy		
Navigate down to an existing hierarchy level.	<code>edit hierarchy-level</code>	<pre>[edit] user@host# edit security zones [edit security zones] user@host#</pre>
Navigate up one level in the hierarchy.	<code>up</code>	<pre>[edit security zones] user@host# up [edit security] user@host#</pre>
Navigate to the top of the hierarchy.	<code>top</code>	<pre>[edit security zones] user@host# top [edit] user@host#</pre>
Commit or Revert Changes		
Commit your configuration.	<code>commit</code>	<pre>[edit] user@host# commit commit complete</pre>
<p>Roll back changes from the current session.</p> <p>Use the rollback command to revert all changes from the current configuration session. When you run the rollback command before exiting your session or committing changes, the software loads the most recently committed configuration onto the device. You must enter the rollback statement at the edit level in the hierarchy.</p>	<code>rollback</code>	<pre>[edit] user@host# rollback load complete</pre>
Exit Configuration Mode		

Task	Command/Statement	Example
Commit the configuration and exit configuration mode.	<code>commit and-quit</code>	[edit] user@host# <code>commit and-quit</code> user@host>
Exit configuration mode without committing your configuration. You must navigate to the top of the hierarchy using the up or top commands before you can exit configuration mode.	<code>exit</code>	[edit] user@host# <code>exit</code> The configuration has been changed but not committed Exit with uncommitted changes? [yes,no] (yes)
Get Help		
Display a list of valid options for the current hierarchy level.	<code>?</code>	[edit] user@host# <code>edit security zones ?</code> Possible completions: <[Enter]> Execute this command > functional-zone Functional zone > security-zone Security zones Pipe through a command [edit]

- Related Documentation**
- [Understanding Junos OS CLI Configuration Mode on page 56](#)
 - [Entering and Exiting the Junos OS CLI Configuration Mode on page 62](#)
 - [Displaying the Current Junos OS Configuration on page 97](#)

Checking the Status of a Device Running Junos OS

You can use **show** commands to check the status of the device and monitor the activities on the device.

To help you become familiar with **show** commands:

- Type **show ?** to display the list of **show** commands you can use to monitor the router:

```
root@> show ?
Possible completions:
accounting      Show accounting profiles and records
aps             Show Automatic Protection Switching information
arp            Show system Address Resolution Protocol table entries
as-path        Show table of known autonomous system paths
bfd            Show Bidirectional Forwarding Detection information
bgp            Show Border Gateway Protocol information
chassis        Show chassis information
class-of-service Show class-of-service (CoS) information
cli            Show command-line interface settings
configuration   Show current configuration
connections     Show circuit cross-connect connections
dvmrp          Show Distance Vector Multicast Routing Protocol
info
dynamic-tunnels Show dynamic tunnel information information
esis           Show end system-to-intermediate system information
```

firewall	Show firewall information
helper	Show port-forwarding helper information
host	Show hostname information from domain name server
igmp	Show Internet Group Management Protocol information
ike	Show Internet Key Exchange information
ilmi	Show interim local management interface information
interfaces	Show interface information
ipsec	Show IP Security information
ipv6	Show IP version 6 information
isis	Show Intermediate System-to-Intermediate System info
l2circuit	Show Layer 2 circuit information
l2vpn	Show Layer 2 VPN information
lACP	Show Link Aggregation Control Protocol information
ldp	Show Label Distribution Protocol information
link-management	Show link management information
llc2	Show LLC2 protocol related information
log	Show contents of log file
mld	Show multicast listener discovery information
mpls	Show Multiprotocol Label Switching information
msdp	Show Multicast Source Discovery Protocol information
multicast	Show multicast information
ntp	Show Network Time Protocol information
ospf	Show Open Shortest Path First information
ospf3	Show Open Shortest Path First version 3 information
passive-monitoring	Show information about passive monitoring
pfe	Show Packet Forwarding Engine information
pgm	Show Pragmatic Generalized Multicast information
pim	Show Protocol Independent Multicast information
policer	Show interface policer counters and information
policy	Show policy information
ppp	Show PPP process information
rip	Show Routing Information Protocol information
ripng	Show Routing Information Protocol for IPv6 info
route	Show routing table information
rsvp	Show Resource Reservation Protocol information
sap	Show Session Announcement Protocol information
security	Show security information
services	Show services information
snmp	Show Simple Network Management Protocol information
system	Show system information
task	Show routing protocol per-task information
ted	Show Traffic Engineering Database information
version	Show software process revision levels
vpls	Show VPLS information
vrrp	Show Virtual Router Redundancy Protocol information

- Use the **show chassis routing-engine** command to view the Routing Engine status:

```

root@> show chassis routing-engine
Routing Engine status:
Slot 0:
  Current state           Master
  Election priority       Master (default)
  Temperature             31 degrees C / 87 degrees F
  CPU temperature         32 degrees C / 89 degrees F
  DRAM                    768 MB
  Memory utilization      84 percent
  CPU utilization:
    User                  0 percent
    Background            0 percent
    Kernel                1 percent
    Interrupt             0 percent

```

```

Idle 99 percent
Model RE-2.0
Serial ID b10000078c10d701
Start time 2005-12-28 13:52:00 PST
Uptime 12 days, 3 hours, 44 minutes, 19 seconds
Load averages: 1 minute 5 minute 15 minute
                 0.02      0.01      0.00

```

- Use the **show system storage** command to view available storage on the device:

```
root@> show system storage
```

Filesystem	Size	Used	Avail	Capacity	Mounted on
/dev/ad0s1a	865M	127M	669M	16%	/
devfs	1.0K	1.0K	0B	100%	/dev
devfs	1.0K	1.0K	0B	100%	/dev/
/dev/md0	30M	30M	0B	100%	/packages/mnt/jbase
/dev/md1	158M	158M	0B	100%	
/packages/mnt/jkernel-9.3B1.5					
/dev/md2	16M	16M	0B	100%	
/packages/mnt/jpfe-M7i-9.3B1.5					
/dev/md3	3.8M	3.8M	0B	100%	
/packages/mnt/jdocs-9.3B1.5					
/dev/md4	44M	44M	0B	100%	
/packages/mnt/jroute-9.3B1.5					
/dev/md5	12M	12M	0B	100%	
/packages/mnt/jcrypto-9.3B1.5					
/dev/md6	25M	25M	0B	100%	
/packages/mnt/jpfe-common-9.3B1.5					
/dev/md7	1.5G	196K	1.4G	0%	/tmp
/dev/md8	1.5G	910K	1.4G	0%	/mfs
/dev/ad0s1e	96M	38K	88M	0%	/config
procfs	4.0K	4.0K	0B	100%	/proc
/dev/ad1s1f	17G	2.6G	13G	17%	/var

Related Documentation

- [Displaying the Junos OS CLI Command and Word History on page 54](#)
- [Managing Programs and Processes Using Junos OS Operational Mode Commands on page 179](#)
- [Viewing Files and Directories on a Device Running Junos OS on page 173](#)

Example: Configuring a Routing Protocol

This topic provides a sample configuration that describes how to configure an OSPF backbone area that has two SONET interfaces.

The final configuration looks like this:

```

[edit]
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/0 {
        hello-interval 5;
        dead-interval 20;
      }
      interface so-0/0/1 {

```

```

        hello-interval 5;
        dead-interval 20;
    }
}
}
}

```

This topic contains the following examples of configuring a routing protocol:

- [Shortcut on page 39](#)
- [Longer Configuration on page 39](#)
- [Making Changes to a Routing Protocol Configuration on page 41](#)

Shortcut

You can create a shortcut for this entire configuration with the following two commands:

```

[edit]
user@host# set protocols ospf area 0.0.0.0 interface so-0/0/0 hello-interval 5
               dead-interval 20
[edit]
user@host# set protocols ospf area 0.0.0.0 interface so-0/0/1 hello-interval 5
               dead-interval 20

```

Longer Configuration

This section provides a longer example of creating the previous OSPF configuration. In the process, it illustrates how to use the different features of the CLI.

1. Enter configuration mode by issuing the **configure** top-level command:

```

user@host> configure
entering configuration mode
[edit]
user@host#

```

Notice that the prompt has changed to a pound sign (#) to indicate configuration mode.

2. To create the above configuration, you start by editing the **protocols ospf** statements:

```

[edit]
user@host# edit protocols ospf
[edit protocols ospf]
user@host#

```

3. Now add the OSPF area:

```

[edit protocols ospf]
user@host# edit area 0.0.0.0
[edit protocols ospf area 0.0.0.0]
user@host#

```

4. Add the first interface:

```

[edit protocols ospf area 0.0.0.0]
user@host# edit interface so0
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]

```

```
user@host#
```

You now have four nested statements.

5. Set the hello and dead intervals.

```
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host# set ?
user@host# set hello-interval 5
user@host# set dead-interval 20
user@host#
```

6. You can see what is configured at the current level with the **show** command:

```
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host# show
hello-interval 5;
dead-interval 20;
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host#
```

7. You are finished at this level, so back up a level and take a look at what you have so far:

```
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host# up
[edit protocols ospf area 0.0.0.0]
user@host# show
interface so-0/0/0 {
    hello-interval 5;
    dead-interval 20;
}
[edit protocols ospf area 0.0.0.0]
user@host#
```

The **interface** statement appears because you have moved to the **area** statement.

8. Add the second interface:

```
[edit protocols ospf area 0.0.0.0]
user@host# edit interface so-0/0/1
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# set hello-interval 5
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# set dead-interval 20
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# up
[edit protocols ospf area 0.0.0.0]
user@host# show
interface so-0/0/0 {
    hello-interval 5;
    dead-interval 20;
}
interface so-0/0/1 {
    hello-interval 5;
    dead-interval 20;
}
[edit protocols ospf area 0.0.0.0]
user@host#
```


9. Back up to the top level and see what you have:

```
[edit protocols ospf area 0.0.0.0]
user@host# top
[edit]
user@host# show
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/0 {
        hello-interval 5;
        dead-interval 20;
      }
      interface so-0/0/1 {
        hello-interval 5;
        dead-interval 20;
      }
    }
  }
}
[edit]
user@host#
```

This configuration now contains the statements you want.

10. Before committing the configuration (and thereby activating it), verify that the configuration is correct:

```
[edit]
user@host# commit check
configuration check succeeds
[edit]
user@host#
```

11. Commit the configuration to activate it on the router:

```
[edit]
user@host# commit
commit complete
[edit]
user@host#
```

Making Changes to a Routing Protocol Configuration

Suppose you decide to use different dead and hello intervals on interface **so-0/0/1**. You can make changes to the configuration.

1. Go directly to the appropriate hierarchy level by typing the full hierarchy path to the statement you want to edit:

```
[edit]
user@host# edit protocols ospf area 0.0.0.0 interface so-0/0/1
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# show
hello-interval 5;
dead-interval 20;
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
```

```
user@host# set hello-interval 7
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# set dead-interval 28
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# top
[edit]
user@host# show
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/0 {
        hello-interval 5;
        dead-interval 20;
      }
      interface so-0/0/1 {
        hello-interval 7;
        dead-interval 28;
      }
    }
  }
}
[edit]
user@host#
```

2. If you decide not to run OSPF on the first interface, delete the statement:

```
[edit]
user@host# edit protocols ospf area 0.0.0.0
[edit protocols ospf area 0.0.0.0]
user@host# delete interface so-0/0/0
[edit protocols ospf area 0.0.0.0]
user@host# top
[edit]
user@host# show
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/1 {
        hello-interval 7;
        dead-interval 28;
      }
    }
  }
}
[edit]
user@host#
```

Everything inside the statement you deleted was deleted with it. You can also eliminate the entire OSPF configuration by simply entering **delete protocols ospf** while at the top level.

3. If you decide to use the default values for the hello and dead intervals on your remaining interface but you want OSPF to run on that interface, delete the hello and dead interval timers:

```
[edit]
user@host# edit protocols ospf area 0.0.0.0 interface so-0/0/1
```

```

[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# delete hello-interval
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# delete dead-interval
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# top
[edit]
user@host# show
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/1;
    }
  }
}
[edit]
user@host#

```

You can set multiple statements at the same time as long as they are all part of the same hierarchy (the path of statements from the top inward, as well as one or more statements at the bottom of the hierarchy). This feature can reduce considerably the number of commands you must enter.

4. To go back to the original hello and dead interval timers on interface **so-0/0/1**, enter:

```

[edit]
user@host# edit protocols ospf area 0.0.0.0 interface so-0/0/1
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# set hello-interval 5 dead-interval 20
[edit protocols ospf area 0.0.0.0 interface so-0/0/1]
user@host# exit
[edit]
user@host# show
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/1 {
        hello-interval 5;
        dead-interval 20;
      }
    }
  }
}
[edit]
user@host#

```

5. You also can recreate the other interface, as you had it before, with only a single entry:

```

[edit]
user@host# set protocols ospf area 0.0.0.0 interface so-0/0/1 hello-interval 5
dead-interval 20
[edit]
user@host# show
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/0 {

```

```
        hello-interval 5;
        dead-interval 20;
    }
    interface so-0/0/1 {
        hello-interval 5;
        dead-interval 20;
    }
}
[edit]
user@host#
```

- Related Documentation**
- [Getting Started with the Junos OS Command-Line Interface on page 29](#)
 - [Displaying the Junos OS CLI Command and Word History on page 54](#)
 - [Interface Naming Conventions Used in the Junos OS Operational Commands on page 172](#)

Rolling Back Junos OS Configuration Changes

This topic shows how to use the **rollback** command to return to the most recently committed Junos OS configuration. The **rollback** command is useful if you make configuration changes and then decide not to keep the changes.

The following procedure shows how to configure an SNMP health monitor on a device running Junos OS and then return to the most recently committed configuration that does not include the health monitor. When configured, the SNMP health monitor provides the network management system (NMS) with predefined monitoring for file system usage, CPU usage, and memory usage on the device.

1. Enter configuration mode:

```
user@host> configure
entering configuration mode
[edit]
user@host#
```

2. Show the current configuration (if any) for SNMP:

```
[edit]
user@host# show snmp
```

No **snmp** statements appear because SNMP has not been configured on the device.

3. Configure the health monitor:

```
[edit]
user@host# set snmp health-monitor
```

4. Show the new configuration:

```
[edit]
user@host# show snmp
health-monitor;
```

The **health-monitor** statement indicates that SNMP health monitoring is configured on the device.

5. Enter the **rollback** configuration mode command to return to the most recently committed configuration:

```
[edit]
user@host# rollback
load complete
```

6. Show the configuration again to make sure your change is no longer present:

```
[edit]
user@host# show snmp
```

No **snmp** configuration statements appear. The health monitor is no longer configured.

7. Enter the **commit** command to activate the configuration to which you rolled back:

```
[edit]
user@host# commit
```

8. Exit configuration mode:

```
[edit]
user@host# exit
Exiting configuration mode
```

You can also use the **rollback** command to return to earlier configurations.

Related Documentation

- [Returning to the Most Recently Committed Junos OS Configuration on page 134](#)

CHAPTER 3

Getting Online Help

- [Getting Online Help from the Junos OS Command-Line Interface on page 47](#)
- [Junos OS CLI Online Help Features on page 50](#)
- [Examples: Using Command Completion in Configuration Mode on page 51](#)
- [Examples: Using the Junos OS CLI Command Completion on page 53](#)
- [Displaying the Junos OS CLI Command and Word History on page 54](#)

Getting Online Help from the Junos OS Command-Line Interface

The Junos OS command-line interface (CLI) has a context-sensitive online help feature that enables you to access information about commands and statements from the Junos OS CLI. This topic contains the following sections:

- [Getting Help About Commands on page 47](#)
- [Getting Help About a String in a Statement or Command on page 48](#)
- [Getting Help About Configuration Statements on page 49](#)
- [Getting Help About System Log Messages on page 49](#)

Getting Help About Commands

Information about commands is provided at each level of the CLI command hierarchy. You can type a question mark to get help about commands:

- If you type the question mark at the command-line prompt, the CLI lists the available commands and options. For example, to view a list of top-level operational mode commands, type a question mark (?) at the command-line prompt.

```
user@host> ?
Possible completions:
clear          Clear information in the system
configure      Manipulate software configuration information
file           Perform file operations
help           Provide help information
mtrace         Trace mtrace packets from source to receiver.
monitor        Real-time debugging
ping           Ping a remote target
quit           Exit the management session
request        Make system-level requests
restart        Restart a software process
set            Set CLI properties, date, time, craft display text
```

```
show      Show information about the system
ssh       Open a secure shell to another host
start     Start a software process
telnet    Telnet to another host
test      Diagnostic debugging commands
traceroute Trace the route to a remote host
user@host>
```

- If you type the question mark after entering the complete name of a command or command option, the CLI lists the available commands and options and then redisplay the command names and options that you typed.

```
user@host> clear ?
Possible completions:
arp        Clear address-resolution information
bgp        Clear BGP information
chassis    Clear chassis information
firewall   Clear firewall counters
igmp       Clear IGMP information
interfaces Clear interface information
ilmi       Clear ILMI statistics information
isis       Clear IS-IS information
ldp        Clear LDP information
log        Clear contents of a log file
mpls       Clear MPLS information
msdp       Clear MSDP information
multicast  Clear Multicast information
ospf       Clear OSPF information
pim        Clear PIM information
rip        Clear RIP information
route      Clear routing table information
rsvp       Clear RSVP information
snmp       Clear SNMP information
system     Clear system status
vrrp       Clear VRRP statistics information
user@host> clear
```

- If you type the question mark in the middle of a command name, the CLI lists possible command completions that match the letters you have entered so far. It then redisplay the letters that you typed. For example, to list all operational mode commands that start with the letter *c*, type the following:

```
user@host> c?
Possible completions:
clear      Clear information in the system
configure  Manipulate software configuration information
user@host> c
```

- For introductory information on using the question mark or the help command, you can also type **help** and press Enter:

```
user@host> help
```

Getting Help About a String in a Statement or Command

You can use the **help** command to display help about a text string contained in a statement or command name:

```
help apropos string
```


string is a text string about which you want to get help. This string is used to match statement or command names as well as to match the help strings that are displayed for the statements or commands.

If the string contains spaces, enclose it in quotation marks (" "). You can also specify a regular expression for the string, using standard UNIX-style regular expression syntax.

For statements or commands which need input data type as STRING, the supported characters set are as follows:

- Any printable ASCII characters
- For characters with space, it should be enclosed in double-quotes
- To have double-quote as the input, it should be escaped with '\'



NOTE: No escape characters are supported in a string other than to escape from double quotes.

Range of supported characters for attributes is 0 through 65499 characters.

Range of supported characters for string type identifiers is 1 through 255 characters.

In configuration mode, this command displays statement names and help text that match the string specified. In operational mode, this command displays command names and help text that match the string specified.

Getting Help About Configuration Statements

You can display help based on text contained in a statement name using the **help topic** and **help reference** commands:

help topic *word*
help reference *statement-name*

The **help topic** command displays usage guidelines for the statement based on information that appears in the Junos OS configuration guides. The **help reference** command displays summary information about the statement based on the summary descriptions that appear in the Junos OS configuration guides.

Getting Help About System Log Messages

You can display help based on a system log tag using the **help syslog** command:

help syslog *syslog-tag*

The **help syslog** command displays the contents of a system log message.

Related Documentation

- [Junos OS CLI Online Help Features on page 50](#)
- [Getting Started with the Junos OS Command-Line Interface on page 29](#)

Junos OS CLI Online Help Features

The Junos OS CLI online help provides the following features for ease of use and error prevention:

- [Help for Omitted Statements on page 50](#)
- [Using CLI Command Completion on page 50](#)
- [Using Command Completion in Configuration Mode on page 51](#)
- [Displaying Tips About CLI Commands on page 51](#)

Help for Omitted Statements

If you have omitted a required statement at a particular hierarchy level, when you attempt to move from that hierarchy level or when you issue the **show** command in configuration mode, a message indicates which statement is missing. For example:

```
[edit protocols pim interface so-0/0/0]
user@host# top
Warning: missing mandatory statement: 'mode'
[edit]
user@host# show
protocols {
  pim {
    interface so-0/0/0 {
      priority 4;
      version 2;
      # Warning: missing mandatory statement(s): 'mode'
    }
  }
}
```

Using CLI Command Completion

The Junos OS CLI provides you a command completion option that enables Junos OS to recognize commands and options based on the initial few letters you typed. That is, you do not always have to remember or type the full command or option name for the CLI to recognize it.

- To display all possible command or option completions, type the partial command followed immediately by a question mark.
- To complete a command or option that you have partially typed, press Tab or the Spacebar. If the partially typed letters begin a string that uniquely identifies a command, the complete command name appears. Otherwise, a prompt indicates that you have entered an ambiguous command, and the possible completions are displayed.

Command completion also applies to other strings, such as filenames, interface names, and usernames. To display all possible values, type a partial string followed immediately by a question mark. To complete a string, press Tab.

Using Command Completion in Configuration Mode

The CLI command completion functions also apply to the commands in configuration mode and to configuration statements. Specifically, to display all possible commands or statements, type the partial string followed immediately by a question mark. To complete a command or statement that you have partially typed, press Tab or the Spacebar.

Command completion also applies to identifiers, with one slight difference. To display all possible identifiers, type a partial string followed immediately by a question mark. To complete an identifier, you must press Tab. This scheme allows you to enter identifiers with similar names; then press the Spacebar when you are done typing the identifier name.

Displaying Tips About CLI Commands

To get tips about CLI commands, issue the **help tip cli** command. Each time you enter the command, a new tip appears. For example:

```
user@host> help tip cli
Junos tip:
Use 'request system software validate' to validate the incoming software
against the current configuration without impacting the running system.
user@host> help tip cli
Junos tip:
Use 'commit and-quit' to exit configuration mode after the commit has
succeeded. If the commit fails, you are left in configuration mode.
```

You can also enter **help tip cli *number*** to associate a tip with a number. This enables you to recall the tip at a later time. For example:

```
user@host> help tip cli 10
JUNOS tip:
Use '#' in the beginning of a line in command scripts to cause the
rest of the line to be ignored.

user@host> help tip cli
JUNOS tip:
Use the 'apply-groups' statement at any level of the configuration
hierarchy to inherit configuration statements from a configuration group.

user@host>
```

Related Documentation

- [Getting Started with the Junos OS Command-Line Interface on page 29](#)
- [Examples: Using the Junos OS CLI Command Completion on page 53](#)

Examples: Using Command Completion in Configuration Mode

List the configuration mode commands:

```
[edit]
user@host# ?
  <[Enter]>      Execute this command
  activate      Remove the inactive tag from a statement
```

annotate	Annotate the statement with a comment
commit	Commit current set of changes
copy	Copy a statement
deactivate	Add the inactive tag to a statement
delete	Delete a data element
edit	Edit a sub-element
exit	Exit from this level
extension	Extension operations
help	Provide help information
insert	Insert a new ordered data element
load	Load configuration from ASCII file
quit	Quit from this level
rename	Rename a statement
replace	Replace character string in configuration
rollback	Roll back to previous committed configuration
run	Run an operational-mode command
save	Save configuration to ASCII file
set	Set a parameter
show	Show a parameter
status	Show users currently editing configuration
top	Exit to top level of configuration
up	Exit one level of configuration
wildcard	Wildcard operations

[edit]user@host#

List all the statements available at a particular hierarchy level:

```
[edit]
user@host# edit ?
Possible completions:
> accounting-options  Accounting data configuration
> chassis             Chassis configuration
> class-of-service    Class-of-service configuration
> firewall            Define a firewall configuration
> forwarding-options  Configure options to control packet sampling
> groups              Configuration groups
> interfaces          Interface configuration
> policy-options      Routing policy option configuration
> protocols            Routing protocol configuration
> routing-instances   Routing instance configuration
> routing-options     Protocol-independent routing option configuration
> snmp                Simple Network Management Protocol
> system              System parameters

user@host# edit protocols ?
Possible completions:
<[Enter]>            Execute this command
> bgp                 BGP options
> connections         Circuit cross-connect configuration
> dvmrp               DVMRP options
> igmp                IGMP options
> isis                IS-IS options
> ldp                 LDP options
> mpls                Multiprotocol Label Switching options
> msdp                MSDP options
> ospf                OSPF configuration
> pim                 PIM options
> rip                 RIP options
> router-discovery    ICMP router discovery options
> rsvp                RSVP options
> sapSession          Advertisement Protocol options
```

```

> vrrp                VRRP options
|                    Pipe through a command

[edit]
user@host# edit protocols
List all commands that start with a particular letter or string:

user@host# edit routing-options a?
Possible completions:
> aggregate           Coalesced routes
> autonomous-system   Autonomous system number

[edit]
user@host# edit routing-options a
List all configured Asynchronous Transfer Mode (ATM) interfaces:

[edit]
user@host# edit interfaces at?
<interface_name>      Interface name
  at-0/2/0             Interface name
  at-0/2/1             Interface name
[edit]
user@host# edit interfaces at

Display a list of all configured policy statements:

[edit]
user@host# show policy-options policy-statement ?
<policy_name>         Name to identify a policy filter
user@host# show policy-options policy-statement
  <policy_name>       Name to identify a policy filter
  lo0only-v4          Name to identify a policy filter
  lo0only-v6          Name to identify a policy filter
  lo2bgp              Name to identify a policy filter

```

- Related Documentation**
- [Examples: Using the Junos OS CLI Command Completion on page 53](#)
 - [Displaying the Junos OS CLI Command and Word History on page 54](#)

Examples: Using the Junos OS CLI Command Completion

The following examples show how you can use the command completion feature in Junos OS. Issue the **show interfaces** command:

```

user@host> sh<Space>ow i<Space>
'i' is ambiguous.
Possible completions:
igmp           Show information about IGMP
interface      Show interface information
isis          Show information about IS-IS

user@host> show in<Space>terfaces
Physical interface: at-0/1/0, Enabled, Physical link is Up
Interface index: 11, SNMP ifIndex: 65
Link-level type: ATM-PVC, MTU: 4482, Clocking: Internal, SONET mode
Speed: OC12, Loopback: None, Payload scrambler: Enabled
Device flags: Present Running
Link flags: 0x01
...

```

```
user@host>
```

Display a list of all log files whose names start with the string “messages,” and then display the contents of one of the files:

```
user@myhost> show log mes?
```

Possible completions:

```
<filename>Log file to display
messagesSize: 1417052, Last changed: Mar 3 00:33
messages.0.gzSize: 145575, Last changed: Mar 3 00:00
messages.1.gzSize: 134253, Last changed: Mar 2 23:00
messages.10.gzSize: 137022, Last changed: Mar 2 14:00
messages.2.grSize: 137112, Last changed: Mar 2 22:00
messages.3.gzSize: 121633, Last changed: Mar 2 21:00
messages.4.gzSize: 135715, Last changed: Mar 2 20:00
messages.5.gzSize: 137504, Last changed: Mar 2 19:00
messages.6.gzSize: 134591, Last changed: Mar 2 18:00
messages.7.gzSize: 132670, Last changed: Mar 2 17:00
messages.8.gzSize: 136596, Last changed: Mar 2 16:00
messages.9.gzSize: 136210, Last changed: Mar 2 15:00
```

```
user@myhost> show log mes<Tab>sages.4<Tab>.gz<Enter>
Jan 15 21:00:00 myhost newsyslog[1381]: logfile turned over
...
```

**Related
Documentation**

- [Displaying the Junos OS CLI Command and Word History on page 54](#)

Displaying the Junos OS CLI Command and Word History

To display a list of recent commands that you issued, use the **show cli history** command:

```
user@host> show cli history 3
01:01:44 -- show bgp next-hop-database
01:01:51 -- show cli history
01:02:51 -- show cli history 3
```

You can press Esc+. (period) or Alt+. (period) to insert the last word of the previous command. Repeat Esc+. or Alt+. to scroll backwards through the list of recently entered words. For example:

```
user@host> show interfaces terse fe-0/0/0
Interface      Admin  Link  Proto  Local  Remote
fe-0/0/0        up    up
fe-0/0/0.0      up    up    inet   192.168.220.1/30

user@host> <Esc>
user@host> fe-0/0/0
```

If you scroll completely to the beginning of the list, pressing Esc+. or Alt+. again restarts scrolling from the last word entered.

**Related
Documentation**

- [Junos OS CLI Online Help Features on page 50](#)

CHAPTER 4

Using Configuration Statements to Configure a Device

- [Understanding Junos OS CLI Configuration Mode on page 56](#)
- [Entering and Exiting the Junos OS CLI Configuration Mode on page 62](#)
- [Notational Conventions Used in Junos OS Configuration Hierarchies on page 64](#)
- [Forms of the configure Command on page 65](#)
- [Using the configure exclusive Command on page 67](#)
- [Using the configure Command on page 68](#)
- [Modifying the Junos OS Configuration on page 68](#)
- [Adding Junos OS Configuration Statements and Identifiers on page 69](#)
- [Deleting a Statement from a Junos OS Configuration on page 70](#)
- [Example: Deleting a Statement from the Junos OS Configuration on page 71](#)
- [Copying a Junos OS Statement in the Configuration on page 73](#)
- [Example: Copying a Statement in the Junos Configuration on page 73](#)
- [Issuing Relative Junos OS Configuration Mode Commands on page 75](#)
- [Renaming an Identifier in a Junos OS Configuration on page 76](#)
- [Examples: Re-Using Configuration on page 76](#)
- [Inserting a New Identifier in a Junos OS Configuration on page 81](#)
- [Example: Inserting a New Identifier in a Junos Configuration on page 81](#)
- [Example: Using the Wildcard Command with the Range Option on page 85](#)
- [Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration on page 89](#)
- [Example: Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration on page 90](#)
- [Adding Comments in a Junos OS Configuration on page 92](#)
- [Example: Including Comments in a Junos OS Configuration by Using the CLI on page 94](#)
- [Updating the configure private Configuration on page 96](#)
- [Displaying the Current Junos OS Configuration on page 97](#)
- [Example: Displaying the Current Junos OS Configuration on page 98](#)

- [Displaying Additional Information About the Junos OS Configuration on page 99](#)
- [Displaying set Commands from the Junos OS Configuration on page 101](#)
- [Displaying Users Currently Editing the Junos OS Configuration on page 104](#)
- [Verifying a Junos OS Configuration on page 104](#)

Understanding Junos OS CLI Configuration Mode

You can configure all properties of Junos OS, including interfaces, general routing information, routing protocols, and user access, as well as several system hardware properties.

As described in “[Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies](#)” on [page 23](#), a router configuration is stored as a hierarchy of statements. In configuration mode, you create the specific hierarchy of configuration statements that you want to use. When you have finished entering the configuration statements, you commit them, which activates the configuration on the router.

You can create the hierarchy interactively or you can create an ASCII text file that is loaded onto the router or switch and then committed.

This topic covers:

- [Configuration Mode Commands on page 57](#)
- [Configuration Statements and Identifiers on page 58](#)
- [Configuration Statement Hierarchy on page 60](#)

Configuration Mode Commands

Table 4 on page 57 summarizes each CLI configuration mode command. The commands are organized alphabetically.

Table 4: Summary of Configuration Mode Commands

Command	Description
activate	Remove the inactive: tag from a statement, effectively reading the statement or identifier to the configuration. Statements or identifiers that have been activated take effect when you next issue the commit command.
annotate	Add comments to a configuration. You can add comments only at the current hierarchy level.
commit	Commit the set of changes to the database and cause the changes to take operational effect.
copy	Make a copy of an existing statement in the configuration.
deactivate	Add the inactive: tag to a statement, effectively commenting out the statement or identifier from the configuration. Statements or identifiers marked as inactive do not take effect when you issue the commit command.
delete	Delete a statement or identifier. All subordinate statements and identifiers contained within the specified statement path are deleted with it.
edit	Move inside the specified statement hierarchy. If the statement does not exist, it is created.
exit	Exit the current level of the statement hierarchy, returning to the level prior to the last edit command, or exit from configuration mode. The quit and exit commands are synonyms.
extension	Manage configurations that are contributed by SDK application packages. Either display or delete user-defined configuration contributed by the named SDK application package. A configuration defined in any native Junos OS package is never deleted by the extension command.
help	Display help about available configuration statements.
insert	Insert an identifier into an existing hierarchy.
load	Load a configuration from an ASCII configuration file or from terminal input. Your current location in the configuration hierarchy is ignored when the load operation occurs.

Table 4: Summary of Configuration Mode Commands (*continued*)

Command	Description
quit	Exit the current level of the statement hierarchy, returning to the level prior to the last edit command, or exit from configuration mode. The quit and exit commands are synonyms.
rename	Rename an existing configuration statement or identifier.
replace	Replace identifiers or values in a configuration.
rollback	Return to a previously committed configuration. The software saves the last 10 committed configurations, including the rollback number, date, time, and name of the user who issued the commit configuration command.
run	Run a top-level CLI command without exiting from configuration mode.
save	Save the configuration to an ASCII file. The contents of the current level of the statement hierarchy (and below) are saved, along with the statement hierarchy containing it. This allows a section of the configuration to be saved, while fully specifying the statement hierarchy.
set	Create a statement hierarchy and set identifier values. This is similar to edit except that your current level in the hierarchy does not change.
show	Display the current configuration.
status	Display the users currently editing the configuration.
top	Return to the top level of configuration command mode, which is indicated by the [edit] banner.
up	Move up one level in the statement hierarchy.
update	Update a private database.
wildcard	Delete a statement or identifier. All subordinate statements and identifiers contained within the specified statement path are deleted with it. You can use regular expressions to specify a pattern. Based on this pattern, you search for items that contain these patterns and delete them.

Configuration Statements and Identifiers

You can configure router or switch properties by including the corresponding statements in the configuration. Typically, a statement consists of a keyword, which is fixed text, and, optionally, an identifier. An identifier is an identifying name that you can define, such as

the name of an interface or a username, which enables you and the CLI to differentiate among a collection of statements.

Table 5 on page 59 describes top-level CLI configuration mode statements.



NOTE: The QFX3500 switch does not support the IS-IS, OSPF, BGP, LDP, MPLS, and RSVP protocols.

Table 5: Configuration Mode Top-Level Statements

Statement	Description
access	Configure the Challenge Handshake Authentication Protocol (CHAP). For information about the statements in this hierarchy, see the <i>Junos OS Administration Library</i> .
accounting-options	Configure accounting statistics data collection for interfaces and firewall filters. For information about the statements in this hierarchy, see the <i>Network Management Administration Guide</i> .
chassis	Configure properties of the router chassis, including conditions that activate alarms and SONET/SDH framing and concatenation properties. For information about the statements in this hierarchy, see the <i>Junos OS Administration Library</i> .
class-of-service	Configure class-of-service parameters. For information about the statements in this hierarchy, see the Junos OS Class of Service Feature Guide for Routing Devices .
firewall	Define filters that select packets based on their contents. For information about the statements in this hierarchy, see the <i>Routing Policies, Firewall Filters, and Traffic Policers Feature Guide</i> .
forwarding-options	Define forwarding options, including traffic sampling options. For information about the statements in this hierarchy, see the <i>Junos OS Network Interfaces Library for Routing Devices</i> .
groups	Configure configuration groups. For information about statements in this hierarchy, see the <i>Junos OS Administration Library</i> .
interfaces	Configure interface information, such as encapsulation, interfaces, virtual channel identifiers (VCIs), and data-link connection identifiers (DLCIs). For information about the statements in this hierarchy, see the <i>Junos OS Network Interfaces Library for Routing Devices</i> .
policy-options	Define routing policies, which allow you to filter and set properties in incoming and outgoing routes. For information about the statements in this hierarchy, see the <i>Routing Policies, Firewall Filters, and Traffic Policers Feature Guide</i> .

Table 5: Configuration Mode Top-Level Statements (*continued*)

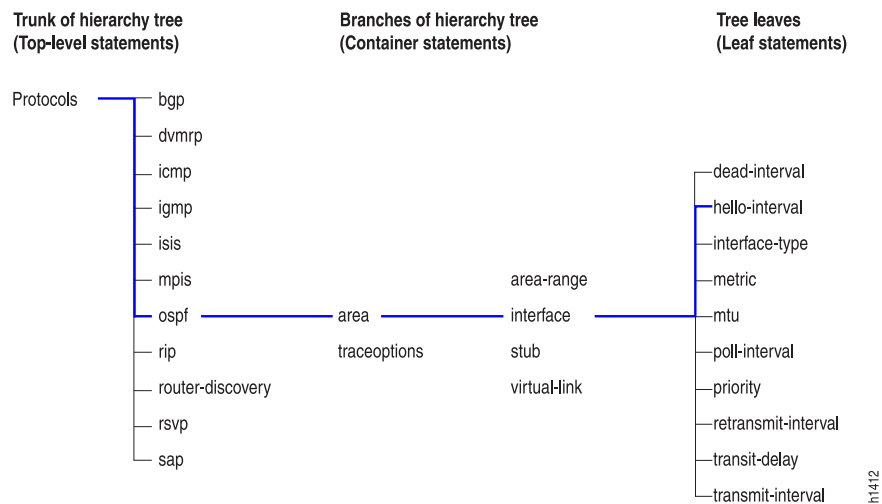
Statement	Description
protocols	Configure routing protocols, including BGP, IS-IS, LDP, MPLS, OSPF, RIP, and RSVP. For information about the statements in this hierarchy, see the chapters that discuss how to configure the individual routing protocols in the <i>Junos OS Routing Protocols Library</i> and the MPLS Applications Feature Guide for Routing Devices .
routing-instances	Configure multiple routing instances. For information about the statements in this hierarchy, see the <i>Junos OS Routing Protocols Library</i> .
routing-options	Configure protocol-independent routing options, such as static routes, autonomous system numbers, confederation members, and global tracing (debugging) operations to log. For information about the statements in this hierarchy, see the <i>Junos OS Routing Protocols Library</i> .
security	Configure IP Security (IPsec) services. For information about the statements in this hierarchy see the <i>Junos OS Administration Library</i> .
snmp	Configure SNMP community strings, interfaces, traps, and notifications. For information about the statements in this hierarchy, see the <i>Network Management Administration Guide</i> .
system	Configure systemwide properties, including the hostname, domain name, Domain Name System (DNS) server, user logins and permissions, mappings between hostnames and addresses, and software processes. For information about the statements in this hierarchy, see the <i>Junos OS Administration Library</i> .

For specific information on configuration statements, see the Junos OS configuration guides.

Configuration Statement Hierarchy

The Junos OS configuration consists of a hierarchy of *statements*. There are two types of statements: *container statements*, which are statements that contain other statements, and *leaf statements*, which do not contain other statements (see [Figure 4 on page 61](#)). All of the container and leaf statements together form the *configuration hierarchy*.

Figure 4: Configuration Mode Hierarchy of Statements



Each statement at the top level of the configuration hierarchy resides at the trunk (or root level) of a hierarchy tree. The top-level statements are container statements, containing other statements that form the tree branches. The leaf statements are the leaves of the hierarchy tree. An individual hierarchy of statements, which starts at the trunk of the hierarchy tree, is called a *statement path*. [Figure 4 on page 61](#) illustrates the hierarchy tree, showing a statement path for the portion of the protocol configuration hierarchy that configures the hello interval on an interface in an OSPF area.

The **protocols** statement is a top-level statement at the trunk of the configuration tree. The **ospf**, **area**, and **interface** statements are all subordinate container statements of a higher statement (they are branches of the hierarchy tree); and the **hello-interval** statement is a leaf on the tree which in this case contains a data value: the length of the hello interval, in seconds.

The CLI represents the statement path shown in [Figure 4 on page 61](#) as **[edit protocols ospf area *area-number* interface *interface-name*]** and displays the configuration as follows:

```
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/0 {
        hello-interval 5;
      }
      interface so-0/0/1 {
        hello-interval 5;
      }
    }
  }
}
```

The CLI indents each level in the hierarchy to indicate each statement's relative position in the hierarchy and generally sets off each level with braces, using an open brace at the beginning of each hierarchy level and a closing brace at the end. If the statement at a hierarchy level is empty, the braces are not printed.

Each leaf statement ends with a semicolon. If the hierarchy does not extend as far as a leaf statement, the last statement in the hierarchy ends with a semicolon.

The configuration hierarchy can also contain “oneliners” at the last level in the hierarchy. Oneliners remove one level of braces in the syntax and display the container statement, its identifiers, the child or leaf statement and its attributes all on one line. For example, in the following sample configuration hierarchy, the line **level 1 metric 10** is a oneliner because the **level** container statement with identifier **1**, its child statement **metric**, and its corresponding attribute **10** all appear on a single line in the hierarchy:

```
[edit protocols]
isis {
  interface ge-0/0/0.0 {
    level 1 metric 10;
  }
}
```

Likewise, in the following example, **dynamic-profile *dynamic-profile-name* aggregate-clients;** is a oneliner because the **dynamic-profile** statement, its identifier ***dynamic-profile-name***, and leaf statement **aggregate-clients** all appear on one line when you run the **show** command in the configuration mode:

```
[edit forwarding-options]
user@host# show
dhcp-relay {
  dynamic-profile dynamic-profile-name aggregate-clients;
}
```

**Related
Documentation**

- [Entering and Exiting the Junos OS CLI Configuration Mode on page 62](#)

Entering and Exiting the Junos OS CLI Configuration Mode

You configure Junos OS by entering configuration mode and creating a hierarchy of configuration mode statements.

- To enter configuration mode, use the **configure** command.

When you enter configuration mode, the following configuration mode commands are available:

```
user@host>configure
entering configuration mode

[edit]
user@host#?
possible completions:
<[Enter]>      Execute this command
activate       Remove the inactive tag from a statement
annotate       Annotate the statement with a comment
commit         Commit current set of changes
copy           Copy a statement
deactivate     Add the inactive tag to a statement
delete         Delete a data element
edit           Edit a sub-element
```

exit	Exit from this level
help	Provide help information
insert	Insert a new ordered data element
load	Load configuration from ASCII file
quit	Quit from this level
rename	Rename a statement
replace	Replace character string in configuration
rollback	Roll back to previous committed configuration
run	Run an operational-mode command
save	Save configuration to ASCII file
set	Set a parameter
show	Show a parameter
status	Show users currently editing configuration
top	Exit to top level of configuration
up	Exit one level of configuration
wildcard	Wildcard operations

[edit]
user@host>

Users must have configure permission to view and use the **configure** command. When in configuration mode, a user can view and modify only those statements for which they have access privileges set. For more information, see the *Junos OS Administration Library*.

- If you enter configuration mode and another user is also in configuration mode, a message shows the user's name and what part of the configuration the user is viewing or editing:

```
user@host> configure
Entering configuration mode
Users currently editing the configuration:
  root terminal d0 (pid 4137) on since 2008-04-09 23:03:07 PDT, idle 7w6d 08:22
```

```
[edit]
The configuration has been changed but not committed
```

```
[edit]
user@host#
```

Up to 32 users can be in configuration mode simultaneously, and they all can make changes to the configuration at the same time.

- To exit configuration mode, use the **exit configuration-mode** configuration mode command from any level, or use the **exit** command from the top level. For example:

```
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host# exit configuration-mode
exiting configuration mode
user@host>

[edit]
user@host# exit
exiting configuration mode
user@host>
```

If you try to exit from configuration mode using the **exit** command and the configuration contains changes that have not been committed, you see a message and prompt:

```
[edit]
user@host# exit
```

```
The configuration has been changed but not committed
Exit with uncommitted changes? [yes,no] (yes) <Enter>
Exiting configuration mode
user@host>
```

- To exit with uncommitted changes without having to respond to a prompt, use the **exit configuration-mode** command. This command is useful when you are using scripts to perform remote configuration.

```
[edit]
user@host# exit configuration-mode
The configuration has been changed but not committed
Exiting configuration mode
user@host>
```

Related Documentation

- [Understanding Junos OS CLI Configuration Mode on page 56](#)
- [Modifying the Junos OS Configuration on page 68](#)
- [Commit Operation When Multiple Users Configure the Software on page 111](#)
- [Displaying the Current Junos OS Configuration on page 97](#)
- [Displaying set Commands from the Junos OS Configuration on page 101](#)
- [Issuing Relative Junos OS Configuration Mode Commands on page 75](#)
- [Using the configure exclusive Command on page 67](#)
- [Updating the configure private Configuration on page 96](#)
- [Switching Between Junos OS CLI Operational and Configuration Modes on page 31](#)

Notational Conventions Used in Junos OS Configuration Hierarchies

When you are working in Junos OS command-line interface (CLI) configuration mode, the banner on the line preceding the prompt indicates the current hierarchy level. In the following example, the level is **[edit protocols ospf]**:

```
[edit protocols ospf]
user@host#
```

(The Junos OS documentation uses **user@host#** as the standard configuration mode prompt. In an actual CLI session, the prompt shows your user ID and the name of the Juniper Networks device you are working on.)

Use the **set ?** command to display the statements that you can include in the configuration at the current level. The **help apropos** command is also context-sensitive, displaying matching statements only at the current level and below.



NOTE: In this document, statements are listed alphabetically within each hierarchy and subhierarchy. If a subhierarchy is sufficiently long that it might be difficult to determine where it ends and its next peer statement begins, the subhierarchy appears at the end of its parent hierarchy instead of in alphabetical order. In this case, a placeholder appears in its actual alphabetical position.

For example, at the [edit interfaces *interface-name* unit *logical-unit-number*] hierarchy level, the family *family-name* subhierarchy has more than 20 child statements, including several subhierarchies with child statements of their own. The full family *family-name* hierarchy appears at the end of its parent hierarchy ([edit interfaces *interface-name* unit *logical-unit-number*]), and the following placeholder appears at its actual alphabetical position:

```
family family-name {
    ... the family subhierarchy appears after the main [edit interfaces interface-name
    unit logical-unit-number] hierarchy ...
}
```

Another exception to alphabetical order is that the **disable** statement always appears first in any hierarchy that includes it.

- Related Documentation**
- *Configuration Features in the Junos OS*
 - *Configuration Mode Commands in the Junos OS*

Forms of the configure Command

The Junos OS supports three forms of the **configure** command: **configure**, **configure private**, and **configure exclusive**. These forms control how users edit and commit configurations and can be useful when multiple users configure the software. See [Table 6 on page 66](#).

Table 6: Forms of the configure Command

Command	Edit Access	Commit Access
configure	<ul style="list-style-type: none"> No one can lock the configuration. All users can make configuration changes. <p>When you enter configuration mode, the CLI displays the following information:</p> <ul style="list-style-type: none"> A list of other users editing the configuration. Hierarchy levels the users are viewing or editing. Whether the configuration has been changed, but not committed. When multiple users enter conflicting configurations, the most recent change to be entered takes precedence. 	<ul style="list-style-type: none"> No one can lock the configuration. All users can commit all changes to the configuration. If you and another user make changes and the other user commits changes, your changes are committed as well.
configure exclusive	<ul style="list-style-type: none"> One user locks the configuration and makes changes without interference from other users. Other users can enter and exit configuration mode, but they cannot commit the configuration. If you enter configuration mode while another user has locked the configuration (with the configure exclusive command), the CLI displays the user and the hierarchy level the user is viewing or editing. If you enter configuration mode while another user has locked the configuration, you can forcibly log out that user with the request system logout operational mode command. For details, see the CLI Explorer. 	
configure private	<ul style="list-style-type: none"> Multiple users can edit the configuration at the same time. Each user has a private candidate configuration to edit independently of other users. When multiple users enter conflicting configurations, the first commit operation takes precedence over subsequent commit operations. 	<ul style="list-style-type: none"> When you commit the configuration, the router verifies that the operational (running) configuration has not been modified by another user before accepting your private candidate configuration as the new operational configuration. If the configuration has been modified by another user, you can merge the modifications into your private candidate configuration and attempt to commit again.

- Related Documentation**
- [Committing a Junos OS Configuration on page 108](#)
 - [Using the configure Command on page 68](#)
 - [Displaying Users Currently Editing the Junos OS Configuration on page 104](#)
 - [Using the configure exclusive Command on page 67](#)
 - [Updating the configure private Configuration on page 96](#)
 - [Displaying set Commands from the Junos OS Configuration on page 101](#)

Using the configure exclusive Command

If you enter configuration mode with the **configure exclusive** command, you lock the candidate *global* configuration (also known as the *shared configuration* or *shared configuration database*) for as long as you remain in configuration mode, allowing you to make changes without interference from other users. Other users can enter and exit configuration mode, but they cannot commit the configuration.

If another user has locked the configuration, and you need to forcibly log the person out, enter the operational mode command **request system logout pid *pid_number***.

If you enter configuration mode and another user is also in configuration mode and has locked the configuration, a message identifies the user and the portion of the configuration that the user is viewing or editing:

```
user@host> configure
Entering configuration mode
Users currently editing the configuration:
root terminal p3 (pid 1088) on since 2000-10-30 19:47:58 EDT, idle 00:00:44
exclusive [edit interfaces so-3/0/0 unit 0 family inet]
```

In configure exclusive mode, any uncommitted changes are discarded when you exit:

```
user@host> configure exclusive
warning: uncommitted changes will be discarded on exit
Entering configuration mode
[edit]
user@host# set system host-name cool
[edit]
user@host# quit
The configuration has been changed but not committed
warning: Auto rollback on exiting 'configure exclusive'
Discard uncommitted changes? [yes,no] (yes)
warning: discarding uncommitted changes
load complete
Exiting configuration mode
```

When you use the **yes** option to exit configure exclusive mode, Junos OS discards your uncommitted changes and rolls back your configuration. The **no** option allows you to continue editing or to commit your changes in configure exclusive mode.

When a user exits from configure exclusive mode while another user is in configure private mode, Junos OS will roll back any uncommitted changes.

- Related Documentation**
- [Adding Junos OS Configuration Statements and Identifiers on page 69](#)
 - [Forms of the configure Command on page 65](#)

Using the configure Command

You can use the **configure** command to not only enter the CLI configuration mode but also to gather other information, such as other users currently in configuration mode.

Up to 32 users can be in configuration mode simultaneously, and they all can make changes to the configuration at the same time. When you commit changes to the configuration, you may be committing a combination of changes you and other users have made. For this reason, you will want to keep track on who if anyone is in configuration mode with you.

To see other users currently logged onto the same device in configuration mode:

- Use the **configure** command to enter the CLI configuration mode.

If there are other users, the message displayed indicates who the users are and what portion of the configuration the each person is viewing or editing.

```
user@host> configure
Entering configuration mode
Current configuration users:
root terminal p3 (pid 1088) on since 1999-05-13 01:03:27 EDT
[edit interfaces so-3/0/0 unit 0 family inet]
The configuration has been changed but not committed
[edit]
user@host#
```

Notice also that If, when you enter configuration mode, the configuration contains changes that have not been committed, another message is displayed:

```
user@host> configure
Entering configuration mode
The configuration has been changed but not committed
[edit]
user@host#
```

This tells you that another user has already made changes to the configuration.

- Related Documentation**
- [Forms of the configure Command on page 65](#)

Modifying the Junos OS Configuration

To configure a device running Junos OS or to modify an existing Junos OS configuration, you add statements to the configuration. For each statement hierarchy, you create the hierarchy starting with a statement at the top level and continuing with statements that move progressively lower in the hierarchy.

To modify the hierarchy, you use two configuration mode commands:

- **edit**—Moves to a particular hierarchy level. If that hierarchy level does not exist, the **edit** command creates it. The **edit** command has the following syntax:

```
edit <statement-path>
```

- **set**—Creates a configuration statement and sets identifier values. After you issue a **set** command, you remain at the same level in the hierarchy. The **set** command has the following syntax:

```
set <statement-path> statement <identifier>
```

statement-path is the hierarchy to the configuration statement and the statement itself. If you have already moved to the statement's hierarchy level, you can omit the statement path. **statement** is the configuration statement itself. **identifier** is a string that identifies an instance of a statement.

You cannot use the **edit** command to change the value of identifiers. You must use the **set** command.

Related Documentation

- [Displaying the Current Junos OS Configuration on page 97](#)
- [Adding Junos OS Configuration Statements and Identifiers on page 69](#)
- [Using the configure exclusive Command on page 67](#)
- [Updating the configure private Configuration on page 96](#)
- [Issuing Relative Junos OS Configuration Mode Commands on page 75](#)

Adding Junos OS Configuration Statements and Identifiers

All properties of a device running Junos OS are configured by including *statements* in the configuration. A statement consists of a keyword, which is fixed text, and, optionally, an *identifier*. An identifier is an identifying name which you define, such as the name of an interface or a username, and which allows you and the CLI to discriminate among a collection of statements.

For example, the following list shows the statements available at the top level of configuration mode:

```
user@host# set?
Possible completions:
> accounting-options  Accounting data configuration
+ apply-groups        Groups from which to inherit configuration data
> chassis             Chassis configuration
> class-of-service    Class-of-service configuration
> firewall            Define a firewall configuration
> forwarding-options  Configure options to control packet sampling
> groups              Configuration groups
> interfaces          Interface configuration
> policy-options      Routing policy option configuration
> protocols           Routing protocol configuration
> routing-instances   Routing instance configuration
> routing-options     Protocol-independent routing option configuration
> snmp               Simple Network Management Protocol
> system              System parameters
```

An angle bracket (>) before the statement name indicates that it is a container statement and that you can define other statements at levels below it. If there is no angle bracket (>) before the statement name, the statement is a leaf statement; you cannot define other statements at hierarchy levels below it.

A plus sign (+) before the statement name indicates that it can contain a set of values. To specify a set, include the values in brackets. For example:

```
[edit]
user@host# set policy-options community my-as1-transit members [65535:10 65535:11]
```

In some statements, you can include an identifier. For some identifiers, such as interface names, you must specify the identifier in a precise format. For example, the interface name so-0/0/0 refers to a SONET/SDH interface that is on the Flexible PIC Concentrator (FPC) in slot 0, in the first PIC location, and in the first port on the Physical Interface Card (PIC). For other identifiers, such as interface descriptive text and policy and firewall term names, you can specify any name, including special characters, spaces, and tabs.

You must enclose in quotation marks (double quotes) identifiers and any strings that include a space or tab character or any of the following characters:

() [] { } ! @ # \$ % ^ & | ' = ?

If you do not type an option for a statement that requires one, a message indicates the type of information required. In this example, you need to type an area number to complete the command:

```
[edit]
user@host# set protocols ospf area<Enter>
^
syntax error, expecting <identifier>
```

Related Documentation

- [Modifying the Junos OS Configuration on page 68](#)
- [Deleting a Statement from a Junos OS Configuration on page 70](#)
- [Copying a Junos OS Statement in the Configuration on page 73](#)
- [Renaming an Identifier in a Junos OS Configuration on page 76](#)
- [Using the configure exclusive Command on page 67](#)
- [Additional Details About Specifying Junos OS Statements and Identifiers on page 141](#)
- [Displaying the Current Junos OS Configuration on page 97](#)

Deleting a Statement from a Junos OS Configuration

To delete a statement or identifier from a Junos OS configuration, use the **delete** configuration mode command. Deleting a statement or an identifier effectively "unconfigures" the functionality associated with that statement or identifier, returning that functionality to its default condition.

```
user@host# delete <statement-path> <identifier>
```

When you delete a statement, the statement and all its subordinate statements and identifiers are removed from the configuration.

For statements that can have more than one identifier, when you delete one identifier, only that identifier is deleted. The other identifiers in the statement remain.

To delete the entire hierarchy starting at the current hierarchy level, do not specify a statement or an identifier in the **delete** command. When you omit the statement or identifier, you are prompted to confirm the deletion:

```
[edit]
user@host# delete
Delete everything under this level? [yes, no] (no)
Possible completions:
no    Don't delete everything under this level
yes    Delete everything under this level
Delete everything under this level? [yes, no] (no)
```



NOTE: You cannot delete multiple statements or identifiers within a hierarchy using a single delete command. You must delete each statement or identifier individually using multiple delete commands. For example, consider the following configuration at the [edit system] hierarchy level:

```
system {
  host-name host-211;
  domain-name domain-122;
  backup-router 192.168.71.254;
  arp;
  authentication-order [ radius password tacplus ];
}
```

To delete the domain-name, host-name, and backup-router from the configuration, you cannot issue a single delete command:

```
user@host> delete system hostname host-211 domain-name domain-122 backup-router
192.168.71.254
```

You can only delete each statement individually:

```
user@host delete system host-name host-211
user@host delete system domain-name domain-122
user@host delete system backup-router 192.168.71.254
```

Related Documentation

- [Example: Deleting a Statement from the Junos OS Configuration on page 71](#)
- [Adding Junos OS Configuration Statements and Identifiers on page 69](#)
- [Copying a Junos OS Statement in the Configuration on page 73](#)

Example: Deleting a Statement from the Junos OS Configuration

The following example shows how to delete the **ospf** statement, effectively unconfiguring OSPF on the router:

```
[edit]
```

```
user@host# set protocols ospf area 0.0.0.0 interface so-0/0/0 hello-interval 5
[edit]
user@host# show
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/0 {
        hello-interval 5;
      }
    }
  }
}
[edit]
user@host# delete protocols ospf
[edit]
user@host# show
[edit]
user@host#
```

Delete all statements from the current level down:

```
[edit]
user@host# edit protocols ospf area 0.0.0.0
[edit protocols ospf area 0.0.0.0]
user@host# set interface so-0/0/0 hello-interval 5
[edit protocols ospf area 0.0.0.0]
user@host# delete
Delete everything under this level? [yes, no] (no) yes
[edit protocols ospf area 0.0.0.0]
user@host# show
[edit]
user@host#
```

Unconfigure a particular property:

```
[edit]
user@host# set interfaces so-3/0/0 speed 100mb
[edit]
user@host# show
interfaces {
  so-3/0/0 {
    speed 100mb;
  }
}
[edit]
user@host# delete interfaces so-3/0/0 speed
[edit]
user@host# show
interfaces {
  so-3/0/0;
}
}
```

- [Example: Using Global Replace in a Junos OS Configuration—Using the upto Option on page 210](#)
- [Deleting a Statement from a Junos OS Configuration on page 70](#)

Copying a Junos OS Statement in the Configuration

When you have many similar statements in a Junos configuration, you can add one statement and then make copies of that statement. Copying a statement duplicates that statement and the entire hierarchy of statements configured under that statement. Copying statements is useful when you are configuring many physical or logical interfaces of the same type.

To make a copy of an existing statement in the configuration, use the configuration mode **copy** command:

```
user@host# copy existing-statement to new-statement
```

Immediately after you have copied a portion of the configuration, the configuration might not be valid. You must check the validity of the new configuration, and if necessary, modify either the copied portion or the original portion for the configuration to be valid.

Related Documentation

- [Example: Copying a Statement in the Junos Configuration on page 73](#)
- [Adding Junos OS Configuration Statements and Identifiers on page 69](#)
- [Examples: Re-Using Configuration on page 76](#)

Example: Copying a Statement in the Junos Configuration

This example shows how you can create one virtual connection (VC) on an interface by copying an existing VC.

- [Requirements on page 73](#)
- [Overview on page 74](#)
- [Configuration on page 74](#)

Requirements

No special configuration beyond device initialization is required before configuring this example.

Before you begin this example, configure the following initial configuration.

```
[edit interfaces]
user@host# show
at-1/0/0 {
  description "PAIX to MAE West"
  encapsulation atm-pvc;
  unit 61 {
    point-to-point;
    vci 0.61;
    family inet {
      address 10.0.1.1/24;
    }
  }
}
```

To quickly configure the *initial configuration* for this example, copy the following commands, paste it into a text file, remove any line breaks and change any details necessary to match your network configuration, copy and paste this command into the CLI at the **[edit]** hierarchy level, and then enter **commit** from configuration mode.

```
set interfaces at-1/0/0 description "PAIX to MAE West"
set interfaces at-1/0/0 encapsulation atm-pvc
set interfaces at-1/0/0 unit 61 point-to-point
set interfaces at-1/0/0 unit 61 vci 0.61
set interfaces at-1/0/0 unit 61 family inet address 10.0.1.1/24
```

Overview

Copying statements is useful when you are configuring many physical or logical interfaces of the same type. You can add one statement and then make copies of that statement. Copying a statement duplicates that statement and the entire hierarchy of statements configured under that statement. In the case of this example, we are adding a virtual connection that is very similar to a virtual connection already configured.

Configuration

CLI Quick Configuration

Start at the **[edit interfaces at-1/0/0]** hierarchy level.

```
copy unit 61 to unit 62
set unit 62 vci 0.62
edit unit 62
replace pattern 10.0.1.1 with 10.0.2.1
```

Configuring by Copying

Step-by-Step Procedure

To configure by copying a configuration:

1. Go to the **[edit interfaces at-1/0/0]** hierarchy level and copy unit 61.

```
[edit interfaces at-1/0/0]
user@host# copy unit 61 to unit 62
```

2. Take a look at the new configuration and see what you need to change to make the configuration valid..

```
user@host# show interfaces at-1/0/0
description "PAIX to MAE West"
encapsulation atm-pvc;
unit 61 {
  point-to-point;
  vci 0.61;
  family inet {
    address 10.0.1.1/24;
  }
}
unit 62 {
  point-to-point;
  vci 0.61;
  family inet {
    address 10.0.1.1/24;
  }
}
```

```
}
```

3. Change the configuration to make it valid.

In this example you want to reconfigure the virtual circuit identifier (VCI) and virtual path identifier (VPI).

```
[edit interfaces at-1/0/0]
user@host# set unit 62 vci 0.62
```

You also want to replace the IP address of the new interface with its own IP address.

```
[edit interfaces at-1/0/0]
user@host# edit unit 62
user@host# replace pattern 10.0.1.1 with 10.0.2.1
```

Results

```
[edit]
show interfaces
at-1/0/0 {
  description "PAIX to MAE West"
  encapsulation atm-pvc;
  unit 61 {
    point-to-point;
    vci 0.61;
    family inet {
      address 10.0.1.1/24;
    }
  }
  unit 62 {
    point-to-point;
    vci 0.62;
    family inet {
      address 10.0.2.1/24;
    }
  }
}
```

Related Documentation • [Copying a Junos OS Statement in the Configuration on page 73](#)

Issuing Relative Junos OS Configuration Mode Commands

The **top** or **up** command followed by another configuration command, including **edit**, **insert**, **delete**, **deactivate**, **annotate**, or **show** enables you to quickly move to the top of the hierarchy or to a level above the area you are configuring.

To issue configuration mode commands from the top of the hierarchy, use the **top** command; then specify a configuration command. For example:

```
[edit interfaces fxp0 unit 0 family inet]
user@host# top edit system login
[edit system login]
user@host#
```

To issue configuration mode commands from a location higher up in the hierarchy, use the **up** configuration mode command; specify the number of levels you want to move up the hierarchy and then specify a configuration command. For example:

```
[edit protocols bgp]
user@host# up 2 activate system
```

**Related
Documentation**

- [Displaying the Current Junos OS Configuration on page 97](#)

Renaming an Identifier in a Junos OS Configuration

When modifying a Junos configuration, you can rename an identifier that is already in the configuration. You can do this either by deleting the identifier (using the **delete** command) and then adding the renamed identifier (using the **set** and **edit** commands), or you can rename the identifier using the **rename** configuration mode command:

```
user@host# rename <statement-path> identifier1 to identifier2
```

**Related
Documentation**

- [Adding Junos OS Configuration Statements and Identifiers on page 69](#)
- [Examples: Re-Using Configuration on page 76](#)
- [Inserting a New Identifier in a Junos OS Configuration on page 81](#)

Examples: Re-Using Configuration

If you need to make changes to the configuration of a device, you can always remove the original configuration settings using the **delete** command and add your new configuration settings using the **set** command. There are, however, other ways of modifying a configuration that are more efficient and easier to use.

This example shows how to use the following configuration mode commands to update an existing configuration:

- **rename**—Rename an existing configuration setting, such as an interface name. This can be useful when you are adding new interfaces to a device.
 - **copy**—Copy a configuration setting and the entire hierarchy of statements configured under that setting. Copying configuration statements is useful when you are configuring many physical or logical interfaces of the same type.
 - **replace**—Make global changes to text patterns in the configuration. For example, if you consistently misspell a word common to the description statement for all of the interfaces on your device, you can fix this mistake with a single command.
- [Requirements on page 77](#)
 - [Overview on page 77](#)
 - [Configuration on page 77](#)

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In the course of the first example in this topic, you will make the following configuration changes:

- Create a new interface with a description that contains a typing error.
- Copy the configuration from the interface that you created to create a new interface.
- Rename one of the interfaces that you created.
- Fix the typing error in the description for the interfaces that you created.

In the second, shorter example, you will experiment with some of the same commands under slightly different circumstances.

Configuration

CLI Quick Configuration

This example does not use commands that are suitable for this section.

Using the Copy, Rename, and Replace Commands to Modify a Loopback Interface Configuration

Step-by-Step Procedure



CAUTION: If your configuration uses any of the loopback interface unit numbers used in this example, you must substitute different loopback interface unit numbers that you are not using in your device's configuration in the following steps to avoid adversely impacting the operational status of your device.

To create and modify a configuration of a loopback interface using the **copy**, **rename**, and **replace** commands:

1. Create a new loopback interface unit number and include a description.
The mistakes in the spelling of *loopback* in the description are intentional.

```
[edit]
user@host# set interfaces lo0 unit 100 description "this is a lopbck interface"
```
2. Display the configuration for the loopback interface you have just added.

```
[edit]
user@host# show interfaces lo0 unit 100
description "this is a lopbck interface";
```
3. Duplicate the loopback interface you have just created, warts and all, from unit 100 to unit 101.

```
[edit]
user@host# copy interfaces lo0 unit 100 to unit 101
```

4. Display the configurations for loopback interfaces lo0 unit 100 and lo0 unit 101.

```
[edit]
user@host# show interfaces lo0 unit 100
description "this is a loppbck interface";
[edit]
user@host# show interfaces lo0 unit 101
description "this is a loppbck interface";
```

The **copy** command duplicates an interface including any child statements such as **description**.

5. Rename the loopback interface lo0 unit 100 to loopback interface lo0 unit 102.

```
[edit]
user@host# rename interfaces lo0 unit 100 to unit 102
```

6. Display the configuration for loopback interface lo0 unit 100.

```
[edit]
user@host# show interfaces lo0 unit 100
[edit]
user@host#
```

You should not see any results from this command. The loopback interface lo0 unit 100 is now gone. The **rename** command replaces the configuration statement indicated with the new configuration.

7. Fix the misspelling of the word *loopback* in the descriptions for loopback interfaces lo0 unit 101 and lo0 unit 102.

```
[edit]
user@host# replace pattern loppbck with loopback
```

8. Display the configuration for loopback interfaces lo0 unit 101 and lo0 102 to verify that the word *loopback* is spelled correctly now.

```
[edit]
user@host# show interfaces lo0 unit 101
description "this is a loopback interface";
[edit]
user@host# show interfaces lo0 unit 102
description "this is a loopback interface";
```

The **replace** command replaces all instances of the pattern specified in the command, unless limited in some way. The next example in this topic shows one way to limit the effect of the **replace** command.

9. From configuration mode, use the **rollback** command to put the device's configuration back to the state it was in before you executed the previous steps.

```
[edit]
user@host# rollback
```

Results From configuration mode, use the **show interfaces lo0 unit 101** and **show interfaces lo0 unit 102** commands to ensure that the device's configuration is back to the state it was in before you executed the steps in this example.

```
[edit]
user@host: show interfaces lo0 unit 101
[edit]
user@host#
```

You should not see any results from this command.

```
[edit]
user@host# show interfaces lo0 unit 102
[edit]
user@host#
```

You should not see any results from this command.

Compare the Copy Command at the Top-Level Configuration Hierarchy Level

Step-by-Step Procedure The previous example shows the **copy**, **rename**, and **replace** commands at the **[edit interfaces interface-name unit logical-interface-number]** hierarchy level. This example shows how some of these commands work at the top level of the CLI configuration mode hierarchy.

The following example requires you to navigate to various levels in the configuration hierarchy. For information about navigating the CLI, see [“Using the CLI Editor in Configuration Mode” on page 34](#) in the *CLI User Guide*.

1. Create an Ethernet interface.

```
[edit]
user@host# set interfaces et-2/0/0 unit 0 family inet address 192.0.2.2
```

2. Copy the interface you just created to another interface.

```
[edit]
user@host# copy interfaces et-2/0/0 to et-2/1/0
```

Compare this **copy** command to the one in Step 3 in the first example, where the **copy** command takes the keyword **unit** before the value to be copied. Notice that the keyword **interfaces** is not repeated after the preposition **to** and before the value to be copied. This happens in some top-level statements with the **copy** command.



TIP: Similarly, in the **rename** command, you do not repeat the keyword part of the statement before the new identifier in some top-level statements.

3. Show your configuration so far.

```
[edit]
user@host# show interfaces
et-2/0/0 {
  unit 0 {
```

```
        family inet {
            address 192.0.2.2/32;
        }
    }
    et-2/1/0 {
        unit 0 {
            family inet {
                address 192.0.2.2/32;
            }
        }
    }
}
```

4. Replace the address for et-2/1/0 with another IP address.

```
[edit interfaces et-2/1/0 unit 0 family inet]
user@host# replace pattern 192.0.2.2 with 192.0.2.40
```

Notice that if you want to change only a specific occurrence of a pattern instead of all of them (as you did in Step 7 in the first example), you need to drill down to that specific hierarchy level before using the **replace** command.

5. Show your interfaces again.

```
[edit]
user@host# show interfaces
et-2/0/0 {
    unit 0 {
        family inet {
            address 192.0.2.2/32;
        }
    }
}
et-2/1/0 {
    unit 0 {
        family inet {
            address 192.0.2.40/32;
        }
    }
}
```

6. From configuration mode, use the **rollback** command to put the device's configuration back to the state it was in before you executed the previous steps.

```
[edit]
user@host# rollback
```

Results From configuration mode, use the **show interfaces et-2/0/0** and **show interfaces et-2/1/0** commands to ensure that the device's configuration is back to the state it was in before you executed the steps in this example.

```
[edit]
user@host# show interfaces et-2/0/0
[edit]
user@host#
```

You should not see any results from this command.


```
[edit]
user@R1# show interfaces et-2/1/0
[edit]
user@host#
```

You should not see any results from this command.

Related Documentation

- [rename on page 287](#)
- [replace on page 288](#)
- [Example: Using Global Replace in a Junos OS Configuration—Using the \n Back Reference on page 206](#)
- [Example: Using Global Replace in a Junos OS Configuration—Using the upto Option on page 210](#)
- [Copying a Junos OS Statement in the Configuration on page 73](#)
- [Example: Copying a Statement in the Junos Configuration on page 73](#)

Inserting a New Identifier in a Junos OS Configuration

When configuring a device running Junos OS, you can enter most statements and identifiers in any order. Regardless of the order in which you enter the configuration statements, the CLI always displays the configuration in a strict order. However, there are a few cases where the ordering of the statements matters because the configuration statements create a sequence that is analyzed in order.

For example, in a routing policy or firewall filter, you define terms that are analyzed sequentially. Also, when you create a named path in dynamic MPLS, you define an ordered list of the transit routers in the path, starting with the first transit router and ending with the last one.

To modify a portion of the configuration in which the statement order matters, use the **insert** configuration mode command:

```
user@host# insert <statement-path> identifier1 (before | after) identifier2
```

If you do not use the **insert** command, but instead simply configure the identifier, it is placed at the end of the list of similar identifiers.

Related Documentation

- [Renaming an Identifier in a Junos OS Configuration on page 76](#)
- [Examples: Re-Using Configuration on page 76](#)
- [Example: Inserting a New Identifier in a Junos Configuration on page 81](#)
- [Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration on page 89](#)

Example: Inserting a New Identifier in a Junos Configuration

This example shows the use of the **insert** command.

Whereas a term added using the **set** command is placed at the end of the existing list of terms, you use the **insert** command to add a term in the order you specify. Specifying the order of statement is important in the cases in which the order of the statements matters because the configuration statements create a sequence that is analyzed in order.

Also notice, as shown in this example, that you must create the term before you can place it using the **insert** command.

- [Requirements on page 82](#)
- [Overview on page 83](#)
- [Configuration on page 83](#)

Requirements

Before you can insert a term, you must configure an initial policy. To quickly configure the initial policy for this example, copy the following commands, paste them into a text file, remove any line breaks and change any details necessary to match your network configuration, copy and paste the commands into the CLI at the **[edit policy-options]** hierarchy level, and then enter **commit** from configuration mode.

```
set policy-statement statics term term1 from route-filter 192.168.0.0/16 orlonger
set policy-statement statics term term1 from route-filter 224.0.0.0/3 orlonger
set policy-statement statics term term1 then reject
set policy-statement statics term term2 from protocol direct
set policy-statement statics term term2 then reject
set policy-statement statics term term3 from protocol static
set policy-statement statics term term3 then reject
set policy-statement statics term term4 then accept
```

Now check that you have the hierarchy correctly configured.

```
[edit policy-options]
user@host# show
policy-statement statics {
  term term1 {
    from {
      route-filter 192.168.0.0/16 orlonger;
      route-filter 224.0.0.0/3 orlonger;
    }
    then reject;
  }
  term term2 {
    from protocol direct;
    then reject;
  }
  term term3 {
    from protocol static;
    then reject;
  }
  term term4 {
    then accept;
  }
}
```

Overview

When configuring a device running Junos OS, you can enter most statements and identifiers in any order. However, there are a few cases, such as in routing policies or firewall filters, in which the order of the statements matters because the configuration statements create a sequence that is analyzed in order.

To modify a portion of the configuration in which the statement order matters, you must use the **insert** configuration mode command. If you use the **set** command instead, the added statement or identifier will be in the wrong place sequentially. The only other way to get the terms of the command in the correct order is to dismantle the configuration and start over.

Configuration

CLI Quick Configuration To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks and change any details necessary to match your network configuration, copy and paste the commands into the CLI at the [edit policy-options] hierarchy level, and then enter commit from configuration mode.

```
[edit]
user@host# rename policy-options policy-statement statics term term4 to term term6
[edit]
user@host# set policy-options policy-statement statics term term4 from protocol local
[edit]
user@host# set policy-options policy-statement statics term term4 then reject
[edit]
user@host# set policy-options policy-statement statics term term5 from protocol
aggregate
[edit]
user@host# set policy-options policy-statement statics term term5 then reject
[edit]
user@host# insert policy-options policy-statement statics term term4 after term term3
[edit]
user@host# insert policy-options policy-statement statics term term5 after term term4
```

Configuring to Insert Terms

Step-by-Step Procedure

1. Determine in what order the terms in your configuration need to go—the original terms and the new terms you plan to add.

In the original configuration, the policy is named **statics** and there are four terms. Each of the first three terms matches on a different match criteria and the resulting matches are rejected. The last term accepts all the rest of the traffic.

In this example, you need to add two terms that weed out additional types of traffic. Both of these terms need to go before the last term in the original configuration.

2. Rename original term4 to term6.

```
[edit]
user@host# rename policy-options policy-statement statics term term4 to term
term6
```

This step preserves the original last term, now renamed term6, as the last term.

3. Create a new term4.

```
[edit]
user@host# set policy-options policy-statement statics term term4 from protocol
local
user@host# set policy-options policy-statement statics term term4 then reject
```

A new term is added that matches traffic from local system addresses and rejects it.

4. Create new term5.

```
[edit]
user@host# set policy-options policy-statement statics term term5 from protocol
aggregate
user@host# set policy-options policy-statement statics term term5 then reject
```

A new term is added that matches traffic from aggregate routes and rejects it.

5. Insert term4 after term3.

```
[edit]
user@host# insert policy-options policy-statement statics term term4 after term
term3
```

6. Insert term5 after term4.

```
[edit]
user@host# insert policy-options policy-statement statics term term5 after term
term4
```

Results

```
[edit]
user@host# show policy-options policy-statement statics
term term1 {
  from {
    route-filter 192.168.0.0/16 orlonger;
    route-filter 224.0.0.0/3 orlonger;
  }
  then reject;
}
term term2 {
  from protocol direct;
  then reject;
}
term term3 {
  from protocol static;
  then accept;
}
term term4 {
  from protocol local;
  then reject;
}
term term5 {
  from protocol aggregate;
  then reject;
}
```

```
term term6 {
  then accept;
}
```

Related Documentation

- [Inserting a New Identifier in a Junos OS Configuration on page 81](#)
- [Adding Junos OS Configuration Statements and Identifiers on page 69](#)

Example: Using the Wildcard Command with the Range Option

- [Requirements on page 85](#)
- [Overview on page 85](#)
- [Configuration on page 86](#)
- [Verification on page 88](#)

Requirements

This example uses the following hardware and software components:

- M Series, MX Series, T Series or EX Series device
- Junos OS Release 12.1 or later running on the device

Overview

The **range** option with the **wildcard** command enables you to specify ranges in **activate**, **deactivate**, **delete**, **protect**, **set**, **show**, and **unprotect** commands. You can use ranges to specify a range of interfaces, logical units, VLANs, and other numbered elements. The **wildcard range** option expands the command you entered into multiple commands, each of which corresponds to one item in the range.

The **wildcard range** option enables you to configure multiple configuration statements using a single **set** command, instead of configuring each of them individually. For example, to configure 24 Gigabit Ethernet interfaces with different port numbers, you can use a single **wildcard range set** command instead of 24 individual **set interfaces** commands.

Similarly, to deactivate a group of 30 logical interfaces, you can use the **wildcard range deactivate** command instead of deactivating each logical interface individually.

You can use **wildcard range** with the **active**, **deactivate**, **delete**, **protect**, **set**, **show**, and **unprotect** configuration commands:

```
user@host# wildcard range ?
```

Possible completions:

activate	Remove the inactive tag from a statement
deactivate	Add the inactive tag to a statement
delete	Delete a data element
protect	Protect the statement
set	Set a parameter
show	Show a parameter
unprotect	Unprotect the statement

You can also specify all configuration hierarchy levels and their child configuration statements in the CLI by using **wildcard range** with the **set** option:

Possible completions:

```
> > access           Network access configuration
> > access-profile    Access profile for this instance
> > accounting-options Accounting data configuration
> > applications      Define applications by protocol characteristics
...
```

Configuration

The following examples show how to configure multiple configuration statements in a single step by using the **range** option with the **wildcard** configuration command:

- [Using the Range Option for Configuring a Series of Named Identifiers for a Configuration Statement on page 86](#)
- [Specifying Multiple Ranges in the Syntax on page 86](#)
- [Specifying a Range and Unique Numbers In the Syntax on page 87](#)
- [Excluding Some Values from a Range on page 87](#)
- [Specifying a Range with a Step Number on page 88](#)

Using the Range Option for Configuring a Series of Named Identifiers for a Configuration Statement

Step-by-Step Procedure

You can configure a series of identifiers for a configuration statement, by specifying a numerical range of values for the identifiers.

- To configure a series of the same type of interface with different port numbers (0 through 23), specify the range for the port numbers by using the following format:

```
[edit]
user@host# wildcard range set interfaces ge-0/0/[0-23] unit 0 family vpls
```

Results Expands to 24 different **set** commands to configure interfaces with port numbers ranging from 0 through 23:

```
[edit]
user@host# set interfaces ge-0/0/0 unit 0 family vpls
user@host# set interfaces ge-0/0/1 unit 0 family vpls
user@host# set interfaces ge-0/0/2 unit 0 family vpls
...
user@host# set interfaces ge-0/0/23 unit 0 family vpls
```

Specifying Multiple Ranges in the Syntax

Step-by-Step Procedure

You can have multiple ranges specified in a **wildcard range** command. Each range must be separated by a comma. You can also have overlapping ranges.

- To specify more than one range in the syntax, include the minimum and maximum values for each range, separated by a comma.

```
[edit]
```

```
user@host# wildcard range protect event-options policy p[1-3,5-7,6-9]
```

Results Expands to the following **set** commands:

```
[edit]
user@host# set protect event-options policy p1
user@host# set protect event-options policy p2
user@host# set protect event-options policy p3
user@host# set protect event-options policy p5
user@host# set protect event-options policy p6
user@host# set protect event-options policy p7
user@host# set protect event-options policy p8
user@host# set protect event-options policy p9
```

Specifying a Range and Unique Numbers In the Syntax

Step-by-Step Procedure You can also specify a combination of a range and unique numbers in the syntax of the **wildcard range** command.

- To specify a range and unique numbers, separate them with a comma.

```
[edit]
user@host# wildcard range protect event-options policy p[1-3,5,7,10]
```

Results Expands to the following **set** commands:

```
[edit]
user@host# set protect event-options policy p1
user@host# set protect event-options policy p2
user@host# set protect event-options policy p3
user@host# set protect event-options policy p5
user@host# set protect event-options policy p7
user@host# set protect event-options policy p10
```

Excluding Some Values from a Range

Step-by-Step Procedure You can exclude certain values from a range by marking the numbers or the range of numbers to be excluded by using an exclamation mark.

- To exclude certain values from a range, include the portion to be excluded with **!** in the syntax.

```
[edit]
user@host# wildcard range protect event-options policy p[1-5,!3-4]
```

Results Expands to the following **set** commands:

```
[edit]
user@host# set protect event-options policy p1
user@host# set protect event-options policy p2
user@host# set protect event-options policy p5
```

Specifying a Range with a Step Number

- Step-by-Step Procedure** You can provide a step number for a range to have a constant interval in the range.
- To provide a step, include the step value in the syntax preceded by a forward slash (/).
- ```
[edit]
user@host# wildcard range protect event-options policy p[1-10/2]
```

**Results** Expands to the following **set** commands:

```
[edit]
user@host# set protect event-options policy p1
user@host# set protect event-options policy p3
user@host# set protect event-options policy p5
user@host# set protect event-options policy p7
user@host# set protect event-options policy p9
```

### Verification

Confirm that the configuration is working properly.

- [Checking the Configuration on page 88](#)

### Checking the Configuration

---

**Purpose** Check the configuration created using the **wildcard range** option. The following sample shows output for the configuration described in “[Using the Range Option for Configuring a Series of Named Identifiers for a Configuration Statement](#)” on page 86.



**Action** user@host> show configuration interfaces

```

ge-0/0/0 {
 unit 0 {
 family vpls;
 }
}
ge-0/0/1 {
 unit 0 {
 family vpls;
 }
}
ge-0/0/2 {
 unit 0 {
 family vpls;
 }
}
ge-0/0/3 {
 unit 0 {
 family vpls;
 }
}
...
ge-0/0/23 {
 unit 0 {
 family vpls;
 }
}

```

**Meaning** The output indicates that 24 Gigabit Ethernet interfaces ranging from **ge-0/0/0** through **ge-0/0/23** are created.

**Related Documentation** • [Using Wildcard Characters in Interface Names on page 203](#)

## Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration

In a Junos configuration, you can deactivate statements and identifiers so that they do not take effect when you issue the **commit** command. Any deactivated statements and identifiers are marked with the **inactive** tag. They remain in the configuration, but are not activated when you issue a **commit** command.

To deactivate a statement or identifier, use the **deactivate** configuration mode command:

```
user@host# deactivate(statement | identifier)
```

To reactivate a statement or identifier, use the **activate** configuration mode command:

```
user@host# activate (statement | identifier)
```

In both commands, the **statement** and **identifier** you specify must be at the current hierarchy level. When you deactivate a statement, that specific statement is completely ignored and is not applied at all when you issue a **commit** command.

To disable a statement, use the **disable** configuration mode command:

In some portions of the configuration hierarchy, you can include a **disable** statement to disable functionality. One example is disabling an interface by including the **disable** statement at the **[edit interface *interface-name*]** hierarchy level. When you disable a functionality, it is activated when you issue a **commit** command but is treated as though it is down or administratively disabled.

**Related  
Documentation**

- [Example: Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration on page 90](#)
- [Adding Junos OS Configuration Statements and Identifiers on page 69](#)

---

## Example: Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration

---

This example shows a common use case in which the **deactivate** and **activate** configuration mode commands are used. It involves dual Routing Engines, master and backup, that have graceful Routing Engine switchover (GRES) configured. The software on both Routing Engines needs to be upgraded. This can easily be accomplished by deactivating GRES, updating the Routing Engines, and then reactivating GRES.



**NOTE:** You can also perform a similar upgrade using the same setup except that nonstop active routing (NSR) is configured instead of GRES. You would need to deactivate NSR and then upgrade the Routing Engines before reactivating NSR.

- 
- [Requirements on page 90](#)
  - [Overview on page 90](#)
  - [Configuration on page 91](#)

### Requirements

This example requires the use of a router with dual Routing Engines that can be upgraded.

Before you begin this example, make sure that your have GRES configured.

### Overview

In this example, there are two Routing Engines. GRES is configured, and the Routing Engines need to be upgraded. To accomplish the upgrading, you need to deactivate the GRES feature, upgrade each of the Routing Engines, and then activate GRES again.

## Configuration

### Configuring the Deactivation and Reactivation of GRES

#### Step-by-Step Procedure

To deactivate and reactivate GRES for Routing Engine upgrade:

1. Show that GRES is enabled for the router.  

```
[edit]
user@host# show chassis
redundancy {
 graceful-switchover;
}
fpc 2 {
 pic 0 {
 tunnel-services {
 bandwidth 1g;
 }
 }
}
```
2. Deactivate GRES.  

```
[edit]
user@host# deactivate chassis redundancy graceful-switchover
user@host# commit
```
3. Show that GRES is deactivated.  

```
[edit]
user@host# show chassis
redundancy {
 inactive: graceful-switchover;
}
fpc 2 {
 pic 0 {
 tunnel-services {
 bandwidth 1g;
 }
 }
}
```
4. Upgrade the Routing Engines one by one.  

For instructions on upgrading Junos OS on dual Routing Engines, see tasks 2 and 3 in *Installing the Software Package on a Router with Redundant Routing Engines*.
5. Reactivate GRES.  

```
[edit]
user@host# activate chassis redundancy graceful-switchover
user@host# commit
```

**Results** Verify that GRES feature is activated again.

```
[edit]
user@host# show chassis
redundancy {
 graceful-switchover;
```

```
}
fpc 2 {
 pic 0 {
 tunnel-services {
 bandwidth 1g;
 }
 }
}
```

**Related  
Documentation**

- [Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration on page 89](#)

---

## Adding Comments in a Junos OS Configuration

You can include comments in a Junos configuration to describe any statement in the configuration. You can add comments interactively in the CLI and by editing the ASCII configuration file.

When configuring interfaces, you can add comments about the interface by including the **description** statement at the **[edit interfaces *interface-name*]** hierarchy level. Any comments you include appear in the output of the **show interfaces** commands. For more information about the **description** statement, see the *Junos OS Network Interfaces Library for Routing Devices*.

- [Adding Comments in the CLI on page 92](#)
- [Adding Comments in a File on page 93](#)

### Adding Comments in the CLI

When you add comments in configuration mode, they are associated with a statement at the current level. Each statement can have one single-line comment associated with it. Before you can associate a comment with a statement, the statement must exist. The comment is placed on the line preceding the statement.

To add comments to a configuration, use the **annotate** configuration mode command:

```
user@host# annotate statement "comment-string"
```

***statement*** is the configuration statement to which you are attaching the comment; it must be at the current hierarchy level. If a comment for the specified ***statement*** already exists, it is deleted and replaced with the new comment.

***comment-string*** is the text of the comment. The comment text can be any length, and you must type it on a single line. If the comment contains spaces, you must enclose it in quotation marks. In the comment string, you can include the comment delimiters ***/\* \*/*** or ***#***. If you do not specify any, the comment string is enclosed with the ***/\* \*/*** comment delimiters.

To delete an existing comment, specify an empty comment string:

```
user@host# annotate statement ""
```

If you add comments with the **annotate** command, you can view the comments within the configuration by entering the **show** configuration mode command or the **show configuration** operational mode command.



**NOTE:** The Junos OS supports annotation up to the last level in the configuration hierarchy, including oneliners. However, annotation of parts (the child statements or identifiers within the oneliner) of the oneliner is not supported. For example, in the following sample configuration hierarchy, annotation is supported up to the level 1 parent hierarchy, but not supported for the metric child statement:

```
[edit protocols]
 isis {
 interface ge-0/0/0.0 {
 level 1 metric 10;
 }
 }
}
```

## Adding Comments in a File

When you edit the ASCII configuration file and add comments, they can be one or more lines and must precede the statement they are associated with. If you place the comments in other places in the file, such as on the same line following a statement or on a separate line following a statement, they are removed when you use the **load** command to open the configuration into the CLI.

The following excerpt from a configuration example illustrates how to place and how not to place comments in a configuration file:

```
/* This comment goes with routing-options */
routing-options {
 /* This comment goes with routing-options traceoptions */
 traceoptions {
 /* This comment goes with routing-options traceoptions tracefile */
 tracefile rpd size 1m files 10;
 /* This comment goes with routing-options traceoptions traceflag task */
 traceflag task;
 /* This comment goes with routing-options traceoptions traceflag general */
 traceflag general;
 }
 autonomous-system 10458; /* This comment is dropped */
}
routing-options {
 rib-groups {
 ifrg {
 import-rib [inet.0 inet.2];
 /* A comment here is dropped */
 }
 dvmrp-rib {
 import-rib inet.2;
 export-rib inet.2;
 }
 }
}
```

```
 /* A comment here is dropped */
 }
 /* A comment here is dropped */
}
/* A comment here is dropped */
}
```

When you include comments in the configuration file directly, you can format comments in the following ways:

- Start the comment with a `/*` and end it with a `*/`. The comment text can be on a single line or can span multiple lines.
- Start the comment with a `#` and end it with a new line (carriage return).

**Related  
Documentation**

- [Adding Junos OS Configuration Statements and Identifiers on page 69](#)
- [Example: Including Comments in a Junos OS Configuration by Using the CLI on page 94](#)

---

## Example: Including Comments in a Junos OS Configuration by Using the CLI

---

Adding comments to a Junos OS configuration makes the configuration file readable and more readily understood by users. Using the Junos OS CLI, you can include comments as you configure by using the **annotate** statement. In this example, comments are added by using the CLI for an already existing configuration:

- [Requirements on page 94](#)
- [Overview on page 95](#)
- [Configuration on page 95](#)

### Requirements

No special configuration beyond device initialization is required before configuring this example.

Before you add a comment, you must configure the following hierarchy on the router.

To quickly configure the *initial configuration* for this example, copy the following command, paste it into a text file, remove any line breaks and change any details necessary to match your network configuration, copy and paste this command into the CLI at the **[edit]** hierarchy level, and then enter **commit** from configuration mode.

```
set protocols ospf area 0.0.0.0 interface so-0/0/0.0 hello-interval 5
```

Now, check that you have this hierarchy configured.

```
user@host# show protocols
ospf {
 area 0.0.0.0 {
 interface so-0/0/0 {
 hello-interval 5;
 }
 }
}
```

```
}
```

## Overview

When you add comments by using the CLI, you do so in configuration mode using the **annotate** statement. Each comment you add is associated with a statement at the current level. Each statement can have one single-line comment associated with it.

To configure the **annotate** statement, move to the level of the statement with which you want to associate a comment. To view the comments, go to the top of the configuration hierarchy and use the **show** command.

## Configuration

### CLI Quick Configuration

To quickly configure the *comments* for this example, copy the following commands, paste them into a text file, remove any line breaks and change any details necessary to match your network configuration, copy and paste the commands into the CLI, starting at the **[edit]** hierarchy level, and then enter **commit** from configuration mode.

```
edit protocols ospf
annotate area 0.0.0.0 "Backbone area configuration added June 15, 1998"
edit area 0.0.0.0
annotate interface so-0/0/0.0 "Interface from router sj1 to router sj2"
```

Notice that the commands are moving you down the hierarchy as you annotate different sections of the hierarchy.

### Including Comments in the CLI Configuration Mode

### Step-by-Step Procedure

This procedure assumes that you have already configured the initial configuration.

To add comments to a configuration:

1. Move to the first hierarchy level to which you need to add a comment.  

```
[edit]
user@host# edit protocols ospf
```
2. Add a comment to the **area** configuration statement by using the **annotate** statement.  

```
[edit protocols ospf]
user@host# annotate area 0.0.0.0 "Backbone area configuration added June 15, 1998"
```
3. Move down a level to the **interface** configuration statement.  

```
[edit protocols ospf]
user@host# edit area 0.0.0.0
```
4. Add a comment to interface so-0/0/0.0 by using the **annotate** statement.  

```
[edit protocols ospf area 0.0.0.0]
user@host# annotate interface so-0/0/0.0 "Interface from router sj1 to router sj2"
```

## Results

---

Move to the top of the hierarchy and use the **show** command to see the comments you added. The comments precede the statement they are associated with.

```
[edit]
user@host# show protocols
ospf {
 /* Backbone area configuration added June 15, 1998 */
 area 0.0.0.0 {
 /* Interface from router sj1 to router sj2 */
 interface so-0/0/0.0 {
 hello-interval 5;
 }
 }
}
```

After you have confirmed that the configuration is correct, enter the **commit** command.

**Related Documentation**

- [Adding Comments in a Junos OS Configuration on page 92](#)

## Updating the configure private Configuration

---

When you are in configure private mode, you must work with a copy of the most recently committed shared configuration. If the global configuration changes, you can issue the **update** command to update your private candidate configuration. When you do this, your private candidate configuration contains a copy of the most recently committed configuration with your private changes merged in. For example:

```
[edit]
user@host# update
[edit]
user@host#
```



**NOTE:** Merge conflicts can occur when you issue the **update** command.

You can also issue the **rollback** command to discard your private candidate configuration changes and obtain the most recently committed configuration:

```
[edit]
user@host# rollback
[edit]
user@host#
```

**Related Documentation**

- [Forms of the configure Command on page 65](#)



## Displaying the Current Junos OS Configuration

To display the current configuration for a device running Junos OS, use the **show** configuration mode command. This command displays the configuration at the current hierarchy level or at the specified level.

```
user@host# show <statement-path>
```

The configuration statements appear in a fixed order, interfaces appear alphabetically by type, and then in numerical order by slot number, PIC number, and port number. Note that when you configure the router, you can enter statements in any order.

You also can use the CLI operational mode **show configuration** command to display the last committed current configuration, which is the configuration currently running on the router:

```
user@host> show configuration
```

When you show a configuration, a timestamp at the top of the configuration indicates when the configuration was last changed:

```
Last commit: 2006-07-18 11:21:58 PDT by echen
version 8.3
```

If you have omitted a required statement at a particular hierarchy level, when you issue the **show** command in configuration mode, a message indicates which statement is missing. As long as a mandatory statement is missing, the CLI continues to display this message each time you issue a **show** command. For example:

```
[edit]
user@host# show
protocols {
 pim {
 interface so-0/0/0 {
 priority 4;
 version 2;
 # Warning: missing mandatory statement(s): 'mode'
 }
 }
}
```

When you issue the **show configuration** command with the **| display set** pipe option to view the configuration as **set** commands, those portions of the configuration that you do not have permissions to view are substituted with the text **ACCESS-DENIED**.

Unsupported statements included in the CLI configuration are displayed with the “unsupported” text in the configuration. For example, if a statement is configured on an unsupported platform, the CLI displays a message that the statement is ignored in the configuration because it is configured on an unsupported platform. When you issue the **show** command with the **| display xml** option, you can see the **unsupported="unsupported"** attribute for configuration that is unsupported.

The “unsupported” attribute included in text configuration or XML configuration is provided to scripts when the **unsupported="unsupported"** attribute is included in the **<get-configuration>** RPC call.

- Related Documentation**
- [Example: Displaying the Current Junos OS Configuration on page 98](#)
  - [Displaying set Commands from the Junos OS Configuration on page 101](#)

---

## Example: Displaying the Current Junos OS Configuration

The following example shows how you can display the current Junos configuration.

Set a configuration:

```
[edit]
user@host# set protocols ospf area 0.0.0.0 interface so-0/0/0 hello-interval 5
```

To display the current configuration:

```
[edit]
user@host# show
protocols {
 ospf {
 area 0.0.0.0 {
 interface so-0/0/0 {
 hello-interval 5;
 }
 }
 }
}
```

Display a particular hierarchy in the configuration:

```
[edit]
user@host# show protocols ospf area 0.0.0.0
interface so-0/0/0 {
 hello-interval 5;
}
```

Move down a level and display the configuration at that level:

```
[edit]
user@host# edit protocols ospf area 0.0.0.0
[edit protocols ospf area 0.0.0.0]
user@host# show
interface so-0/0/0 {
 hello-interval 5;
}
```

Set and commit a configuration:

```
[edit]
user@host# set protocols ospf area 0.0.0.0 interface so-0/0/0 hello-interval 5
[edit]
user@host# commit
commit complete
```

```
[edit]
user@host# quit
exiting configuration mode
```

Display the last committed configuration:

```
user@host> show configuration
Last commit: 2006-08-10 11:21:58 PDT by user
version 8.3
protocols {
 ospf {
 area 0.0.0.0 {
 interface so-0/0/0 {
 hello-interval 5;
 }
 }
 }
}
```

**Related Documentation** • [Displaying the Current Junos OS Configuration on page 97](#)

## Displaying Additional Information About the Junos OS Configuration

In configuration mode only, to display additional information about the configuration, use the **display detail** command after the pipe ( | ) in conjunction with a **show** command. The additional information includes the help string that explains each configuration statement and the permission bits required to add and modify the configuration statement.

```
user@host# show <hierarchy-level> | display detail
```

For example:

```
[edit]
user@host# show | display detail
##
version: Software version information
require: system
##
version "3.4R1 [tlim]";
system {
 ##
 ## host-name: Host name for this router
 ## match: ^[:alnum:]._-]+$
 ## require: system
 ##
}
host-name router-name;
##
domain-name: Domain name for this router
match: ^[:alnum:]._-]+$
require: system
##
domain-name isp.net;
##
```

```
backup-router: Address of router to use while booting
##
backup-router 192.168.100.1;
root-authentication {
 ##
 ## encrypted-password: Encrypted password string
 ##
 encrypted-password "$ABC123"; # SECRET-DATA
}
##
name-server: DNS name servers
require: system
##
name-server {
 ##
 ## name-server: DNS name server address
 ##
 208.197.1.0;
}
login {
 ##
 ## class: User name (login)
 ## match: ^[:alnum:._-]+$
 ##
 class super-user {
 ##
 ## permissions: Set of permitted operation categories
 ##
 permissions all;
 }
 ...
 ##
 ## services: System services
 ## require: system
 ##
 services {
 ## services: Service name
 ##
 ftp;
 ##
 ## services: Service name
 ##
 telnet;
 ##
 }
 syslog {
 ##
 ## file-name: File to record logging data
 ##
 file messages {
 ##
 ## Facility type
 ## Level name
 ##
 any notice;
 ##
 }
 }
}
```

```

 ## Facility type
 ## Level name
 ##
 authorization info;
 }
}
chassis {
 alarm {
 sonet {
 ##
 ## lol: Loss of light
 ## alias: loss-of-light
 ##
 lol red;
 }
 }
}
interfaces {
 ##
 ## Interface name
 ##
 at-2/1/1 {
 atm-options {
 ##
 ## vpi: Virtual path index
 ## range: 0 .. 255
 ## maximum-vcs: Maximum number of virtual circuits on this VP
 ##
 vpi 0 maximum-vcs 512;
 }
 ##
 ## unit: Logical unit number
 ## range: 0 .. 16384
 ##
 unit 0 {
 ##
 ## vci: ATM point-to-point virtual circuit identifier ([vpi.]vci)
 ##
 vci 0.128;
 }
 }
}
...

```

**Related Documentation**

- [Displaying set Commands from the Junos OS Configuration on page 101](#)

## Displaying set Commands from the Junos OS Configuration

In configuration mode, you can display the configuration as a series of configuration mode commands required to re-create the configuration. This is useful if you are not familiar

with how to use configuration mode commands or if you want to cut, paste, and edit the displayed configuration.

To display the configuration as a series of configuration mode commands, which are required to re-create the configuration from the top level of the hierarchy as **set** commands, issue the **show** configuration mode command with the **display set** option:

```
user@host# show | display set
```

This topic contains the following examples:

- [Example: Displaying set Commands from the Configuration on page 102](#)
- [Example: Displaying Required set Commands at the Current Hierarchy Level on page 102](#)
- [Example: Displaying set Commands with the match Option on page 103](#)

### Example: Displaying set Commands from the Configuration

Display the **set** commands from the configuration at the **[edit interfaces]** hierarchy level:

```
[edit interfaces fe-0/0/0]
user@host# show
unit 0 {
 family inet {
 address 192.107.1.230/24;
 }
 family iso;
 family mpls;
}
inactive: unit 1 {
 family inet {
 address 10.0.0.1/8;
 }
}
user@host# show | display set
set interfaces fe-0/0/0 unit 0 family inet address 192.107.1.230/24
set interfaces fe-0/0/0 unit 0 family iso
set interfaces fe-0/0/0 unit 0 family mpls
set interfaces fe-0/0/0 unit 1 family inet address 10.0.0.1/8
deactivate interfaces fe-0/0/0 unit 1
```

To display the configuration as a series of configuration mode commands required to re-create the configuration from the current hierarchy level, issue the **show** configuration mode command with the **display set relative** option:

```
user@host# show | display set relative
```

### Example: Displaying Required set Commands at the Current Hierarchy Level

Display the configuration as a series of configuration mode commands required to re-create the configuration from the current hierarchy level:

```
[edit interfaces fe-0/0/0]
user@host# show
unit 0 {
 family inet {
```

```

 address 192.107.1.230/24;
 }
 family iso;
 family mpls;
}
inactive: unit 1 {
 family inet {
 address 10.0.0.1/8;
 }
}
user@host# show | display set relative
set unit 0 family inet address 192.107.1.230/24
set unit 0 family iso
set unit 0 family mpls
set unit 1 family inet address 10.0.0.1/8
deactivate unit 1

```

To display the configuration as **set** commands and search for text matching a regular expression by filtering output, specify the **match** option after the pipe ( | ):

```
user@host# show | display set | match regular-expression
```

### Example: Displaying set Commands with the match Option

Display IP addresses associated with an interface:

```

xe-2/3/0 {
 unit 0 {
 family inet {
 address 192.107.9.106/30;
 }
 }
}
so-5/1/0 {
 unit 0 {
 family inet {
 address 192.107.9.15/32 {
 destination 192.107.9.192;
 }
 }
 }
}
lo0 {
 unit 0 {
 family inet {
 address 127.0.0.1/32;
 }
 }
}
user@host# show interfaces | display set | match address
set interfaces xe-2/3/0 unit 0 family inet address 192.168.9.106/30
set interfaces so-5/1/0 unit 0 family inet address 192.168.9.15/32 destination 192.168.9.192
set interfaces lo0 unit 0 family inet address 127.0.0.1/32

```

**Related Documentation**

- [Displaying the Current Junos OS Configuration on page 97](#)

## Displaying Users Currently Editing the Junos OS Configuration

---

To display the users currently editing the configuration, use the **status** configuration mode command:

```
user@host# status
Users currently editing the configuration:
rchen terminal p0 (pid 55691) on since 2006-03-01 13:17:25 PST
[edit interfaces]
```

The system displays who is editing the configuration (**rchen**), where the user is logged in (**terminal p0**), the date and time the user logged in (**2006-03-01 13:17:25 PST**), and what level of the hierarchy the user is editing (**[edit interfaces]**).

If you issue the **status** configuration mode command and a user has scheduled a candidate configuration to become active for a future time, the system displays who scheduled the commit (**root**), where the user is logged in (**terminal d0**), the date and time the user logged in (**2002-10-31 14:55:15 PST**), and that a commit is pending (**commit at**).

```
[edit]
user@host# status
Users currently editing the configuration:
root terminal d0 (pid 767) on since 2002-10-31 14:55:15 PST, idle 00:03:09
commit at
```

For information about how to schedule a commit, see [“Scheduling a Junos OS Commit Operation” on page 113](#).

If you issue the **status** configuration mode command and a user is editing the configuration in configure exclusive mode, the system displays who is editing the configuration (**root**), where the user is logged in (**terminal d0**), the date and time the user logged in (**2002-11-01 13:05:11 PST**), and that a user is editing the configuration in configure exclusive mode (**exclusive [edit]**).

```
[edit]
user@host# status
Users currently editing the configuration:
root terminal d0 (pid 2088) on since 2002-11-01 13:05:11 PST
exclusive [edit]
```

- Related Documentation**
- [Forms of the configure Command on page 65](#)
  - [Using the configure exclusive Command on page 67](#)

## Verifying a Junos OS Configuration

---

To verify that the syntax of a Junos configuration is correct, use the configuration mode **commit check** command:

```
[edit]
user@host# commit check
configuration check succeeds
[edit]
```



user@host#

If the **commit check** command finds an error, a message indicates the location of the error.

- Related Documentation**
- [Adding Junos OS Configuration Statements and Identifiers on page 69](#)
  - [Committing a Junos OS Configuration on page 108](#)



## CHAPTER 5

# Committing a Junos OS Configuration

- [Junos OS Commit Model for Router or Switch Configuration on page 107](#)
- [Committing a Junos OS Configuration on page 108](#)
- [Committing a Junos OS Configuration and Exiting Configuration Mode on page 110](#)
- [Commit Operation When Multiple Users Configure the Software on page 111](#)
- [Activating a Junos OS Configuration but Requiring Confirmation on page 112](#)
- [Scheduling a Junos OS Commit Operation on page 113](#)
- [Monitoring the Junos OS Commit Process on page 114](#)
- [Adding a Comment to Describe the Committed Configuration on page 115](#)
- [Backing Up the Committed Configuration on the Alternate Boot Drive on page 116](#)
- [Junos OS Batch Commits Overview on page 116](#)
- [Example: Configuring Batch Commit Server Properties on page 117](#)

## Junos OS Commit Model for Router or Switch Configuration

---

The router or switch configuration is saved using a commit model—a candidate configuration is modified as desired and then committed to the system. When a configuration is committed, the router or switch checks the configuration for syntax errors, and if no errors are found, the configuration is saved as **juniper.conf.gz** and activated. The formerly active configuration file is saved as the first rollback configuration file (**juniper.conf.1.gz**), and any other rollback configuration files are incremented by 1. For example, **juniper.conf.1.gz** is incremented to **juniper.conf.2.gz**, making it the second rollback configuration file. The router or switch can have a maximum of 49 rollback configurations (numbered 1 through 49) saved on the system.

On the router or switch, the active configuration file and the first three rollback files (**juniper.conf.gz.1**, **juniper.conf.gz.2**, **juniper.conf.gz.3**) are located in the **/config** directory. If the file **rescue.conf.gz** is saved on the system, this file should also be saved in the **/config** directory. The factory default files are located in the **/etc/config** directory.

There are two mechanisms used to propagate the configurations between Routing Engines within a router or switch:

- **Synchronization**—Propagates a configuration from one Routing Engine to a second Routing Engine within the same router or switch chassis.



**NOTE:** The QFX3500 switch has only one Routing Engine.

To synchronize configurations, use the **commit synchronize** CLI command. If one of the Routing Engines is locked, the synchronization fails. If synchronization fails because of a locked configuration file, you can use the **commit synchronize force** command. This command overrides the lock and synchronizes the configuration files.

- **Distribution**—Propagates a configuration across the routing plane on a multichassis router or switch. Distribution occurs automatically. There is no user command available to control the distribution process. If a configuration is locked during a distribution of a configuration, the locked configuration does not receive the distributed configuration file, so the synchronization fails. You need to clear the lock before the configuration and resynchronize the routing planes.



**NOTE:** When you use the **commit synchronize force** CLI command on a multichassis platform, the forced synchronization of the configuration files does not affect the distribution of the configuration file across the routing plane. If a configuration file is locked on a router or switch remote from the router or switch where the command was issued, the synchronization fails on the remote router or switch. You need to clear the lock and reissue the **synchronize** command.

**Related  
Documentation**

- *Configuring Junos OS for the First Time on a Router or Switch with a Single Routing Engine*
- [commit on page 264](#)

---

## Committing a Junos OS Configuration

---

To save Junos OS configuration changes to the configuration database and to activate the configuration on the router, use the **commit** configuration mode command. You can issue the **commit** command from any hierarchy level:

```
[edit]
user@host# commit
commit complete
[edit]
user@host#
```

When you enter the **commit** command, the configuration is first checked for syntax errors (**commit check**). Then, if the syntax is correct, the configuration is activated and becomes the current, operational router configuration.

You can issue the **commit** command from any hierarchy level.

A configuration commit can fail for any of the following reasons:

- The configuration includes incorrect syntax, which causes the commit check to fail.

- The candidate configuration that you are trying to commit is larger than 700 MB.
- The configuration is locked by a user who entered the **configure exclusive** command.

If the configuration contains syntax errors, a message indicates the location of the error, and the configuration is not activated. The error message has the following format:

```
[edit edit-path]
'offending-statement;
error-message
```

For example:

```
[edit firewall filter login-allowed term allowed from]
'icmp-type [echo-request echo-reply]';
keyword 'echo-reply' unrecognized
```

You must correct the error before recommitting the configuration. To return quickly to the hierarchy level where the error is located, copy the path from the first line of the error and paste it at the configuration mode prompt at the **[edit]** hierarchy level.

The uncommitted, candidate configuration file is **/var/run/db/juniper.db**. It is limited to 700 MB. If the commit fails with a message **configuration database size limit exceeded**, view the file size from configuration mode by entering the command **run file list /var/run/db detail**. You can simplify the configuration and reduce the file size by creating configuration groups with wildcards or defining less specific match policies in your firewall filters.



**NOTE:** CLI commit-time warnings displayed for configuration changes at the **[edit interfaces]** hierarchy level are removed and are logged as system log messages.

This is also applicable to VRRP configuration at the following hierarchy levels:

- **[edit interfaces *interface-name* unit *logical-unit-number* family (*inet* | *inet6*) address *address*]**
- **[edit logical-systems *logical-system-name* interfaces *interface-name* unit *logical-unit-number* family (*inet* | *inet6*) address *address*]**

When you commit a configuration, you commit the entire configuration in its current form. If more than one user is modifying the configuration, committing it saves and activates the changes of all the users.



---

**NOTE:**

- If you are using Junos OS in a Common Criteria environment, system log messages are created whenever a secret attribute is changed (for example, password changes or changes to the RADIUS shared secret). These changes are logged during the following configuration load operations:

- load merge
  - load replace
  - load override
  - load update

For more information, see the *Secure Configuration Guide for Common Criteria and Junos-FIPS*.

- We do not recommend performing a commit operation on the backup Routing Engine when graceful Routing Engine switchover is enabled on the router.



---

**NOTE:** If you configure the same IP address for a management interface or internal interface such as `fxp0` and an external physical interface such as `ge-0/0/1`, when graceful Routing Engine switchover (GRES) is enabled, the CLI displays an appropriate commit error message that identical addresses have been found on the private and public interfaces. In such cases, you must assign unique IP addresses for the two interfaces that have duplicate addresses.

The management Ethernet interface used for the TX Matrix Plus router, T1600 or T4000 routers in a routing matrix, and PTX Series Packet Transport Routers, is `em0`. Junos OS automatically creates the router's management Ethernet interface, `em0`.

---

**Related Documentation**

- [Committing a Junos OS Configuration and Exiting Configuration Mode on page 110](#)
- [Activating a Junos OS Configuration but Requiring Confirmation on page 112](#)
- [Backing Up the Committed Configuration on the Alternate Boot Drive on page 116](#)
- [Forms of the configure Command on page 65](#)

---

## Committing a Junos OS Configuration and Exiting Configuration Mode

---

To save Junos OS configuration changes, activate the configuration on the device and exit configuration mode, using the **commit and-quit** configuration mode command. This command succeeds only if the configuration contains no errors.

```
[edit]
user@host# commit and-quit
commit complete
```

```

exiting configuration mode
user@host>

```



**NOTE:** We do not recommend performing a commit operation on the backup Routing Engine when graceful Routing Engine switchover is enabled on the router.

**Related  
Documentation**

- [Activating a Junos OS Configuration but Requiring Confirmation on page 112](#)

## Commit Operation When Multiple Users Configure the Software

Up to 32 users can be in configuration mode simultaneously, and they all can be making changes to the configuration. All changes made by all users are visible to everyone editing the configuration—the changes become visible as soon as the user presses the Enter key at the end of a command that changes the configuration, such as **set**, **edit**, or **delete**.

When any of the users editing the configuration issues a **commit** command, all changes made by all users are checked and activated.

If you enter configuration mode with the **configure private** command, each user has a private candidate configuration to edit somewhat independently of other users. When you commit the configuration, only your own changes get committed. To synchronize your copy of the configuration after other users have committed changes, you can run the **update** command in configuration mode. A commit operation also updates all of the private candidate configurations. For example, suppose user X and user Y are both in **configure private** mode, and user X commits a configuration change. When user Y performs a subsequent commit operation and then views the new configuration, the new configuration seen by user Y includes the changes made by user X.

If you enter configuration mode with the **configure exclusive** command, you lock the candidate configuration for as long as you remain in configuration mode, allowing you to make changes without interference from other users. Other users can enter and exit configuration mode, but they cannot commit the configuration. This is true even if the other users entered configuration mode before you enter the **configure exclusive** command. For example, suppose user X is already in the **configure private** or **configure** mode. Then suppose user Y enters the **configure exclusive** mode. User X cannot commit any changes to the configuration, even if those changes were entered before user Y logged in. If user Y exits **configure exclusive** mode, user X can then commit the changes made in **configure private** or **configure** mode.

**Related  
Documentation**

- [Committing a Junos OS Configuration on page 108](#)
- [Forms of the configure Command on page 65](#)
- [Displaying Users Currently Editing the Junos OS Configuration on page 104](#)

## Activating a Junos OS Configuration but Requiring Confirmation

---

When you commit the current candidate configuration, you can require an explicit confirmation for the commit to become permanent. This is useful if you want to verify that a configuration change works correctly and does not prevent access to the router. If the change prevents access or causes other errors, the router automatically returns to the previous configuration and restores access after the rollback confirmation timeout passes. This feature is called automatic rollback.

To commit the current candidate configuration but require an explicit confirmation for the commit to become permanent, use the **commit confirmed** configuration mode command:

```
[edit]
user@host# commit confirmed
commit confirmed will be automatically rolled back in 10 minutes unless confirmed
commit complete
#commit confirmed will be rolled back in 10 minutes
[edit]
user@host#
```

Once you have verified that the change works correctly, you can keep the new configuration active by entering a **commit** or **commit check** command within 10 minutes of the **commit confirmed** command. For example:

```
[edit]
user@host# commit check
commit confirmed will be automatically rolled back in 10 minutes unless confirmed
commit complete
#commit confirmed will be rolled back in 10 minutes
[edit]
user@host#
```

If the commit is not confirmed within a certain time (10 minutes by default), Junos OS automatically rolls back to the previous configuration and a broadcast message is sent to all logged-in users.

To show when a rollback is scheduled after a **commit confirmed** command, enter the **show system commit** command. For example:

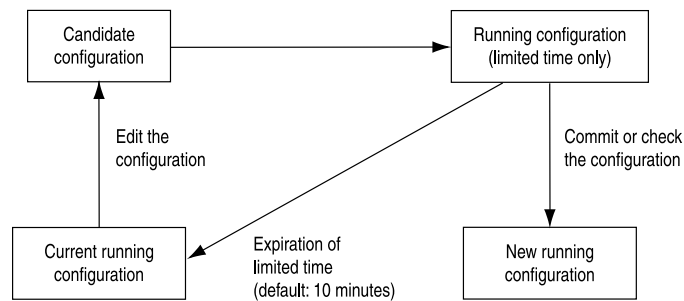
```
user@host> show system commit
0 2005-01-05 15:00:37 PST by root via cli commit confirmed, rollback in 3mins
```

Like the **commit** command, the **commit confirmed** command verifies the configuration syntax and reports any errors. If there are no errors, the configuration is activated and begins running on the router.

Figure 5 on page 113 illustrates how the **commit confirmed** command works.



Figure 5: Confirm a Configuration



To change the amount of time before you have to confirm the new configuration, specify the number of minutes when you issue the command:

```

[edit]
user@host# commit confirmed minutes
commit complete
[edit]
user@host#

```

In Junos OS Release 11.4 and later, you can also use the **commit confirmed** command in the **[edit private]** configuration mode.

#### Related Documentation

- [Scheduling a Junos OS Commit Operation on page 113](#)
- [Committing a Junos OS Configuration on page 108](#)

## Scheduling a Junos OS Commit Operation

You can schedule when you want your candidate configuration to become active. To save Junos OS configuration changes and activate the configuration on the router at a future time or upon reboot, use the **commit at** configuration mode command, specifying **reboot** or a future time at the **[edit]** hierarchy level:

```

[edit]
user@host # commit at string

```

Where **string** is **reboot** or the future time to activate the configuration changes. You can specify time in two formats:

- A time value in the form **hh:mm[:ss]** (hours, minutes, and optionally seconds)—Commit the configuration at the specified time, which must be in the future but before 11:59:59 PM on the day the **commit at** configuration mode command is issued. Use 24-hour time for the **hh** value; for example, **04:30:00** is 4:30:00 AM, and **20:00** is 8:00 PM. The time is interpreted with respect to the clock and time zone settings on the router.
- A date and time value in the form **yyyy-mm-dd hh:mm[:ss]** (year, month, date, hours, minutes, and, optionally, seconds)—Commit the configuration at the specified day and time, which must be after the **commit at** command is issued. Use 24-hour time for the **hh** value. For example, **2003-08-21 12:30:00** is 12:30 PM on August 21, 2003. The time is interpreted with respect to the clock and time zone settings on the router.

Enclose the **string** value in quotation marks (" "). For example, **commit at "18:00:00"**. For date and time, include both values in the same set of quotation marks. For example, **commit at "2005-03-10 14:00:00"**.

A commit check is performed immediately when you issue the **commit at** configuration mode command. If the result of the check is successful, then the current user is logged out of configuration mode, and the configuration data is left in a read-only state. No other commit can be performed until the scheduled commit is completed.



**NOTE:** If Junos OS fails before the configuration changes become active, all configuration changes are lost.

You cannot enter the **commit at** configuration command after you issue the **request system reboot** command.

You cannot enter the **request system reboot** command once you schedule a commit operation for a specific time in the future.

You cannot commit a configuration when a scheduled commit is pending. For information about how to cancel a scheduled configuration by means of the **clear** command, see the [CLI Explorer](#).



**NOTE:** We do not recommend performing a commit operation on the backup Routing Engine when graceful Routing Engine switchover is enabled on the router.

**Related  
Documentation**

- [Committing a Junos OS Configuration on page 108](#)
- [Monitoring the Junos OS Commit Process on page 114](#)

---

## Monitoring the Junos OS Commit Process

To monitor the Junos commit process, use the **display detail** command after the pipe with the **commit** command:

```
user@host# commit | display detail
```

For example:

```
[edit]
user@host# commit | display detail
2003-09-22 15:39:39 PDT: exporting juniper.conf
2003-09-22 15:39:39 PDT: setup foreign files
2003-09-22 15:39:39 PDT: propagating foreign files
2003-09-22 15:39:39 PDT: complete foreign files
2003-09-22 15:39:40 PDT: copying configuration to juniper.data+
2003-09-22 15:39:40 PDT: dropping unchanged foreign files
2003-09-22 15:39:40 PDT: daemons checking new configuration
2003-09-22 15:39:41 PDT: commit wrapup...
```

```

2003-09-22 15:39:42 PDT: activating '/var/etc/ntp.conf'
2003-09-22 15:39:42 PDT: activating '/var/etc/kmd.conf'
2003-09-22 15:39:42 PDT: activating '/var/db/juniper.data'
2003-09-22 15:39:42 PDT: notifying daemons of new configuration
2003-09-22 15:39:42 PDT: signaling 'Firewall daemon', pid 24567, signal 1,
status 0
2003-09-22 15:39:42 PDT: signaling 'Interface daemon', pid 24568, signal 1,
status 0
2003-09-22 15:39:43 PDT: signaling 'Routing protocol daemon', pid 25679,
signal 1, status 0
2003-09-22 15:39:43 PDT: signaling 'MIB2 daemon', pid 24549, signal 1,
status 0
2003-09-22 15:39:43 PDT: signaling 'NTP daemon', pid 37863, signal 1, status 0
2003-09-22 15:39:43 PDT: signaling 'Sonet APS daemon', pid 24551, signal 1,
status 0
2003-09-22 15:39:43 PDT: signaling 'VRRP daemon', pid 24552, signal 1,
status 0
2003-09-22 15:39:43 PDT: signaling 'PFE daemon', pid 2316, signal 1, status 0
2003-09-22 15:39:43 PDT: signaling 'Traffic sampling control daemon', pid 24553
signal 1, status 0
2003-09-22 15:39:43 PDT: signaling 'IPsec Key Management daemon', pid
24556, signal 1, status 0
2003-09-22 15:39:43 PDT: signaling 'Forwarding UDP daemon', pid 2320,
signal 1, status 0
commit complete

```

**Related  
Documentation**

- [Committing a Junos OS Configuration on page 108](#)
- [Adding a Comment to Describe the Committed Configuration on page 115](#)

## Adding a Comment to Describe the Committed Configuration

You can include a comment that describes changes to the committed configuration. To do so, include the `commit comment` statement. The comment can be as long as 512 bytes and you must type it on a single line.

```

[edit]
user@host# commit comment comment-string

```

*comment-string* is the text of the comment.



**NOTE:** You cannot include a comment with the `commit check` command.

To add a comment to the `commit` command, include the `comment` statement after the `commit` command:

```

[edit]
user@host# commit comment "add user joe"
commit complete
[edit]
user@host#

```

To add a comment to the **commit confirmed** command, include the **comment** statement after the **commit confirmed** command:

```
[edit]
user@host# commit confirmed comment "add customer to port 27"
commit confirmed will be automatically rolled back in 10 minutes unless confirmed
commit complete
[edit]
user@host#
```

To view these commit comments, issue the **show system commit** operational mode command.

In Junos OS Release 11.4 and later, you can also use the **commit confirmed** command in the **[edit private]** configuration mode.

- Related Documentation**
- [Committing a Junos OS Configuration on page 108](#)
  - [Backing Up the Committed Configuration on the Alternate Boot Drive on page 116](#)

---

## Backing Up the Committed Configuration on the Alternate Boot Drive

After you commit the configuration and are satisfied that it is running successfully, you should issue the **request system snapshot** command to back up the new software onto the **/altconfig** file system. If you do not issue the **request system snapshot** command, the configuration on the alternate boot drive will be out of sync with the configuration on the primary boot drive.

The **request system snapshot** command backs up the root file system to **/altroot**, and **/config** to **/altconfig**. The root and **/config** file systems are on the router's flash drive, and the **/altroot** and **/altconfig** file systems are on the router's hard disk (if available).



**NOTE:** For more information about backing up the file system on an ACX Series Universal Access Router, see *Understanding System Snapshot on an ACX Series Router*.

After you issue the **request system snapshot** command, you cannot return to the previous version of the software because the running and backup copies of the software are identical.

- Related Documentation**
- [Committing a Junos OS Configuration on page 108](#)

---

## Junos OS Batch Commits Overview

Junos OS provides a batch commit feature that aggregates or merges multiple configuration edits from different CLI sessions or users and adds them to a batch commit queue. A batch commit server running on the device takes one or more jobs from the batch commit queue, applies the configuration changes to the shared configuration database, and then commits the configuration changes in a single commit operation.

Batches are prioritized by the commit server based on priority of the batch specified by the user or the time when the batch job is added. When one batch commit is complete, the next set of configuration changes are aggregated and loaded into the batch queue for the next session of the batch commit operation. Batches are created until there are no commit entries left in the queue directory.

When compared to the regular commit operation where all commits are independently committed sequentially, batch commits save time and system resources by committing multiple small configuration edits in a single commit operation.

Batch commits are performed from the **[edit batch]** configuration mode. The commit server properties can be configured at the **[edit system commit server]** hierarchy level.

## Aggregation and Error Handling

When there is a load-time error in one of the aggregated jobs, the commit job that encounters the error is discarded and the remaining jobs are aggregated and committed.

For example, if there are five commit jobs (**commit-1**, **commit-2**, **commit-3**, **commit-4**, and **commit-5**) being aggregated, and **commit-3** encounters an error while loading, **commit-3** is discarded and **commit-1**, **commit-2**, **commit-4**, and **commit-5** are aggregated and committed.

If there is an error during the commit operation when two or more jobs are aggregated and committed, the aggregation is discarded and each of those jobs is committed individually like a regular commit operation.

For example, if there are five commit jobs (**commit-1**, **commit-2**, **commit-3**, **commit-4**, and **commit-5**) that are aggregated and if there is a commit error caused because of **commit-3**, the aggregation is discarded, **commit-1**, **commit-2**, **commit-3**, **commit-4**, and **commit-5** are committed individually, and the CLI reports a commit error for **commit-3**.

### Related Documentation

- [Example: Configuring Batch Commit Server Properties on page 117](#)

---

## Example: Configuring Batch Commit Server Properties

This example shows how to configure batch commit server properties to manage batch commit operations.

- [Requirements on page 117](#)
- [Overview on page 118](#)
- [Configuration on page 118](#)
- [Verification on page 120](#)

## Requirements

This example uses the following hardware and software components:

- MX Series 3D Universal Edge Router
- Junos OS Release 12.1 or later running on the device

## Overview

You can control how the batch commit queue is handled by the commit server by configuring the server properties at the **[edit system commit server]** hierarchy level. This enables you to control how many commit jobs are aggregated or merged into a single batch commit, the maximum number of jobs that can be added to the queue, days to keep batch commit error logs, interval between two batch commits, and tracing operations for batch commit operations.

## Configuration

**CLI Quick Configuration** To quickly configure this section of the example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level. You can configure the commit server properties from either the regular **[edit]** mode or the **[edit batch]** mode.

**Device R0**

```
set system commit server maximum-aggregate-pool 4
set system commit server maximum-entries 500
set system commit server commit-interval 5
set system commit server days-to-keep-error-logs 30
set system commit server traceoptions file commitd_nov
set system commit server traceoptions flag all
```

---

### Configuring the Commit Server Properties

---

- Step-by-Step Procedure**
1. (Optional) Configure the number of commit transactions to aggregate or merge in a single commit operation.

The default value for **maximum-aggregate-pool** is 5.



**NOTE:** Setting **maximum-aggregate-pool** to 1 commits each of the jobs individually.

In this example, the number of commit transactions is set to 4 indicating that four different commit jobs are aggregated into a single commit before the commit operation is initiated.

```
[edit system commit server]
user@R0# set maximum-aggregate-pool 4
```

2. (Optional) Configure the maximum number of jobs allowed in a batch.

This limits the number of commits jobs that are added to the queue.

```
[edit system commit server]
user@R0# set maximum-entries 500
```



**NOTE:** If you set **maximum-entries** to 1, the commit server cannot add more than one job to the queue, and the CLI displays an appropriate message when you try to commit more than one job.

3. (Optional) Configure the time (in seconds) to wait before starting the next batch commit operation.

```
[edit system commit server]
user@R0# set commit-interval 5
```

4. (Optional) Configure the number of days to keep error logs.

The default value is 30 days.

```
[edit system commit server]
user@R0# set days-to-keep-error-logs 30
```

5. (Optional) Configure tracing operations to log batch commit events.

In this example, the filename for logging batch commit events is **commitd\_nov**, and all traceoption flags are set.

```
[edit system commit server]
user@R0# set traceoptions commitd_nov
user@R0# set traceoptions flag all
```

**Results** From configuration mode, confirm your configuration by entering the **show system commit server** command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R0# show system commit server
maximum-aggregate-pool 4;
maximum-entries 500;
commit-interval 5;
days-to-keep-error-logs 30;
traceoptions {
 file commitd_nov;
 flag all;
}
```

### Committing the Configuration from Batch Configuration Mode

**Step-by-Step Procedure** To commit the configuration from the **[edit batch]** mode, do one of the following:

- Log in to the device and enter **commit**.

```
[edit batch]
user@R0# commit
Added to commit queue request-id: 1000
```

- To assign a higher priority to a batch commit job, issue the **commit** command with the **priority** option.

```
[edit batch]
user@R0# commit priority
Added to commit queue request-id: 1001
```

- To commit a configuration without aggregating the configuration changes with other commit jobs in the queue, issue the **commit** command with the **atomic** option.

```
[edit batch]
user@R0# commit atomic
Added to commit queue request-id: 1002
```

- To commit a configuration without aggregating the configuration changes with other commit jobs in the queue, and issuing a higher priority to the commit job, issue the **commit** command with the **atomic priority** option.

[edit batch]

user@R0# **commit atomic priority**

Added to commit queue request-id: 1003

## Verification

Confirm that the configuration is working properly.

- [Checking the Batch Commit Server Status on page 120](#)
- [Checking the Batch Commit Status on page 120](#)
- [Viewing the Patch Files in a Batch Commit Job on page 121](#)
- [Viewing the Trace Files for Batch Commit Operations on page 123](#)

---

### Checking the Batch Commit Server Status

**Purpose** Check the status of the batch commit server.

**Action** user@R0> **show system commit server**  
Commit server status : Not running

By default, the status of the commit server is **Not running**. The commit server starts running only when a batch commit job is added to the queue.

When a batch commit job is added to the queue, the status of the commit server changes to **Running**.

user@R0> **show system commit server**

Commit server status : Running  
Jobs in process:  
1003 1004 1005

**Meaning** The **Jobs in process** field lists the commit IDs of jobs that are in process.

---

### Checking the Batch Commit Status

**Purpose** Check the commit server queue for the status of the batch commits.



**Action** user@R0> show system commit server queue

```
Pending commits:
 Id: 1005
 Last Modified: Tue Nov 1 23:56:43 2011

Completed commits:
 Id: 1000
 Last Modified: Tue Nov 1 22:46:43 2011
 Status: Successfully committed 1000

 Id: 1002
 Last Modified: Tue Nov 1 22:50:35 2011
 Status: Successfully committed 1002

 Id: 1004
 Last Modified: Tue Nov 1 22:51:48 2011
 Status: Successfully committed 1004

 Id: 1007
 Last Modified: Wed Nov 2 01:08:04 2011
 Status: Successfully committed 1007

 Id: 1009
 Last Modified: Wed Nov 2 01:16:45 2011
 Status: Successfully committed 1009

 Id: 1010
 Last Modified: Wed Nov 2 01:19:25 2011
 Status: Successfully committed 1010

 Id: 1011
 Last Modified: Wed Nov 2 01:28:16 2011
 Status: Successfully committed 1011

Error commits:
 Id: 1008
 Last Modified: Wed Nov 2 01:08:18 2011
 Status: Error while committing 1008
```

**Meaning** **Pending commits** displays commit jobs that are added to the commit queue but are not committed yet. **Completed commits** displays the list of commit jobs that are successful. **Error commits** are commits that failed because of an error.

### Viewing the Patch Files in a Batch Commit Job

**Purpose** View the timestamps, patch files, and the status of each of the commit jobs. Patch files show the configuration changes that occur in each commit operation that is added to the batch commit queue.

**Action** 1. Issue the **show system commit server queue patch** command to view the patches for all commit operations.

```
user@R0> show system commit server queue patch
Pending commits:
 none
```

## Completed commits:

```
Id: 1000
Last Modified: Tue Nov 1 22:46:43 2011
Status: Successfully committed 1000
```

## Patch:

```
[edit groups]
 re1 { ... }
+ GRP-DHCP-POOL-NOACCESS {
+ access {
+ address-assignment {
+ pool <*> {
+ family inet {
+ dhcp-attributes {
+ maximum-lease-time 300;
+ grace-period 300;
+ domain-name verizon.net;
+ name-server {
+ 4.4.4.1;
+ 4.4.4.2;
+ }
+ }
+ }
+ }
+ }
+ }
+ }
```

```
Id: 1002
Last Modified: Tue Nov 1 22:50:35 2011
Status: Successfully committed 1002
```

## Patch:

```
[edit]
+ snmp {
+ community abc;
+ }
```

```
Id: 1010
Last Modified: Wed Nov 2 01:19:25 2011
Status: Successfully committed 1010
```

## Patch:

```
[edit system syslog]
 file test { ... }
+ file j {
+ any any;
+ }
```

## Error commits:

```
Id: 1008
Last Modified: Wed Nov 2 01:08:18 2011
Status: Error while committing 1008
```

## Patch:

```
[edit system]
+ radius-server {
+ 10.1.1.1 port 222;
+ }
```

The output shows the changes in configuration for each commit job ID.

- To view the patch for a specific commit job ID, issue the **show system commit server queue patch id <id-number>** command.

```

user@R0> show system commit server queue patch id 1000
Completed commits:
 Id: 1000
 Last Modified: Tue Nov 1 22:46:43 2011
 Status: Successfully committed 1000

Patch:
[edit system]
+ radius-server {
+ 192.168.69.162 secret teH.bTc/RVbPM;
+ 192.168.64.10 secret teH.bTc/RVbPM;
+ 192.168.60.52 secret teH.bTc/RVbPM;
+ 192.168.60.55 secret teH.bTc/RVbPM;
+ 192.168.4.240 secret teH.bTc/RVbPM;
+ }

```

**Meaning** The output shows the patch created for a commit job. The + or - sign indicates the changes in the configuration for a specific commit job.

### Viewing the Trace Files for Batch Commit Operations

**Purpose** View the trace files for batch commit operations. You can use the trace files for troubleshooting purposes.

- Action**
- Issue the **file show /var/log/<filename>** command to view all entries in the log file.

```

user@R0> file show /var/log/commitd_nov

```

The output shows commit server event logs and other logs for batch commits.

```

Nov 1 22:46:43 Successfully committed 1000
Nov 1 22:46:43 pausing after commit for 0 seconds
...
Nov 1 22:46:43 Done working on queue
...

Nov 1 22:47:17 maximum-aggregate-pool = 5
Nov 1 22:47:17 maximum-entries= 0
Nov 1 22:47:17 asynchronous-prompt = no
Nov 1 22:47:17 commit-interval = 0
Nov 1 22:47:17 days-to-keep-error-logs = -1
...
Nov 1 22:47:17 Added to commit queue request-id: 1001
Nov 1 22:47:17 Commit server status=running
Nov 1 22:47:17 No need to pause
...

Nov 1 22:47:18 Error while committing 1001
Nov 1 22:47:18 doing rollback
...

```

- To view log entries only for successful batch commit operations, issue the **file show /var/log/<filename>** command with the **| match committed** pipe option.

```

user@R0> file show /var/log/commitd_nov | match committed

```

The output shows batch commit job IDs for successful commit operations.

```
Nov 1 22:46:43 Successfully committed 1000
Nov 1 22:50:35 Successfully committed 1002
Nov 1 22:51:48 Successfully committed 1004
Nov 2 01:08:04 Successfully committed 1007
Nov 2 01:16:45 Successfully committed 1009
Nov 2 01:19:25 Successfully committed 1010
Nov 2 01:28:16 Successfully committed 1011
```

- To view log entries only for failed batch commit operations, issue the **file show** `/var/log/<filename>` command with the **| match "Error while"** pipe option.

```
user@R0> file show /var/log/commitd_nov | match "Error while"
```

The output shows commit job IDs for failed commit operations.

```
Nov 1 22:47:18 Error while committing 1001
Nov 1 22:51:10 Error while committing 1003
Nov 1 22:52:15 Error while committing 1005
...
```

- To view log entries only for commit server events, issue the **file show** `/var/log/<filename>` command with the **| match "commit server"** pipe option.

```
user@R0> file show /var/log/commitd_nov | match "commit server"
```

The output shows commit server event logs.

```
Nov 1 22:46:39 Commit server status=running
Nov 1 22:46:39 Commit server jobs=1000
Nov 1 22:46:43 Commit server status=not running
Nov 1 22:46:43 Commit server jobs=
Nov 1 22:47:17 Commit server status=running
Nov 1 22:47:18 Commit server jobs=1001
Nov 1 22:47:18 2 errors reported by commit server
Nov 1 22:47:18 Commit server status=not running
Nov 1 22:47:18 Commit server jobs=
Nov 1 22:50:31 Commit server status=running
Nov 1 22:50:31 Commit server jobs=1002
Nov 1 22:50:35 Commit server status=not running
Nov 1 22:50:35 Commit server jobs=
Nov 1 22:51:09 Commit server status=running
Nov 1 22:51:10 Commit server jobs=1003
Nov 1 22:51:10 2 errors reported by commit server
Nov 1 22:51:10 Commit server status=not running
...
```

#### Related Documentation

- [Junos OS Batch Commits Overview on page 116](#)
- [commit-interval \(Batch Commits\) on page 269](#)
- [days-to-keep-error-logs \(Batch Commits\) on page 271](#)
- [maximum-aggregate-pool \(Batch Commits\) on page 283](#)
- [maximum-entries \(Batch Commits\) on page 283](#)
- [maximum-entries on page 283](#)
- [server \(Batch Commits\) on page 292](#)
- [traceoptions \(Batch Commits\) on page 305](#)

## CHAPTER 6

# Managing Configurations

- [Understanding How the Junos OS Configuration Is Stored on page 125](#)
- [Comparing Configuration Changes with a Prior Version on page 126](#)
- [Understanding the show | compare | display xml Command Output on page 128](#)
- [Returning to the Most Recently Committed Junos OS Configuration on page 134](#)
- [Returning to a Previously Committed Junos OS Configuration on page 135](#)
- [Saving a Configuration to a File on page 140](#)
- [Additional Details About Specifying Junos OS Statements and Identifiers on page 141](#)
- [Loading a Configuration from a File or the Terminal on page 144](#)
- [Examples: Loading a Configuration from a File on page 147](#)
- [Creating and Returning to a Rescue Configuration on page 149](#)
- [Compressing the Current Configuration File on page 149](#)
- [Example: Protecting the Junos OS Configuration from Modification or Deletion on page 151](#)
- [Synchronizing Routing Engines on page 158](#)
- [Configuring Multiple Routing Engines to Synchronize Committed Configurations Automatically on page 161](#)

### Understanding How the Junos OS Configuration Is Stored

When you edit a configuration, you work in a copy of the current configuration to create a candidate configuration. The changes you make to the candidate configuration are visible in the CLI immediately, so if multiple users are editing the configuration at the same time, all users can see all changes.

To have a candidate configuration take effect, you *commit* the changes. At this point, the candidate file is checked for proper syntax, activated, and marked as the current, operational software configuration file. If multiple users are editing the configuration, when you commit the candidate configuration, all changes made by all the users take effect.

In addition to saving the current configuration, the CLI saves the current operational version and the previous 49 versions of committed configurations. The most recently committed configuration is version 0, which is the current operational version and the

default configuration that the system returns to if you roll back to a previous configuration. The oldest saved configuration is version 49.

By default, Junos OS saves the current configuration and three previous versions of the committed configuration on the CompactFlash card. The currently operational Junos OS configuration is stored in the file **juniper.conf.gz**, and the last three committed configurations are stored in the files **juniper.conf.1.gz**, **juniper.conf.2.gz**, and **conf.3.gz**. These four files are located in the router or switch's CompactFlash card in the directory **/config**.

The remaining 46 previous versions of committed configurations, the files **juniper.conf.4** through **juniper.conf.49**, are stored in the directory **/var/db/config** on the hard disk.

**Related  
Documentation**

- *Using Junos OS to Specify the Number of Configurations Stored on the CompactFlash Card*
- [Returning to the Most Recently Committed Junos OS Configuration on page 134](#)
- [Returning to a Previously Committed Junos OS Configuration on page 135](#)
- [Loading a Configuration from a File or the Terminal on page 144](#)

---

## Comparing Configuration Changes with a Prior Version

---

In configuration mode only, when you have made changes to the configuration and want to compare the candidate configuration with a prior version, you can use the **compare** command to display the configuration. The **compare** command compares the candidate configuration with either the current committed configuration or a configuration file and displays the differences between the two configurations. To compare configurations, specify the **compare** command after the pipe:

```
[edit]
user@host# show | compare (filename| rollback n)
```

**filename** is the full path to a configuration file. The file must be in the proper format: a hierarchy of statements.

**n** is the index into the list of previously committed configurations. The most recently saved configuration is number 0, and the oldest saved configuration is number 49. If you do not specify arguments, the candidate configuration is compared against the active configuration file (**/config/juniper.conf**).

The comparison output uses the following conventions:

- Statements that are only in the candidate configuration are prefixed with a plus sign (+).
- Statements that are only in the comparison file are prefixed with a minus sign (-).
- Statements that are unchanged are prefixed with a single blank space ( ).

The following example shows various changes, then a comparison of the candidate configuration with the active configuration, showing only the changes made at the **[edit protocols bgp]** hierarchy level:

```
[edit]
user@host# edit protocols bgp
[edit protocols bgp]
user@host# show
group my-group {
 type internal;
 hold-time 60;
 advertise-inactive;
 allow 10.1.1.1/8;
}
group fred {
 type external;
 peer-as 33333;
 allow 10.2.2.2/8;
}
group test-peers {
 type external;
 allow 10.3.3.3/8;
}
[edit protocols bgp]
user@host# set group my-group hold-time 90
[edit protocols bgp]
user@host# delete group my-group advertise-inactive
[edit protocols bgp]
user@host# set group fred advertise-inactive
[edit protocols bgp]
user@host# delete group test-peers
[edit protocols bgp]
user@host# show | compare
[edit protocols bgp group my-group]
- hold-time 60;
+ hold-time 90;
- advertise-inactive;
[edit protocols bgp group fred]
+ advertise-inactive;
[edit protocols bgp]
- group test-peers {
 - type external;
 - allow 10.3.3.3/8;
}
[edit protocols bgp]
user@host# show
group my-group {
 type internal;
 hold-time 90;
 allow 10.1.1.1/8;
}
group fred {
 type external;
 advertise-inactive;
 peer-as 33333;
```

```
 allow 10.2.2.2/8;
}
```

**Related  
Documentation**

- [Creating and Returning to a Rescue Configuration on page 138](#)

---

## Understanding the `show | compare | display xml` Command Output

The **compare | display xml** filter compares the candidate configuration with the current committed configuration and displays the differences between the two configurations in XML. To compare configurations, enter **compare | display xml** after the pipe ( | ) symbol in either operational or configuration mode.

Example in operational mode:

```
user@host> show configuration | compare | display xml
```

Example in configuration mode:

```
[edit]
user@host# show | compare | display xml
```

You can enter a specific configuration hierarchy immediately preceding the **compare** filter, for example, **show configuration system syslog | compare | display xml**. In configuration mode, you can navigate to a hierarchy where the command is applied.

The differences from the compare filter function are output in XML. The **configuration** tag starts the output. The context for changes is established with hierarchy name tags relative to the root of the compare. For element changes, an **operation** attribute are output in the tag where a change occurs. This attribute has the value **create**, **delete**, or **merge**. For metadata changes, the metadata name is specified. For example, if a statement is marked inactive, the **inactive="inactive"** attribute and value are output. The nc namespace is used when necessary to indicate that an attribute is in the NETCONF namespace rather than the Junos OS namespace.

The following sections explain the XML that is generated for particular types of configuration changes. The corresponding text changes are shown for comparison.

- [Adding a Statement \(create Operation\) on page 129](#)
- [Deleting a Statement \(delete Operation\) on page 129](#)
- [Changing a Statement \(delete and create Operations\) on page 130](#)
- [Changing Metadata \(inactive Attribute and Operation\) on page 131](#)
- [Adding an Annotation \(comment Tag and create Operation\) on page 132](#)
- [Changing an Annotation \(comment Tag, and delete and create Operations\) on page 132](#)
- [Adding a Statement Inside a Container \(create Operation, and insert and key Attributes\) on page 133](#)
- [Changing the Order Inside a Container \(merge Operation, and insert and key Attributes\) on page 134](#)



## Adding a Statement (create Operation)

The following example shows the addition of IPv4 address 2.2.2.2 to unit 1. The tags through **name** provide the context for the addition. The **operation="create"** attribute indicates that a **unit** statement was created and is defined by the configuration within the **unit** tag.

```
[edit interfaces ge-0/0/0]
user@host> show configuration | compare
[edit interfaces ge-0/0/0]
+ unit 1 {
+ family inet {
+ address 2.2.2.2/32;
+ }
+ }

[edit interfaces ge-0/0/0]
user@host# show | compare | display xml
<configuration>
 <interfaces>
 <interface>
 <name>ge-0/0/0</name>
 <unit nc:operation="create">
 <name>1</name>
 <family>
 <inet>
 <address>
 <name>2.2.2.2/32</name>
 </address>
 </inet>
 </family>
 </unit>
 </interface>
 </interfaces>
</configuration>
```

## Deleting a Statement (delete Operation)

The following example shows the deletion of a simple statement in the configuration hierarchy. The tags through **system** provide the context for the deletion. The **operation="delete"** attribute indicates that the **services** statement was deleted. The configuration following the **services** statement was deleted though is not output.

```
[edit system]
user@host> show configuration | compare
[edit system]
- services {
- ftp;
- }

[edit system]
user@host# show | compare | display xml
<configuration>
 <system>
 <services operation="delete"/>
 </system>
</configuration>
```

The following example shows the deletion of unit 1 from the ge-0/0/0 interface. The configuration following the **unit** statement was deleted though is not output.

```
[edit interfaces ge-0/0/0]
user@host> show configuration | compare
[edit interfaces ge-0/0/0]
- unit 1 {
- family inet {
- address 2.2.2.2/32;
- }
- }

[edit interfaces ge-0/0/0]
user@host# show | compare | display xml
<configuration>
 <interfaces>
 <interface>
 <name>ge-0/0/0</name>
 <unit nc:operation="delete">
 <name>1</name>
 </unit>
 </interface>
 </interfaces>
</configuration>
```

The following example shows the deletion of the **apply-groups** configuration. The groups that are deleted are not output.

```
[edit]
user@host# delete apply-groups

[edit]
user@host> show configuration | compare
[edit]
- apply-groups [g1 g2 g3];

[edit]
user@host# show | compare | display xml
<configuration>
 <apply-groups operation="delete"/>
</configuration>
```

## Changing a Statement (delete and create Operations)

The following example shows a change in a statement in the hierarchy. The tags through **system** provide the context for the change. The **operation="delete"** attribute indicates that the **host-name** statement was deleted. The configuration following the **host-name** statement was deleted though is not output. The **operation="create"** attribute indicates that a **host-name** statement was created and is defined by the configuration within the **host-name** tag.

```
[edit system]
user@host> show configuration | compare
[edit system]
- host-name router1;
+ host-name router2;

[edit system]
user@host# show | compare | display xml
```

```

<configuration>
 <system>
 <host-name nc:operation="delete"/>
 <host-name nc:operation="create">router2</host-name>
 </system>
</configuration>

```

## Changing Metadata (inactive Attribute and Operation)

The following example shows the inactivation of a statement in the hierarchy. The tags through **system** provide the context for the change. The **inactive="inactive"** attribute indicates that the **syslog** statement was inactivated.

```

[edit system]
user@host> show configuration | compare
[edit system]
! inactive: syslog { ... }

[edit system]
user@host# show | compare | display xml
<configuration>
 <system>
 <syslog inactive="inactive"/>
 </system>
</configuration>

```

The following example shows the addition of an inactive **syslog** statement. The **operation="create"** attribute indicates that the **syslog** statement was created and is defined by the configuration within the **syslog** tag. The **inactive="inactive"** attribute indicates that the **syslog** statement was inactivated.

```

[edit system]
user@host> show configuration | compare
[edit system]
+ inactive: syslog {
+ file foo {
+ any any;
+ }
+ }

[edit system]
user@host# show | compare | display xml
<configuration>
 <system>
 <syslog nc:operation="create"
 inactive="inactive">
 <file>
 <name>foo</name>
 <contents>
 <name>any</name>
 <any/>
 </contents>
 </file>
 </syslog>
 </system>
</configuration>

```

## Adding an Annotation (comment Tag and create Operation)

The following example shows the addition of a comment to a statement. The tags through **syslog** provide the context for the annotation. The **operation="create"** attribute for the **junos:comment** tag indicates that a comment was added to the **[edit system syslog]** hierarchy.

```
[edit system]
user@host> show configuration | compare
[edit system]
+ /* my-comments-simple */
 syslog { ... }

[edit system]
user@host# show | compare | display xml
<configuration>
 <system>
 <junos:comment nc:operation="create">/* my-comments-simple
*/</junos:comment>
 <syslog/>
 </system>
</configuration>
```

The following example shows the addition of a comment to a statement. The tags through **syslog** provide the context for the annotation. The **operation="create"** attribute for the **junos:comment** tag indicates that a comment was added to the **[edit system syslog]** hierarchy for the statement output within the **syslog** tag.

```
[edit system syslog]
user@host> show configuration | compare
+ /* my-comments-ele */
 file f1 { ... }

[edit system syslog]
user@host# show | compare | display xml
<configuration>
 <system>
 <syslog>
 <junos:comment nc:operation="create">/* my-comments-elem
*/</junos:comment>
 <file>
 <name>f1</name>
 </file>
 </syslog>
 </system>
</configuration>
```

## Changing an Annotation (comment Tag, and delete and create Operations)

The following example shows the change of a comment for a statement. The tags through **system** provide the context for the annotation. The **operation="delete"** attribute for the **junos:comment** tag indicates that a comment was deleted from the **[edit system]** hierarchy at the **syslog** statement. The **operation="create"** attribute for the **junos:comment** tag

indicates that a comment was added to the **[edit system]** hierarchy for the **syslog** statement.

```
[edit system]
user@host> show configuration | compare
- /* my-comments-1 */
+ /* my-comments-2 */
 syslog { ... }

[edit system]
user@host# show | compare | display xml
<configuration>
 <system>
 <junos:comment nc:operation="delete"/>
 <junos:comment nc:operation="create">/* my-comments-2
*/</junos:comment>
 <syslog/>
 </system>
</configuration>
```

### Adding a Statement Inside a Container (create Operation, and insert and key Attributes)

The following example shows the addition of a **file** statement at the **[edit system syslog]** hierarchy. The tags through **syslog** provide the context for the addition. The **operation="create"** attribute for the **file** tag indicates that a **file** statement was added. The **yang:insert="after"** attribute indicates that the file was added after the position indicated by the **yang:key="[name='file-1']"** attribute. The **file-1** value represents the position within the existing **file** statements, where one is the first file. In this example, the new **file** statement was added after the first file.

```
[edit system syslog]
user@host> show configuration | compare
[edit system syslog]
 file file-1 { ... }
+ file file-2 {
+ any any;
+ }

[edit system syslog]
user@host# show | compare | display xml
<configuration>
 <system>
 <syslog>
 <file nc:operation="create"
 yang:insert="after"
 yang:key="[name='file-1']">
 <name>file-2</name>
 <contents>
 <name>any</name>
 <any/>
 </contents>
 </file>
 </syslog>
 </system>
</configuration>
```

## Changing the Order Inside a Container (merge Operation, and insert and key Attributes)

The following example shows the change in order of **file** statements at the **[edit system syslog]** hierarchy. The tags through **syslog** provide the context for the change. The **operation="merge"** attribute for the **file** tag indicates that an existing **file** statement was moved. The **yang:insert="after"** attribute indicates that the file was moved after the file in the position indicated by the **yang:key="[name='file-1']"** attribute. The **file-1** value represents a position within the existing **file** statements, where one is the first file. The value at the **name** tag, **file-3**, represents a position within the existing file statements. In this example, the **file** statement in the third position was moved after the first file.

```
[edit system syslog]
user@host> show configuration | compare
[edit system syslog]
 file f1 { ... }
! file f3 { ... }

[edit system syslog]
user@host# show | compare | display xml
<configuration>
 <system>
 <syslog>
 <file nc:operation="merge"
 yang:insert="after"
 yang:key="[name='file-1']">
 <name>file-3</name>
 </file>
 </syslog>
 </system>
 </configuration>
```

### Related Documentation

- [Using Regular Expressions with the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 188](#)
- [Pipe \( | \) Filter Functions in the Junos OS Command-Line Interface on page 190](#)
- [Using the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 187](#)

---

## Returning to the Most Recently Committed Junos OS Configuration

To return to the most recently committed configuration and load it into configuration mode without activating it, use the **rollback** configuration mode command:

```
[edit]
user@host# rollback

load complete
```

To activate the configuration to which you rolled back, use the **commit** command:

```
[edit]
user@host# rollback
load complete
[edit]
user@host# commit
```

- Related Documentation**
- [Rolling Back Junos OS Configuration Changes on page 44](#)
  - [Returning to a Previously Committed Junos OS Configuration on page 135](#)
  - [Understanding How the Junos OS Configuration Is Stored on page 125](#)

## Returning to a Previously Committed Junos OS Configuration

This topic explains how you can return to a configuration prior to the most recently committed one, and contains the following sections:

- [Returning to a Configuration Prior to the One Most Recently Committed on page 135](#)
- [Displaying Previous Configurations on page 135](#)
- [Comparing Configuration Changes with a Prior Version on page 136](#)
- [Creating and Returning to a Rescue Configuration on page 138](#)
- [Saving a Configuration to a File on page 139](#)

### Returning to a Configuration Prior to the One Most Recently Committed

To return to a configuration prior to the most recently committed one, include the configuration number, 0 through 49, in the **rollback** command. The most recently saved configuration is number 0 (which is the default configuration to which the system returns), and the oldest saved configuration is number 49.

```
[edit]
user@host# rollback number
load complete
```

### Displaying Previous Configurations

To display previous configurations, including the rollback number, date, time, the name of the user who committed changes, and the method of commit, use the **rollback ?** command.

```
[edit]
user@host# rollback ?
Possible completions:
<[Enter]> Execute this command
<number> Numeric argument
0 2005-02-27 12:52:10 PST by abc via cli
1 2005-02-26 14:47:42 PST by def via cli
2 2005-02-14 21:55:45 PST by ghi via cli
3 2005-02-10 16:11:30 PST by jkl via cli
4 2005-02-10 16:02:35 PST by mno via cli
5 2005-03-16 15:10:41 PST by pqr via cli
6 2005-03-16 14:54:21 PST by stu via cli
7 2005-03-16 14:51:38 PST by vwx via cli
8 2005-03-16 14:43:29 PST by yzz via cli
9 2005-03-16 14:15:37 PST by abc via cli
10 2005-03-16 14:13:57 PST by def via cli
11 2005-03-16 12:57:19 PST by root via other
12 2005-03-16 10:45:23 PST by root via other
```

```
13 2005-03-16 10:08:13 PST by root via other
14 2005-03-16 01:20:56 PST by root via other
15 2005-03-16 00:40:37 PST by ghi via cli
16 2005-03-16 00:39:29 PST by jkl via cli
17 2005-03-16 00:32:36 PST by mno via cli
18 2005-03-16 00:31:17 PST by pqr via cli
19 2005-03-15 19:59:00 PST by stu via cli
20 2005-03-15 19:53:39 PST by vwx via cli
21 2005-03-15 18:07:19 PST by yzz via cli
22 2005-03-15 17:59:03 PST by abc via cli
23 2005-03-15 15:05:14 PST by def via cli
24 2005-03-15 15:04:51 PST by ghi via cli
25 2005-03-15 15:03:42 PST by jkl via cli
26 2005-03-15 15:01:52 PST by mno via cli
27 2005-03-15 14:58:34 PST by pqr via cli
28 2005-03-15 13:09:37 PST by root via other
29 2005-03-12 11:01:20 PST by stu via cli
30 2005-03-12 10:57:35 PST by vwx via cli
31 2005-03-11 10:25:07 PST by yzz via cli
32 2005-03-10 23:40:58 PST by abc via cli
33 2005-03-10 23:40:38 PST by def via cli
34 2005-03-10 23:14:27 PST by ghi via cli
35 2005-03-10 23:10:16 PST by jkl via cli
36 2005-03-10 23:01:51 PST by mno via cli
37 2005-03-10 22:49:57 PST by pqr via cli
38 2005-03-10 22:24:07 PST by stu via cli
39 2005-03-10 22:20:14 PST by vwx via cli
40 2005-03-10 22:16:56 PST by yzz via cli
41 2005-03-10 22:16:41 PST by abc via cli
42 2005-03-10 20:44:00 PST by def via cli
43 2005-03-10 20:43:29 PST by ghi via cli
44 2005-03-10 20:39:14 PST by jkl via cli
45 2005-03-10 20:31:30 PST by root via other
46 2005-03-10 18:57:01 PST by mno via cli
47 2005-03-10 18:56:18 PST by pqr via cli
48 2005-03-10 18:47:49 PST by stu via cli
49 2005-03-10 18:47:34 PST by vw via cli
| Pipe through a command
[edit]
```

## Comparing Configuration Changes with a Prior Version

In configuration mode only, when you have made changes to the configuration and want to compare the candidate configuration with a prior version, you can use the **compare** command to display the configuration. The **compare** command compares the candidate configuration with either the current committed configuration or a configuration file and displays the differences between the two configurations. To compare configurations, specify the **compare** command after the pipe:

```
[edit]
user@host# show | compare (filename| rollback n)
```

***filename*** is the full path to a configuration file. The file must be in the proper format: a hierarchy of statements.



*n* is the index into the list of previously committed configurations. The most recently saved configuration is number 0, and the oldest saved configuration is number 49. If you do not specify arguments, the candidate configuration is compared against the active configuration file (`/config/juniper.conf`).

The comparison output uses the following conventions:

- Statements that are only in the candidate configuration are prefixed with a plus sign (+).
- Statements that are only in the comparison file are prefixed with a minus sign (-).
- Statements that are unchanged are prefixed with a single blank space ( ).

The following example shows various changes, then a comparison of the candidate configuration with the active configuration, showing only the changes made at the **[edit protocols bgp]** hierarchy level:

```
[edit]
user@host# edit protocols bgp
[edit protocols bgp]
user@host# show
group my-group {
 type internal;
 hold-time 60;
 advertise-inactive;
 allow 10.1.1.1/8;
}
group fred {
 type external;
 peer-as 33333;
 allow 10.2.2.2/8;
}
group test-peers {
 type external;
 allow 10.3.3.3/8;
}
[edit protocols bgp]
user@host# set group my-group hold-time 90
[edit protocols bgp]
user@host# delete group my-group advertise-inactive
[edit protocols bgp]
user@host# set group fred advertise-inactive
[edit protocols bgp]
user@host# delete group test-peers
[edit protocols bgp]
user@host# show | compare
[edit protocols bgp group my-group]
-hold-time 60;
+hold-time 90;
-advertise-inactive;
[edit protocols bgp group fred]
+advertise-inactive;
[edit protocols bgp]
-group test-peers {
```

```
-type external;
-allow 10.3.3.3/8;
}
[edit protocols bgp]
user@host# show
group my-group {
 type internal;
 hold-time 90;
 allow 10.1.1.1/8;
}
group fred {
 type external;
 advertise-inactive;
 peer-as 3333;
 allow 10.2.2.2/8;
}
```

## Creating and Returning to a Rescue Configuration

A rescue configuration allows you to define a known working configuration or a configuration with a known state that you can roll back to at any time. This alleviates the necessity of having to remember the rollback number with the **rollback** command. You use the rescue configuration when you need to roll back to a known configuration or as a last resort if your router or switch configuration and the backup configuration files become damaged beyond repair.

To save the most recently committed configuration as the rescue configuration so that you can return to it at any time, issue the **request system configuration rescue save** command:

```
user@host> request system configuration rescue save
```

To return to the rescue configuration, use the **rollback rescue** configuration mode command:

```
[edit]
user@host# rollback rescue
load complete
```



**NOTE:** If the rescue configuration does not exist, or if the rescue configuration is not a complete, viable configuration, then the rollback command fails, an error message appears, and the current configuration remains active.

To activate the rescue configuration that you have loaded, use the **commit** command:

```
[edit]
user@host# rollback rescue
load complete
[edit]
user@host# commit
```

To delete an existing rescue configuration, issue the **request system configuration rescue delete** command:

```
user@host> request system configuration rescue delete
user@host>
```

For more information about the **request system configuration rescue delete** and **request system configuration rescue save** commands, see the [CLI Explorer](#).

## Saving a Configuration to a File

Save Junos OS configuration to a file so that you can edit it with a text editor of your choice. You can save your current configuration to an ASCII file, which saves the configuration in its current form, including any uncommitted changes. If more than one user is modifying the configuration, all changes made by all users are saved.

To save software configuration changes to an ASCII file, use the **save** configuration mode command:

```
[edit]
user@host# save filename
[edit]
user@host#
```

The contents of the current level of the statement hierarchy (and below) are saved, along with the statement hierarchy containing it. This allows a section of the configuration to be saved, while fully specifying the statement hierarchy.

By default, the configuration is saved to a file in your home directory, which is on the flash drive.

When you issue this command from anywhere in the hierarchy (except the top level), a **replace** tag is automatically included at the beginning of the file. You can use the **replace** tag to control how a configuration is loaded from a file.

```
user@host> file show /var/home/user/myconf
replace:
protocols {
 bgp {
 disable;
 group int {
 type internal;
 }
 }
 isis {
 disable;
 interface all {
 level 1 disable;
 }
 interface fxp0.0 {
 disable;
 }
 }
 ospf {
 traffic-engineering;
 reference-bandwidth 4g;
 ...
 }
}
```

```
}
```

**Related Documentation**

- [Returning to the Most Recently Committed Junos OS Configuration on page 134](#)
- [Loading a Configuration from a File or the Terminal on page 144](#)
- [Specifying Filenames and URLs on page 176](#)

---

## Saving a Configuration to a File

Save Junos OS configuration to a file so that you can edit it with a text editor of your choice. You can save your current configuration to an ASCII file, which saves the configuration in its current form, including any uncommitted changes. If more than one user is modifying the configuration, all changes made by all users are saved.

To save software configuration changes to an ASCII file, use the **save** configuration mode command:

```
[edit]
user@host# save filename
[edit]
user@host#
```

The contents of the current level of the statement hierarchy (and below) are saved, along with the statement hierarchy containing it. This allows a section of the configuration to be saved, while fully specifying the statement hierarchy.

By default, the configuration is saved to a file in your home directory, which is on the flash drive.

When you issue this command from anywhere in the hierarchy (except the top level), a **replace** tag is automatically included at the beginning of the file. You can use the **replace** tag to control how a configuration is loaded from a file.

```
user@host> file show /var/home/user/myconf
replace:
protocols {
 bgp {
 disable;
 group int {
 type internal;
 }
 }
 isis {
 disable;
 interface all {
 level 1 disable;
 }
 interface fxp0.0 {
 disable;
 }
 }
 ospf {
 traffic-engineering;
```

```

 reference-bandwidth 4g;
 ...
}
}

```

## Additional Details About Specifying Junos OS Statements and Identifiers

This topic provides more detailed information about CLI container and leaf statements so that you can better understand how you must specify them when creating ASCII configuration files. It also describes how the CLI performs type checking to verify that the data you entered is in the correct format.

- [Specifying Statements on page 141](#)
- [Performing CLI Type Checking on page 143](#)

### Specifying Statements

Statements are shown one of two ways, either with braces or without:

- Statement name and identifier, with one or more lower level statements enclosed in braces:

```

statement-name1 identifier-name {
 statement-name2;
 additional-statements;
}

```

- Statement name, identifier, and a single identifier:

```

statement-name identifier-name1 identifier-name2;

```

The **statement-name** is the name of the statement.

The **identifier-name** is a name or other string that uniquely identifies an instance of a statement. An identifier is used when a statement can be specified more than once in a configuration.

When specifying a statement, you must specify either a statement name or an identifier name, or both, depending on the statement hierarchy.

You specify identifiers in one of the following ways:

- **identifier-name**—The **identifier-name** is a keyword used to uniquely identify a statement when a statement can be specified more than once in a statement.
- **identifier-name value**—The **identifier-name** is a keyword, and the **value** is a required option variable.
- **identifier-name [value1 value2 value3 ...]**—The **identifier-name** is a keyword that accepts multiple values. The brackets are required when you specify a set of values; however, they are optional when you specify only one value.

The following examples illustrate how statements and identifiers are specified in the configuration:

```
protocol { # Top-level statement (statement-name).
 ospf { # Statement under "protocol" (statement-name).
 area 0.0.0.0 { # OSPF area "0.0.0.0" (statement-name identifier-name),
 interface so-0/0/0 { # which contains an interface named "so-0/0/0."
 hello-interval 25; # Identifier and value (identifier-name value).
 priority 2; # Identifier and value (identifier-name value).
 disable; # Flag identifier (identifier-name).
 }
 interface so-0/0/1; # Another instance of "interface," named so-0/0/1,
 } # this instance contains no data, so no braces
 } # are displayed.
}

policy-options { # Top-level statement (statement-name).
 term term1 { # Statement under "policy-options"
 # (statement-name value).
 from { # Statement under "term" (statement-name).
 route-filter 10.0.0.0/8 orlonger reject; # One identifier ("route-filter")
 with
 route-filter 127.0.0.0/8 orlonger reject; # multiple values.
 route-filter 128.0.0.0/16 orlonger reject;
 route-filter 149.20.64.0/24 orlonger reject;
 route-filter 172.16.0.0/12 orlonger reject;
 route-filter 191.255.0.0/16 orlonger reject;
 }
 then { # Statement under "term" (statement-name).
 next term; # Identifier (identifier-name).
 }
 }
}
```

When you create an ASCII configuration file, you can specify statements and identifiers in one of the following ways. However, each statement has a preferred style, and the CLI uses that style when displaying the configuration in response to a configuration mode **show** command.

- Statement followed by identifiers:

***statement-name identifier-name [...] identifier-name value [...];***

- Statement followed by identifiers enclosed in braces:

```
statement-name {
 identifier-name;
 [...]
 identifier-name value;
 [...]
}
```

- For some repeating identifiers, you can use one set of braces for all the statements:

```
statement-name {
 identifier-name value1;
 identifier-name value2;
}
```

## Performing CLI Type Checking

When you specify identifiers and values, the CLI performs type checking to verify that the data you entered is in the correct format. For example, for a statement in which you must specify an IP address, the CLI requires you to enter an address in a valid format. If you have not, an error message indicates what you need to type. [Table 7 on page 143](#) lists the data types the CLI checks.

**Table 7: CLI Configuration Input Types**

| Data Type                                                                                             | Format                                                                                         | Examples                                                                                                                                                                                                   |
|-------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Physical interface name (used in the <b>[edit interfaces]</b> hierarchy)                              | <i>type-fpc/pic/port</i>                                                                       | Correct: so-0/0/1<br><br>Incorrect: so-0                                                                                                                                                                   |
| Full interface name                                                                                   | <i>type-fpc/pic/port&lt;:channel&gt;.logical</i>                                               | Correct: so-0/0/1.0<br><br>Incorrect: so-0/0/1                                                                                                                                                             |
| Full or abbreviated interface name (used in places other than the <b>[edit interfaces]</b> hierarchy) | <i>type-&lt;fpc&lt;/pic/port&gt;&gt;&lt;&lt;:channel&gt;.logical&gt;</i>                       | Correct: so, so-1, so-1/2/3:4.5                                                                                                                                                                            |
| IP address                                                                                            | <i>0xhex-bytesoctet&lt;.octet&lt;.octet.&lt;octet&gt;&gt;&gt;</i>                              | Correct: 1.2.3.4, 0x01020304, 128.8.1, 128.8<br><br>Sample translations:<br><br>1.2.3 becomes 1.2.3.0<br>0x01020304 becomes 1.2.3.4<br>0x010203 becomes 0.1.2.3                                            |
| IP address (destination prefix) and prefix length                                                     | <i>0xhex-bytes&lt;/length&gt;octet&lt;octet&lt;.octet.&lt;octet&gt;&gt;&gt;&lt;/length&gt;</i> | Correct: 10/8, 128.8/16, 1.2.3.4/32, 1.2.3.4<br><br>Sample translations:<br><br>1.2.3 becomes 1.2.3.0/32<br>0x01020304 becomes 1.2.3.4/32<br>0x010203 becomes 0.1.2.3/32<br>default becomes 0.0.0.0/0      |
| International Organization for Standardization (ISO) address                                          | <i>hex-nibble&lt;hex-nibble ...&gt;</i>                                                        | Correct: 47.1234.2345.3456.00, 47123423453456.00, 47.12.34.23.45.34.56.00<br><br>Sample translations:<br><br>47123456 becomes 47.1234.56<br>47.12.34.56 becomes 47.1234.56<br>4712.3456 becomes 47.1234.56 |

Table 7: CLI Configuration Input Types (*continued*)

| Data Type                 | Format                                                                                | Examples                                                                                                                                                                                           |
|---------------------------|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OSPF area identifier (ID) | <i>0xhex-bytesoctet&lt;.octet&lt;.octet&lt;.octet<br/>&gt;&gt;&gt; decimal-number</i> | <p>Correct: 54, 0.0.0.54, 0x01020304, 1.2.3.4</p> <p>Sample translations:</p> <p>54 becomes 0.0.0.54</p> <p>257 becomes 0.0.1.1</p> <p>128.8 becomes 128.8.0.0</p> <p>0x010203 becomes 0.1.2.3</p> |

**Related Documentation** • [Entering and Exiting the Junos OS CLI Configuration Mode on page 62](#)

## Loading a Configuration from a File or the Terminal

You can create a file containing configuration data for a device running Junos OS, copy the file to the local router, and then load the file into the CLI. After you have loaded the file, you can commit it to activate the configuration on the router, or you can edit the configuration interactively using the CLI and commit it at a later time.

You can also create a configuration while typing at the terminal and then load it. Loading a configuration from the terminal is generally useful when you are cutting existing portions of the configuration and pasting them elsewhere in the configuration.

To load an existing configuration file that is located on the router, use the **load** configuration mode command:

```
[edit]
user@host# load (factory-default | merge | override | patch | replace | set | update)
filename <relative> <json>
```

For information about specifying the filename, see [“Specifying Filenames and URLs” on page 176](#).

To load a configuration from the terminal, use the following version of the **load** configuration mode command. Press Ctrl-d to end the input.

```
[edit]
user@host# load (factory-default | merge | override | patch | replace | set | update)
terminal <relative> <json>
```

To replace an entire configuration, specify the **override** option at any level of the hierarchy. A **load override** operation completely replaces the current candidate configuration with the file you are loading. Thus, if you saved a complete configuration, use this option.

An **override** operation discards the current candidate configuration and loads the configuration in **filename** or the configuration that you type at the terminal. When you use the **override** option and commit the configuration, all system processes reparse the configuration. .



To replace portions of a configuration, specify the **replace** option. The **load replace** operation looks for **replace:** tags that you added to the loaded file, and replaces the parts of the candidate configuration with whatever is specified after the tag. This is useful when you want more control over exactly what is being changed. For this operation to work, you must include **replace:** tags in the file or configuration you type at the terminal. The software searches for the **replace:** tags, deletes the existing statements of the same name, if any, and replaces them with the incoming configuration. If there is no existing statement of the same name, the **replace** operation adds to the configuration the statements marked with the **replace:** tag.

If, in an **override** or **merge** operation, you specify a file or type text that contains **replace:** tags, the **replace:** tags are ignored and the **override** or **merge** operation is performed.

If you are performing a **replace** operation and the file you specify or text you type does not contain any **replace:** tags, the **replace** operation is effectively equivalent to a **merge** operation. This might be useful if you are running automated scripts and cannot know in advance whether the scripts need to perform a **replace** or a **merge** operation. The scripts can use the **replace** operation to cover either case.

The **load merge** operation merges the configuration from the saved file or terminal with the existing candidate configuration. This is useful if you are adding new configuration sections. For example, suppose that you are adding a BGP configuration to the **[edit protocols]** hierarchy level, where there was no BGP configuration before. You can use the **load merge** operation to combine the incoming configuration with the existing candidate configuration. If the existing configuration and the incoming configuration contain conflicting statements, the statements in the incoming configuration override those in the existing configuration.

To replace only those parts of the configuration that have changed, specify the **update** option at any level of the hierarchy. The **load update** operation compares the candidate configuration and the new configuration data, and only changes the parts of the candidate configuration that are different from the new configuration. You would use this, for example, if there is an existing BGP configuration and the file you are loading changes it in some way.

The **merge**, **override**, and **update** options support loading configuration data in JavaScript Object Notation (JSON) format. When loading configuration data that uses JSON format, you must specify the **json** option in the command.

To change part of the configuration with a patch file, specify the **patch** option. The **load patch** operation loads a file or terminal input that contains configuration changes. First, on a device that already has the configuration changes, you type the **show | compare** command to output the differences between two configurations. Then you can load the differences on another router. The advantage of the **load patch** command is that it saves you from having to copy snippets from different hierarchy levels into a text file prior to loading them into the target device. This might be a useful time saver if you are configuring several devices with the same options. For example, suppose that you configure a routing policy on router1 and you want to replicate the policy configuration on router2, router3, and router4. You can use the **load patch** operation.

First, run the **show | compare** command.

```
user@router1# show | compare rollback 3
[edit protocols ospf]
+ export default-static;
- export static-default
[edit policy-options]
+ policy-statement default-static {
+ from protocol static;
+ then accept;
+ }
```

Copy the output of the **show | compare** command to the clipboard, making sure to include the hierarchy levels. On router2, router3, and router4, type **load patch terminal** and paste the output. Press Enter and then press Ctrl-d to end the operation. If the patch input specifies different values for an existing statement, the patch input overrides the existing statement.

To use the **merge**, **replace**, **set**, or **update** option without specifying the full hierarchy level, specify the **relative** option. This option loads the incoming configuration relative to your current edit point in the configuration hierarchy. For example:

```
[edit system]
user@host# show static-host-mapping
bob sysid 987.654.321ab
[edit system]
user@host# load replace terminal relative
[Type ^D at a new line to end input]
replace: static-host-mapping {
 bob sysid 0123.456.789bc;
}
load complete
[edit system]
user@host# show static-host-mapping
bob sysid 0123.456.789bc;
```

To load a configuration that contains **set** configuration mode commands, specify the **set** option. This option executes the configuration instructions line by line as they are stored in a file or from a terminal. The instructions can contain any configuration mode command, such as **set**, **edit**, **exit**, and **top**.

To copy a configuration file from another network system to the local router, you can use the SSH and Telnet utilities, as described in the [CLI Explorer](#).



**NOTE:** If you are using Junos OS in a Common Criteria environment, system log messages are created whenever a secret attribute is changed (for example, password changes or changes to the RADIUS shared secret). These changes are logged during the following configuration load operations:

```
load merge
load replace
load override
load update
```

For more information, see the *Secure Configuration Guide for Common Criteria and Junos-FIPS*.

---

## Examples: Loading a Configuration from a File

Figure 6: Overriding the Current Configuration



Figure 7: Using the replace Option

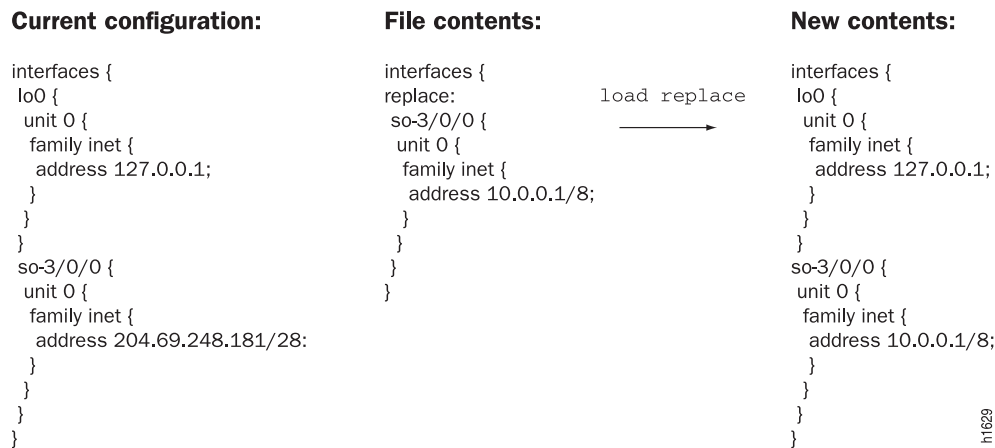


Figure 8: Using the merge Option

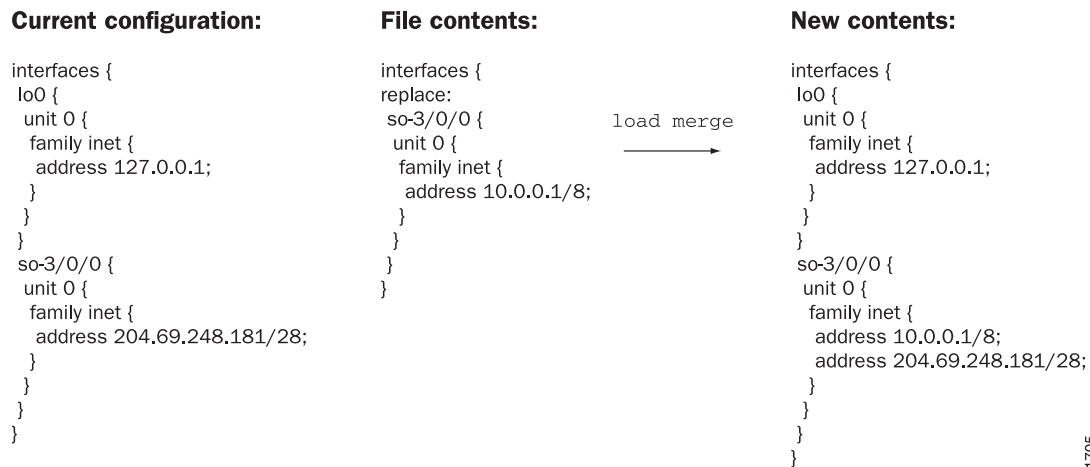


Figure 9: Using a Patch File

**Current configuration:**

```

interfaces {
 fxp0 {
 unit 0 {
 family inet {
 address 192.168.6.193/24;
 }
 }
 }
 lo0 {
 unit 0 {
 family inet {
 address 127.0.0.1/32;
 }
 }
 }
}

```

**File contents:**

```

(edit interfaces)
+ so-0/0/0 {
+ unit 0 {
+ family inet {
+ address 10.0.0.1/8;
+ }
+ }
+ }

```

load patch

**New contents:**

```

interfaces {
 so-0/0/0 {
 unit 0 {
 family inet {
 address 10.0.0.1/8;
 }
 }
 }
 fxp0 {
 unit 0 {
 family inet {
 address 192.168.6.193/24;
 }
 }
 }
 lo0 {
 unit 0 {
 family inet {
 address 127.0.0.1/32;
 }
 }
 }
}

```

h1969

Figure 10: Using the set Option

**File contents:**

```

edit access
set profile p1 client cl ike
edit profile p1 client cl ike
set pre-shared-key ascii-text "abcd"
set allowed-proxy-pair local 1.1.1.1 remote 2.2.2.2
exit
deactivate profile p1
top
edit system
set radius-server 1.1.1.1

```

load set

**New contents:**

```

system {
 radius-server {
 1.1.1.1;
 }
}
access {
 inactive: profile p1 {
 client cl {
 ike {
 allowed-proxy-pair local 1.1.1.1/32 remote 2.2.2.2/32;
 pre-shared-key ascii-text "9Ydg4ZDjqf5FVw"; ## SECRET-DATA
 }
 }
 }
}
}

```

g017215

**Related Documentation**

- [Loading a Configuration from a File or the Terminal on page 144](#)

## Creating and Returning to a Rescue Configuration

A rescue configuration allows you to define a known working configuration or a configuration with a known state that you can roll back to at any time. This alleviates the necessity of having to remember the rollback number with the **rollback** command. You use the rescue configuration when you need to roll back to a known configuration or as a last resort if your router or switch configuration and the backup configuration files become damaged beyond repair.

To save the most recently committed configuration as the rescue configuration so that you can return to it at any time, issue the **request system configuration rescue save** command:

```
user@host> request system configuration rescue save
```

To return to the rescue configuration, use the **rollback rescue** configuration mode command:

```
[edit]
user@host# rollback rescue
load complete
```



**NOTE:** If the rescue configuration does not exist, or if the rescue configuration is not a complete, viable configuration, then the **rollback** command fails, an error message appears, and the current configuration remains active.

To activate the rescue configuration that you have loaded, use the **commit** command:

```
[edit]
user@host# rollback rescue
load complete
[edit]
user@host# commit
```

To delete an existing rescue configuration, issue the **request system configuration rescue delete** command:

```
user@host> request system configuration rescue delete
user@host>
```

For more information about the **request system configuration rescue delete** and **request system configuration rescue save** commands, see the [CLI Explorer](#).

### Related Documentation

- [Comparing Configuration Changes with a Prior Version on page 126](#)
- [Saving a Configuration to a File on page 139](#)

## Compressing the Current Configuration File

By default, the current operational configuration file is compressed and is stored in the file **juniper.conf.gz** the **/config** file system, along with the last three committed versions

of the configuration. If you have large networks, the current configuration file might exceed the available space in the **/config** file system. Compressing the current configuration file enables the file to fit in the file system, typically reducing the size of the file by 90 percent. You might want to compress your current operation configuration files when they reach 3 megabytes (MB) in size.

When you compress the current configuration file, the names of the configuration files change. To determine the size of the files in the **/config** file system, issue the **file list /config detail** command.



**NOTE:** We recommend that you compress the configuration files (this is the default) to minimize the amount of disk space that they require.

- If you want to compress the current configuration file, include the **compress-configuration-files** statement at the **[edit system]** hierarchy level:  

```
[edit system]
compress-configuration-files;
```
- Commit the current configuration file to include the **compression-configuration-files** statement. Commit the configuration again to compress the current configuration file:

```
[edit system]
user@host# set compress-configuration-files
user@host# commit
commit complete
user@host# commit
commit complete
```

- If you do not want to compress the current operational configuration file, include the **no-compress-configuration-files** statement at the **[edit system]** hierarchy level:  

```
[edit system]
no-compression-configuration-files;
```
- Commit the current configuration file to include the **no-compress-configuration-files** statement. Commit the configuration again to uncompress the current configuration file:

```
[edit system]
user@host# commit
commit complete
user@host# commit
commit complete
```

**Related  
Documentation**

- [Junos OS Commit Model for Router or Switch Configuration on page 107](#)
- [compress-configuration-files](#)

## Example: Protecting the Junos OS Configuration from Modification or Deletion

---

This example shows how to use the **protect** and **unprotect** commands in the configuration mode to protect and unprotect the CLI configuration.

- [Requirements on page 151](#)
- [Overview on page 151](#)
- [Protecting a Parent-Level Hierarchy on page 152](#)
- [Protecting a Child Hierarchy on page 152](#)
- [Protecting a Configuration Statement Within a Hierarchy on page 152](#)
- [Protecting a List of Identifiers for a Configuration Statement on page 153](#)
- [Protecting an Individual Member from a Homogenous List on page 153](#)
- [Unprotecting a Configuration on page 154](#)
- [Verification on page 154](#)

### Requirements

This example uses the following hardware and software components:

- A M Series, MX Series, PTX Series, or T Series device
- Junos OS 11.2 or later running on all devices

### Overview

The Junos OS enables you to protect the device configuration from being modified or deleted by other users. This can be accomplished by using the **protect** command in the configuration mode of the CLI. Likewise, you can also unprotect a protected configuration by using the **unprotect** command.

These commands can be used at any level of the configuration hierarchy—a top-level parent hierarchy or a configuration statement or an identifier within the lowest level of the hierarchy.

If a configuration hierarchy is protected, users cannot perform the following activities:

- Deleting or modifying a hierarchy or a statement or identifier within the protected hierarchy
- Inserting a new configuration statement or an identifier within the protected hierarchy
- Renaming a statement or identifier within the protected hierarchy
- Copying a configuration into a protected hierarchy
- Activating or deactivating statements within a protected hierarchy
- Annotating a protected hierarchy

## Protecting a Parent-Level Hierarchy

- Step-by-Step Procedure** To protect a configuration at the top level of the hierarchy:
- Identify the hierarchy that you want to protect and issue the **protect** command for the hierarchy at the **[edit]** hierarchy level.
- For example, if you want to protect the entire **[edit access]** hierarchy level, issue the following command:
- ```
[edit]
user@host# protect access
```
- Results** Protects all elements under the parent hierarchy.



NOTE:

- If you issue the **protect** command for a hierarchy that is not used in the configuration, the Junos OS CLI displays the following error message:
- ```
[edit]
user@host# protect access
warning: statement not found
```
- 

## Protecting a Child Hierarchy

- Step-by-Step Procedure** To protect a child hierarchy contained within a parent hierarchy:
- Navigate to the parent container hierarchy. Use the **protect** command for the hierarchy at the parent level.
- For example, if you want to protect the **[edit system syslog console]** hierarchy level, use the following command at the **[edit system syslog]** hierarchy level.
- ```
[edit system syslog]
user@host# protect console
```
- Results** Protects all elements under the child hierarchy.

Protecting a Configuration Statement Within a Hierarchy

- Step-by-Step Procedure** To protect a configuration statement within a hierarchy level:
- Navigate to the hierarchy level containing the statement that you want to protect and issue the **protect** command for the hierarchy.
- For example, if you want to protect the **host-name** statement under the **[edit system]** hierarchy level, issue the following command:
- ```
[edit system]
user@host# protect host-name
```



## Protecting a List of Identifiers for a Configuration Statement

**Step-by-Step Procedure** Some configuration statements can take multiple values. For example, the **address** statement at the **[edit system login deny-sources]** hierarchy level can take a list of hostnames, IPv4 addresses, or IPv6 addresses. Suppose you have the following configuration:

```
[edit system login]
deny-sources {
 address [172.17.28.19 172.17.28.20 172.17.28.21 172.17.28.22];
}
```

- To protect all the addresses for the **address** statement, issue the following command at the **[edit]** level:

```
[edit]
user@host# protect system login deny-sources address
```

**Results** All the addresses ([172.17.28.19 172.17.28.20 172.17.28.21 172.17.28.22]) for the **address** statement are protected.

## Protecting an Individual Member from a Homogenous List

**Step-by-Step Procedure** Suppose you have the following configuration:

```
[edit groups]
test1 {
 system {
 name-server {
 10.1.2.1;
 10.1.2.2;
 10.1.2.3;
 10.1.2.4;
 }
 }
}
```

- To protect one or more individual addresses for the **name-server** statement, issue the following command at the **[edit]** level:

```
[edit]
user@host# protect groups test1 system name-server 10.1.2.1
user@host# protect groups test1 system name-server 10.1.2.4
```

**Results** Addresses 10.1.2.1 and 10.1.2.4 are protected.

## Unprotecting a Configuration

**Step-by-Step Procedure** Suppose you have the following configuration at the **[edit system]** hierarchy level:

```
protect: system {
 host-name bigping;
 domain-search 10.1.2.1;
 login {
 deny-sources {
 protect: address [172.17.28.19 172.17.28.173 172.17.28.0 174.0.0.0];
 }
 }
}
```

- To unprotect the entire **[edit system]** hierarchy level, issue the following command at the **[edit]** level:

```
[edit]
user@host# unprotect system
```

**Results** The entire **system** hierarchy level is unprotected.

## Verification

### Verify That a Hierarchy Is Protected Using the show Command

---

**Purpose** To check that a configuration hierarchy is protected.

**Action** In the configuration mode, issue the **show** command at the **[edit]** hierarchy level to see all the configuration hierarchies and configuration statements that are protected.



**NOTE:** All protected hierarchies or statements are prefixed with a **protect:** string.

---

```
...
protect: system {
 host-name bigping;
 domain-search 10.1.2.1;
 login {
 deny-sources {
 protect: address [172.17.28.19 172.17.28.173 172.17.28.0 174.0.0.0];
 }
 }
}
...
```

### Verify That a Hierarchy Is Protected by Attempting to Modify a Configuration

---

**Purpose** To verify that a configuration is protected by trying to modify the configuration using the **activate**, **copy**, **insert**, **rename**, and **delete** commands.

**Action** To verify that a configuration is protected:

1. Try using the **activate**, **copy**, **insert**, **rename**, and **delete** commands for a top-level hierarchy or a child-level hierarchy or a statement within the hierarchy.

For a protected hierarchy or statement, the Junos OS displays an appropriate warning that the command has not executed. For example:

```
protect: system {
 host-name a;
 inactive: domain-search [a b];
}
```

2. To verify that the hierarchy is protected, try issuing the **activate** command for the **domain-search** statement:

**[edit system]**

```
user@host# activate system domain-search
```

The Junos OS CLI displays an appropriate message:

```
warning: [system] is protected, 'system domain-search' cannot be activated
```

### Verify Usage of the protect Command

**Purpose** To view the **protect** commands used for protecting a configuration.

- Action**
1. Navigate to the required hierarchy.
  2. Issue the **show | display set relative** command.

```
user@host> show | display set relative
set system host-name bigping
set system domain-search 10.1.2.1
set system login deny-sources address 172.17.28.19
set system login deny-sources address 172.17.28.173
set system login deny-sources address 172.17.28.0
set system login deny-sources address 174.0.0.0
protect system login deny-sources address
protect system
```

### View the Configuration in XML

**Purpose** To check if the protected hierarchies or statements are also displayed in the XML. Protected hierarchies, statements, or identifiers are displayed with the **| display xml** attribute in the XML.

**Action** To view the configuration in XML:

1. Navigate to the hierarchy you want to view and issue the **show** command with the pipe symbol and option **| display xml**:

[edit system]

```
user@host# show | display xml
[edit]
user@host# show system | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/11.2IO/junos">
 <configuration junos:changed-seconds="1291279234"
junos:changed-localtime="2010-12-02 00:40:34 PST">
 <system protect="protect">
 <host-name>bigping</host-name>
 <domain-search>10.1.2.1</domain-search>
 <login>
 <message>

 \jnpr

 \tUNAUTHORIZED USE OF THIS ROUTER
 \tIS STRICTLY PROHIBITED!

 </message>
 <class>
 <name>a</name>
 <allow-commands>commit-synchronize</allow-commands>
 <deny-commands>commit</deny-commands>
 </class>
 <deny-sources>
 <address protect="protect">172.17.28.19</address>
 <address protect="protect">172.17.28.173</address>
 <address protect="protect">172.17.28.0</address>
 <address protect="protect">174.0.0.0</address>
 </deny-sources>
 </login>
 <syslog>
 <archive>
 </archive>
 </syslog>
 </system>
 </configuration>
 <cli>
 <banner>[edit]</banner>
 </cli>
</rpc-reply>
```



**NOTE:** Loading an XML configuration with the `unprotect="unprotect"` tag unprotects an already protected hierarchy. For example, suppose you load the following XML hierarchy:

```
<protocols unprotect="unprotect">
 <ospf>
 <area>
 <name>0.0.0.0</name>
 <interface>
 <name>all</name>
 </interface>
```

```
 </area>
 </ospf>
</protocols>
```

The `[edit protocols]` hierarchy becomes unprotected if it is already protected.

---

## Synchronizing Routing Engines

---

If your router has two Routing Engines, you can manually direct one Routing Engine to synchronize its configuration with the other by issuing the **commit synchronize** command. The Routing Engine on which you execute this command (requesting Routing Engine) copies and loads its candidate configuration to the other (responding Routing Engine). Both Routing Engines then perform a syntax check on the candidate configuration file being committed. If no errors are found, the configuration is activated and becomes the current operational configuration on both Routing Engines.

The **commit synchronize** command does not work if the responding Routing Engine has uncommitted configuration changes. However, you can enforce commit synchronization on the Routing Engines by using the **force** option. When you issue the **commit synchronize** command with the **force** option from one Routing Engine, the configuration sessions on the other Routing Engine will be terminated and its configuration synchronized with that on the Routing Engine from which you issued the command.



**NOTE:** We recommend that you use the **force** option only if you are unable to resolve the issues that caused the **commit synchronize** command to fail.

---

For example, if you are logged in to **re1** (requesting Routing Engine) and you want **re0** (responding Routing Engine) to have the same configuration as **re1**, issue the **commit synchronize** command on **re1**. **re1** copies and loads its candidate configuration to **re0**. Both Routing Engines then perform a syntax check on the candidate configuration file being committed. If no errors are found, **re1**'s candidate configuration is activated and becomes the current operational configuration on both Routing Engines.



**NOTE:** When you issue the **commit synchronize** command, you must use the groups **re0** and **re1**. For information about how to use the **apply-groups** statement, see [“Applying the Junos OS Configuration Group” on page 219](#).

The responding Routing Engine must be running Junos OS Release 5.0 or later.

For information about issuing the **commit synchronize** command on a routing matrix, see the *Junos OS Administration Library*.

---

To synchronize a Routing Engine's current operational configuration file with the other, log in to the Routing Engine from which you want to synchronize and issue the **commit synchronize** command:

```
[edit]
user@host# commit synchronize
re0:
configuration check succeeds
re1:
commit complete
re0:
commit complete
```



**NOTE:** You can also add the **commit synchronize** statement at the **[edit system]** hierarchy level so that a **commit** command automatically invokes a **commit synchronize** command by default. For more information, see the *Junos OS Administration Library*.

To enforce a **commit synchronize** on the Routing Engines, log in to the Routing Engine from which you want to synchronize and issue the **commit synchronize** command with the **force** option:

```
[edit]
user@host# commit synchronize force
re0:
re1:
commit complete
re0:
commit complete
[edit]
user@host#
```



**NOTE:**

- If you have nonstop routing enabled on your router, you must enter the **commit synchronize** command from the master Routing Engine after you make any changes to the configuration. If you enter this command on the backup Routing Engine, the Junos OS displays a warning and commits the configuration.
- Starting with Junos OS Release 9.3, accounting of backup Routing Engine events or operations is not supported on accounting servers such as TACACS+ or RADIUS. Accounting is only supported for events or operations on a master Routing Engine.

For the **commit** synchronization process, the master Routing Engine commits the configuration and sends a copy of the configuration to the backup Routing Engine. Then the backup Routing Engine loads and commits the configuration. So, the **commit** synchronization between the master and backup Routing Engines takes place one Routing

Engine at a time. If the configuration has a large text size or many apply-groups, commit times can be longer than desired.

You can use the **commit fast-synchronize** statement to have the synchronization between the master and backup Routing Engines occur simultaneously instead of sequentially. This can reduce the time needed for synchronization because the commits on the master and backup Routing Engines occur in parallel.

Include the **fast-synchronize** statement at the **[edit system]** hierarchy level to have the synchronization occur simultaneously between the master and the backup Routing Engines:

```
[edit system]
commit fast-synchronize;
```



NOTE:

- When the **fast-synchronize** statement is configured, the commits on the master Routing Engine and the backup Routing Engine run in parallel. In this process, the configuration is validated only on the Routing Engine where you execute the **commit** command. Therefore, it is recommended not to include too many configuration details in groups like **re0** and **re1**, because the configuration specified in group **re0** is applied only if the current Routing Engine is in slot 0. Likewise, the configuration specified in group **re1** is applied only if the current Routing Engine is in slot 1.
- Ensure that the Junos OS software version running on both the Routing Engines is same.

You can use the **commit synchronize scripts** command to synchronize a Routing Engine's configuration and all commit, event, lib, op, and SNMP scripts with the other Routing Engine. If the **load-scripts-from-flash** statement is configured for the requesting Routing Engine, the device synchronizes the scripts from flash memory on the requesting Routing Engine to flash memory on the responding Routing Engine. Otherwise, the device synchronizes the scripts from the hard disk on the requesting Routing Engine to the hard disk on the responding Routing Engine. The device synchronizes all scripts regardless of whether they are enabled in the configuration or have been updated since the last synchronization.

To synchronize a Routing Engine's configuration file and all scripts with the other Routing Engine, log in to the Routing Engine from which you want to synchronize, and issue the **commit synchronize scripts** command:

```
[edit]
user@host# commit synchronize scripts
re0:
configuration check succeeds
re1:
commit complete
re0:
commit complete
```



If the commit check operation fails for the requesting Routing Engine, the process stops, and the scripts are not copied to the responding Routing Engine. If the commit check or commit operation fails for the responding Routing Engine, the scripts are still synchronized, since the synchronization occurs prior to the commit check operation on the responding Routing Engine.

Include the **synchronize** statement at the **[edit system scripts]** hierarchy level to synchronize scripts every time you issue a **commit synchronize** command.

```
[edit system scripts]
synchronize;
```



#### NOTE:

- If commit fails on either Routing Engine, the commit process is rolled back on the other Routing Engine as well. This ensures that both Routing Engines have the same configuration.
- When the **fast-synchronize** statement is configured, the commits on the master Routing Engine and the backup Routing Engine run in parallel. In this process, the configuration is validated only on the Routing Engine where you execute the **commit** command. Therefore, it is recommended not to include too many configuration details in groups like **re0** and **re1**, because the configuration specified in group **re0** is applied only if the current Routing Engine is in slot 0. Likewise, the configuration specified in group **re1** is applied only if the current Routing Engine is in slot 1.
- Ensure that the Junos OS software version running on both the Routing Engines is same.

#### Related Documentation

- *Configuring the Junos OS to Support Redundancy on Routers Having Multiple Routing Engines or Switching Boards*
- *Junos OS Routing Engine Components and Processes*
- *Configuring Junos OS for the First Time on a Device with Dual Routing Engines*

## Configuring Multiple Routing Engines to Synchronize Committed Configurations Automatically

If your router or switch has multiple Routing Engines, you can manually direct one Routing Engine to synchronize its configuration with the others by issuing the **commit synchronize** command.

To make the Routing Engines synchronize automatically whenever a configuration is committed, include the **commit synchronize** statement at the **[edit system]** hierarchy level:

```
[edit system]
commit synchronize;
```

The Routing Engine on which you execute the **commit** command (requesting Routing Engine) copies and loads its candidate configuration to the other (responding) Routing Engines. All Routing Engines then perform a syntax check on the candidate configuration file being committed. If no errors are found, the configuration is activated and becomes the current operational configuration on all Routing Engines.

For the commit synchronization process, the master Routing Engine commits the configuration and sends a copy of the configuration to the backup Routing Engine. Then the backup Routing Engine loads and commits the configuration. So, the commit synchronization between the master and backup Routing Engines takes place one Routing Engine at a time. If the configuration has a large text size or many apply-groups, commit times can be longer than desired.

You can use the **commit fast-synchronize** statement to have the synchronization between the master and backup Routing Engines occur simultaneously instead of sequentially. This can reduce the time needed for synchronization because the commits on the master and backup Routing Engines occur in parallel.

Include the **fast-synchronize** statement at the **[edit system]** hierarchy level to have synchronize occur simultaneously between the master and the backup Routing Engines:

```
[edit system]
commit fast-synchronize
```



NOTE:

- If commit fails on either Routing Engine, the commit process is rolled back on the other Routing Engine as well. This ensures that both Routing Engines have the same configuration.
- When the fast-synchronize statement is configured, the commits on the master Routing Engine and the backup Routing Engine run in parallel. In this process, the configuration is validated only on the Routing Engine where you execute the commit command. Therefore, it is recommended not to include too many configuration details in groups like re0 and re1, because the configuration specified in group re0 is applied only if the current Routing Engine is in slot 0. Likewise, the configuration specified in group re1 is applied only if the current Routing Engine is in slot 1.
- Ensure that the Junos OS software version running on both the Routing Engines is same.

**Related  
Documentation**

- [Junos OS Commit Model for Router or Switch Configuration on page 107](#)

## CHAPTER 7

# Using Operational Commands to Monitor a Device

- [Overview of Junos OS CLI Operational Mode Commands on page 163](#)
- [Junos OS Operational Mode Commands That Combine Other Commands on page 166](#)
- [Understanding the Brief, Detail, Extensive, and Terse Options of Junos OS Operational Commands on page 167](#)
- [Controlling the Scope of an Operational Mode Command on page 168](#)
- [Monitoring Who Uses the Junos OS CLI on page 171](#)
- [Interface Naming Conventions Used in the Junos OS Operational Commands on page 172](#)
- [Viewing Files and Directories on a Device Running Junos OS on page 173](#)
- [Displaying Junos OS Information on page 177](#)
- [Managing Programs and Processes Using Junos OS Operational Mode Commands on page 179](#)
- [Using the Junos OS CLI Comment Character # for Operational Mode Commands on page 184](#)
- [Example: Using Comments in Junos OS Operational Mode Commands on page 184](#)

## Overview of Junos OS CLI Operational Mode Commands

---

This topic provides an overview of Junos OS CLI operational mode commands and contains the following sections:

- [CLI Command Categories on page 163](#)
- [Commonly Used Operational Mode Commands on page 165](#)

### CLI Command Categories

When you log in to a device running Junos OS and the CLI starts, there are several broad groups of CLI commands:

- Commands for controlling the CLI environment—Some set commands in the **set** hierarchy configure the CLI display screen. For information about these commands, see [“Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies” on page 23](#).
- Commands for monitoring and troubleshooting—The following commands display information and statistics about the software and test network connectivity. Detailed command descriptions are provided in the *Junos OS Interfaces Command Reference*.
  - **clear**—Clear statistics and protocol database information.
  - **mtrace**—Trace mtrace packets from source to receiver.
  - **monitor**—Perform real-time debugging of various software components, including the routing protocols and interfaces.
  - **ping**—Determine the reachability of a remote network host.
  - **show**—Display the current configuration and information about interfaces, routing protocols, routing tables, routing policy filters, system alarms, and the chassis.
  - **test**—Test the configuration and application of policy filters and autonomous system (AS) path regular expressions.
  - **traceroute**—Trace the route to a remote network host.
- Commands for connecting to other network systems—The **ssh** command opens Secure Shell connections, and the **telnet** command opens telnet sessions to other hosts on the network. For information about these commands, see the [CLI Explorer](#).
- Commands for copying files—The **copy** command copies files from one location on the router or switch to another, from the router or switch to a remote system, or from a remote system to the router or switch. For information about these commands, see the [CLI Explorer](#).
- Commands for restarting software processes—The commands in the **restart** hierarchy restart the various Junos OS processes, including the routing protocol, interface, and SNMP. For information about these commands, see the [CLI Explorer](#).
- A command—**request**—for performing system-level operations, including stopping and rebooting the router or switch and loading Junos OS images. For information about this command, see the [CLI Explorer](#).
- A command—**start**—to exit the CLI and start a UNIX shell. For information about this command, see the [CLI Explorer](#).
- A command—**configure**—for entering configuration mode, which provides a series of commands that configure Junos OS, including the routing protocols, interfaces, network management, and user access. For information about the CLI configuration commands, see [“Understanding Junos OS CLI Configuration Mode” on page 56](#).
- A command—**quit**—to exit the CLI. For information about this command, see the [CLI Explorer](#).
- For more information about the CLI operational mode commands, see the [CLI Explorer](#).

## Commonly Used Operational Mode Commands

Table 8 on page 165 lists some operational commands you may find useful for monitoring router or switch operation. For a complete description of operational commands, see the Junos OS command references.



**NOTE:** The QFX3500 switch does not support the IS-IS, OSPF, BGP, MPLS, and RSVP protocols.

**Table 8: Commonly Used Operational Mode Commands**

Items to Check	Description	Command
Software version	Versions of software running on the router or switch	<b>show version</b>
Log files	Contents of the log files	<b>monitor</b>
	Log files and their contents and recent user logins	<b>show log</b>
Remote systems	Host reachability and network connectivity	<b>ping</b>
	Route to a network system	<b>traceroute</b>
Configuration	Current system configuration	<b>show configuration</b>
Manipulate files	List of files and directories on the router or switch	<b>file list</b>
	Contents of a file	<b>file show</b>
Interface information	Detailed information about interfaces	<b>show interfaces</b>
Chassis	Chassis alarm status	<b>show chassis alarms</b>
	Information currently on craft display	<b>show chassis craft-interface</b>
	Router or switch environment information	<b>show chassis environment</b>
	Hardware inventory	<b>show chassis hardware</b>
Routing table information	Information about entries in the routing tables	<b>show route</b>
Forwarding table information	Information about data in the kernel's forwarding table	<b>show route forwarding-table</b>
IS-IS	Adjacent routers or switches	<b>show isis adjacency</b>
OSPF	Display standard information about OSPF neighbors	<b>show ospf neighbor</b>
BGP	Display information about BGP neighbors	<b>show bgp neighbor</b>

Table 8: Commonly Used Operational Mode Commands (*continued*)

Items to Check	Description	Command
MPLS	Status of interfaces on which MPLS is running	<b>show mpls interface</b>
	Configured LSPs on the router or switch, as well as all ingress, transit, and egress LSPs	<b>show mpls lsp</b>
	Routes that form a label-switched path	<b>show route label-switched-path</b>
RSVP	Status of interfaces on which RSVP is running	<b>show rsvp interface</b>
	Currently active RSVP sessions	<b>show rsvp session</b>
	RSVP packet and error counters	<b>show rsvp statistics</b>

**Related  
Documentation**

- [Junos OS Operational Mode Commands That Combine Other Commands on page 166](#)
- [Understanding the Brief, Detail, Extensive, and Terse Options of Junos OS Operational Commands on page 167](#)

## Junos OS Operational Mode Commands That Combine Other Commands

In some cases, some Junos OS operational commands are created from a combination of other operational commands. These commands can be useful shortcuts for collecting information about the device, as shown in [Figure 11 on page 167](#).

Figure 11: Commands That Combine Other Commands

The **request support information** command provides output from a combination of other operational commands.

```

user@host> request support information

root@host> show system uptime

Current time: 2007-02-16 13:10:08 PST
System booted: 2007-02-02 09:21:50 PST (2w0d 03:48 ago)
Protocols started: 2007-02-02 09:24:42 PST (2w0d 03:45 ago)
Last configured: 2007-02-16 03:04:58 PST (10:05:10 ago) by root
1:10PM up 14 days, 3:48, 2 users, load averages: 0.01, 0.02, 0.00

root@host> show version detail

Hostname: host
Model: m320
JUNOS Base OS boot [8.3-R1.1]

root@host> show system core-dumps

/var/tmp/*core*: No such file or directory
/var/crash/kernel.*: No such file or directory

/var/crash/cores:
total 9780
-rw-r--r-- 1 root wheel 4990976 Feb 9 15:39
core-FPC2.core.0.060209.1539

root@host> show chassis hardware detail

Hardware inventory:
Item Version Part number Serial number Description
Chassis
Backplane REV 07 710-001517 AW44 31 M20 Backplane
Power Supply B REV 09 740-001466 0042 33 DC Power Supply

```

#### Related Documentation

- [Overview of Junos OS CLI Operational Mode Commands on page 163](#)
- [Understanding the Brief, Detail, Extensive, and Terse Options of Junos OS Operational Commands on page 167](#)

## Understanding the Brief, Detail, Extensive, and Terse Options of Junos OS Operational Commands

The Junos OS operational mode commands can include **brief**, **detail**, **extensive**, or **terse** options. You can use these options to control the amount of information you want to view.

1. Use the **?** prompt to list options available for the command. For example:

```

user@host> show interfaces fe-1/1/1 ?
Possible completions:
<[Enter]> Execute this command
brief Display brief output
descriptions Display interface description strings
detail Display detailed output
extensive Display extensive output
media Display media information
snmp-index SNMP index of interface
statistics Display statistics and detailed output
terse Display terse output
| Pipe through a command

```

2. Choose the option you wish to use with the command. (See [Figure 12 on page 168](#).)

Figure 12: Command Output Options

Command output with the **brief** option.

```

user@host> show interfaces fe-1/1/1 brief
Physical interface: fe-1/1/1, Enabled, Physical link is Down
Link-level type: Ethernet, MTU: 1514, Speed: 100mbps, Loopback:
Disabled, Source filtering: Disabled,
Flow control: Enabled
Device flags : Present Running Down
Interface flags: Hardware-Down SNMP-Traps Internal: 0x4000
Link flags : None

```

Command output with the **terse** option.

```

user@host> show interfaces fe-1/1/1 terse
Interface Admin Link Proto Local Remote
fe-1/1/1 up down

```

Command output with the **extensive** option.

```

user@host> show interfaces fe-1/1/1 extensive
Physical interface: fe-1/1/1, Enabled, Physical link is Down
Interface index: 141, SNMP ifIndex: 33, Generation: 24
Link-level type: Ethernet, MTU: 1514, Speed: 100mbps, Loopback:
Disabled, Source filtering: Disabled,
Flow control: Enabled
Device flags : Present Running Down
Interface flags: Hardware-Down SNMP-Traps Internal: 0x4000
Link flags : None
CoS queues : 4 supported, 4 maximum usable queues
Hold-times : Up 0 ms, Down 0 ms
Current address: 00:90:69:d0:f8:9e, Hardware address: 00:90:69:d0:f8:9e
Last flapped : 2007-02-02 09:26:25 PST (2w0d 03:40 ago)
Statistics last cleared: Never
Traffic statistics:
Input bytes : 0 0 bps
Output bytes : 0 0 bps
Input packets: 0 0 pps
Output packets: 0 0 pps
--(more)--

```

#### Related Documentation

- [Overview of Junos OS CLI Operational Mode Commands on page 163](#)
- [Controlling the Scope of an Operational Mode Command on page 168](#)

## Controlling the Scope of an Operational Mode Command

The Junos OS CLI operational commands include options that you can use to identify specific components on a device running Junos OS. For example:

1. Type the **show interfaces** command to display information about all interfaces on the router.

```

user@host> show interfaces
Physical interface: so-0/0/0, Enabled, Physical link is Up
Interface index: 128, SNMP ifIndex: 23
Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode, Speed: OC3,
Loopback: None, FCS: 16, Payload scrambler: Enabled
Device flags : Present Running
Interface flags: Point-To-Point SNMP-Traps Internal: 0x4000
Link flags : Keepalives
Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
Keepalive: Input: 13861 (00:00:05 ago), Output: 13891 (00:00:01 ago)
LCP state: Opened
NCP state: inet: Opened, inet6: Not-configured, iso: Opened, mppls:
Not-configured
CHAP state: Closed
PAP state: Closed

```



```

CoS queues : 4 supported, 4 maximum usable queues
Last flapped : 2008-06-02 17:16:14 PDT (1d 14:21 ago)
Input rate : 40 bps (0 pps)
Output rate : 48 bps (0 pps)

```

---(more)---

2. To display information about a specific interface, type that interface as a command option:

```

user@host> show interfaces fe-0/1/3
Physical interface: fe-0/1/3, Enabled, Physical link is Up
 Interface index: 135, SNMP ifIndex: 30
 Link-level type: Ethernet, MTU: 1514, Speed: 100mbps, MAC-REWRITE Error:
None,
 Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled
 Device flags : Present Running
 Interface flags: SNMP-Traps Internal: 0x4000
 Link flags : None
 CoS queues : 4 supported, 4 maximum usable queues
 Current address: 00:05:85:8f:c8:22, Hardware address: 00:05:85:8f:c8:22
 Last flapped : 2008-06-02 17:16:15 PDT (1d 14:28 ago)
 Input rate : 0 bps (0 pps)
 Output rate : 0 bps (0 pps)
 Active alarms : None
 Active defects : None

user@host>

```

## Operational Mode Commands on a TX Matrix Router or TX Matrix Plus Router

When you issue operational mode commands on the TX Matrix router, CLI command options allow you to restrict the command output to show only a component of the routing matrix rather than the routing matrix as a whole.

These are the options shown in the CLI:

- **scc**—The TX Matrix router (or switch-card chassis)
- **sfc**—The TX Matrix Plus router (also referred to as or switch-fabric chassis)
- **lcc number**—A specific router in a routing matrix based on a TX Matrix router or a TX Matrix Plus router.
- **all-lcc**—All T640 routers (in a routing matrix based on a TX Matrix router) or all T1600 routers or T4000 routers (in a routing matrix based on a TX Matrix Plus router).

If you specify none of these options, then the command applies by default to the whole routing matrix.

## Examples of Routing Matrix Command Options

The following output samples, using the **show version** command, demonstrate some different options for viewing information about the routing matrix.

```

user@host> show version ?
Possible completions:
 <[Enter]> Execute this command
 all-lcc Show software version on all LCC chassis

```

brief	Display brief output
detail	Display detailed output
lcc	Show software version on specific LCC (0..3)
scc	Show software version on the SCC
	Pipe through a command

### Sample Output: No Routing Matrix Options Specified

```
user@host> show version
scc-re0:

Hostname: scc
Model: TX Matrix
JUNOS Base OS boot [7.0-20040630.0]
JUNOS Base OS Software Suite [7.0-20040629.0]
JUNOS Kernel Software Suite [7.0-20040630.0]
JUNOS Packet Forwarding Engine Support (T-Series) [7.0-20040630.0]
JUNOS Routing Software Suite [7.0-20040630.0]
JUNOS Online Documentation [7.0-20040630.0]
JUNOS Crypto Software Suite [7.0-20040630.0]
lcc0-re0:

Hostname: lcc0
Model: t640
JUNOS Base OS boot [7.0-20040630.0]
JUNOS Base OS Software Suite [7.0-20040629.0]
JUNOS Kernel Software Suite [7.0-20040630.0]
JUNOS Packet Forwarding Engine Support (T-Series) [7.0-20040630.0]
JUNOS Routing Software Suite [7.0-20040630.0]
JUNOS Online Documentation [7.0-20040630.0]
JUNOS Crypto Software Suite [7.0-20040630.0]
JUNOS Support Tools Package [7.0-20040630.0]
lcc1-re0:

Hostname: lcc1
Model: t640
JUNOS Base OS boot [7.0-20040630.0]
JUNOS Base OS Software Suite [7.0-20040629.0]
JUNOS Kernel Software Suite [7.0-20040630.0]
JUNOS Packet Forwarding Engine Support (T-Series) [7.0-20040630.0]
JUNOS Routing Software Suite [7.0-20040630.0]
JUNOS Online Documentation [7.0-20040630.0]
JUNOS Crypto Software Suite [7.0-20040630.0]
JUNOS Support Tools Package [7.0-20040630.0]
```

### Sample Output: TX Matrix Router Only (scc Option)

```
user@host> show version scc
Hostname: scc
Model: TX Matrix
JUNOS Base OS boot [7.0-20040630.0]
JUNOS Base OS Software Suite [7.0-20040629.0]
JUNOS Kernel Software Suite [7.0-20040630.0]
JUNOS Packet Forwarding Engine Support (T-Series) [7.0-20040630.0]
JUNOS Routing Software Suite [7.0-20040630.0]
JUNOS Online Documentation [7.0-20040630.0]
JUNOS Crypto Software Suite [7.0-20040630.0]
```

### Sample Output: Specific T640 Router (lcc number Option)

```
user@host> show version lcc 0
```

```
lcc0-re0:
```

```

Hostname: lcc0
Model: t640
JUNOS Base OS boot [7.0-20040630.0]
JUNOS Base OS Software Suite [7.0-20040629.0]
JUNOS Kernel Software Suite [7.0-20040630.0]
JUNOS Packet Forwarding Engine Support (T-Series) [7.0-20040630.0]
JUNOS Routing Software Suite [7.0-20040630.0]
JUNOS Online Documentation [7.0-20040630.0]
JUNOS Crypto Software Suite [7.0-20040630.0]
JUNOS Support Tools Package [7.0-20040630.0]
```

### Sample Output: All T640 Routers (all-lcc Option)

```
user@host> show version all-lcc
lcc0-re0:
```

```

Hostname: lcc0
Model: t640
JUNOS Base OS boot [7.0-20040630.0]
JUNOS Base OS Software Suite [7.0-20040629.0]
JUNOS Kernel Software Suite [7.0-20040630.0]
JUNOS Packet Forwarding Engine Support (T-Series) [7.0-20040630.0]
JUNOS Routing Software Suite [7.0-20040630.0]
JUNOS Online Documentation [7.0-20040630.0]
JUNOS Crypto Software Suite [7.0-20040630.0]
JUNOS Support Tools Package [7.0-20040630.0]
lcc1-re0:
```

```

Hostname: lcc1
Model: t640
JUNOS Base OS boot [7.0-20040630.0]
JUNOS Base OS Software Suite [7.0-20040629.0]
JUNOS Kernel Software Suite [7.0-20040630.0]
JUNOS Packet Forwarding Engine Support (T-Series) [7.0-20040630.0]
JUNOS Routing Software Suite [7.0-20040630.0]
JUNOS Online Documentation [7.0-20040630.0]
JUNOS Crypto Software Suite [7.0-20040630.0]
JUNOS Support Tools Package [7.0-20040630.0]
```

- Related Documentation**
- [Interface Naming Conventions Used in the Junos OS Operational Commands on page 172](#)
  - [Using the Junos OS CLI Comment Character # for Operational Mode Commands on page 184](#)

## Monitoring Who Uses the Junos OS CLI

Depending upon how you configure Junos OS, multiple users can log in to the router, use the CLI, and configure or modify the software configuration.

If, when you enter configuration mode, another user is also in configuration mode, a notification message is displayed that indicates who the user is and what portion of the configuration the person is viewing or editing:

```
user@host> configure
Entering configuration mode
```

```
Users currently editing the configuration:
 root terminal d0 (pid 4137) on since 2008-04-09 23:03:07 PDT, idle 7w6d 08:22
 [edit]
The configuration has been changed but not committed

[edit]
user@host#
```

**Related  
Documentation**

- [Entering and Exiting the Junos OS CLI Configuration Mode on page 62](#)
- [Controlling the Junos OS CLI Environment on page 247](#)

---

## Interface Naming Conventions Used in the Junos OS Operational Commands

This topic explains the interface naming conventions used in the Junos OS operational commands, and contains the following sections:

- [Physical Part of an Interface Name on page 172](#)
- [Logical Part of an Interface Name on page 172](#)
- [Channel Identifier Part of an Interface Name on page 173](#)

### Physical Part of an Interface Name

The physical interface naming conventions for Junos OS platforms is as follows:

- On SRX devices, the unique name of each network interface has the following format to identify the physical device that corresponds to a single physical network connector:

*type-slot/pim-or-ioc/port*

- On other platforms, when you display information about an interface, you specify the interface type, the slot in which the Flexible PIC Concentrator (FPC) is installed, the slot on the FPC in which the PIC is located, and the configured port number.

In the physical part of the interface name, a hyphen (-) separates the media type from the FPC number, and a slash (/) separates the FPC, PIC, and port numbers:

*type-fpc/pic/port*



**NOTE:** Exceptions to the *type-fpc/pic/port* physical description include the aggregated Ethernet and aggregated SONET/SDH interfaces, which use the syntax *aenumber* and *asnumber*, respectively.

---

### Logical Part of an Interface Name

The logical unit part of the interface name corresponds to the logical unit number, which can be a number from 0 through 16,384. In the virtual part of the name, a period (.) separates the port and logical unit numbers:

- SRX devices:

*type-slot/pim-or-ioc/port:channel.unit*

- Other platforms:

*type-fpc/pic/port.logical*

## Channel Identifier Part of an Interface Name

The channel identifier part of the interface name is required only on channelized interfaces. For channelized interfaces, channel 0 identifies the first channelized interface. For channelized intelligent queuing (IQ) interfaces, channel 1 identifies the first channelized interface.



**NOTE:** Depending on the type of channelized interface, up to three levels of channelization can be specified. For more information, see the *Junos Network Interfaces Configuration Guide*.

A colon (:) separates the physical and virtual parts of the interface name:

- SRX devices:

*type-slot/pim-or-ioc/port:channel*  
*type-slot/pim-or-ioc/port:channel:channel*  
*type-slot/pim-or-ioc/port:channel:channel:channel*

- Other platforms:

*type-fpc/pic/port:channel*  
*type-fpc//pic/port:channel:channel*  
*type-fpc/pic/port:channel:channel:channel*

### Related Documentation

- [Example: Configuring Interfaces Using Junos OS Configuration Groups on page 228](#)
- [Junos OS Network Interfaces Library for Routing Devices](#)

## Viewing Files and Directories on a Device Running Junos OS

Junos OS stores information in files on the device, including configuration files, log files, and router software files. This topic shows some examples of operational commands that you can use to view files and directories on a device running Junos OS.

Sections include:

- [Directories on the Router or Switch on page 173](#)
- [Listing Files and Directories on page 174](#)
- [Specifying Filenames and URLs on page 176](#)

### Directories on the Router or Switch

[Table 9 on page 174](#) lists some standard directories on a device running Junos OS.

Table 9: Directories on the Router

Directory	Description
<code>/config</code>	This directory is located on the device's router's internal flash drive. It contains the active configuration ( <b>juniper.conf</b> ) and rollback files 1, 2, and 3.
<code>/var/db/config</code>	This directory is located on the router's device's hard drive and contains rollback files 4 through 49.
<code>/var/tmp</code>	This directory is located on the device's hard drive. It holds core files from the various processes on the Routing Engines. Core files are generated when a particular process crashes and are used by Juniper Networks engineers to diagnose the reason for failure.
<code>/var/log</code>	This directory is located on the device's hard drive. It contains files generated by both the device's logging function as well as the <b>traceoptions</b> command.
<code>/var/home</code>	This directory is located on the device's hard drive. It contains a subdirectory for each configured user on the device. These individual user directories are the default file location for many Junos OS commands.
<code>/altroot</code>	This directory is located on the device's hard drive and contains a copy of the root file structure from the internal flash drive. This directory is used in certain disaster recovery modes where the internal flash drive is not operational.
<code>/altconfig</code>	This directory is located on the device's hard drive and contains a copy of the <code>/config</code> file structure from the internal flash drive. This directory is also used in certain disaster recovery modes when the internal flash drive is not operational.

## Listing Files and Directories

You can view the device's directory structure as well as individual files by issuing the **file** command in operational mode.

- To get help about the **file** command, type the following:

```

user@host> file ?
Possible completions:
<[Enter]> Execute this command
archive Archives files from the system
checksum Calculate file checksum
compare Compare files
copy Copy files (local or remote)
delete Delete files from the system
list List file information
rename Rename files
show Show file contents
source-address Local address to use in originating the connection
| Pipe through a command
user@host> file

```

Help shows that the **file** command includes several options for manipulating files.

2. Use the **list** option to see the directory structure of the device. For example, to show the files located in your home directory on the device:

```
user@host> file list
.ssh/
common
```

The default directory for the **file list** command is the home directory of the user logged in to the device. In fact, the user's home directory is the default directory for most of Junos OS commands requiring a filename.

3. To view the contents of other file directories, specify the directory location. For example:

```
user@host> file list /config
juniper.conf
juniper.conf.1.gz
juniper.conf.2.gz
juniper.conf.3.gz
```

4. You can also use the device's context-sensitive help system to locate a directory. For example:

```
user@host> file list /?
Possible completions:
<[Enter]> Execute this command
<path> Path to list
/COPYRIGHT Size: 6355, Last changed: Feb 13 2005
/altconfig/ Last changed: Aug 07 2007
/altroot/ Last changed: Aug 07 2007
/bin/ Last changed: Apr 09 22:31:35
/boot/ Last changed: Apr 09 23:28:39
/config/ Last changed: Apr 16 22:35:35
/data/ Last changed: Aug 07 2007
/dev/ Last changed: Apr 09 22:36:21
/etc/ Last changed: Apr 11 03:14:22
/kernel Size: 27823246, Last changed: Aug 07 2007
/mfs/ Last changed: Apr 09 22:36:49
/mnt/ Last changed: Jan 11 2007
/modules/ Last changed: Apr 09 22:33:54
/opt/ Last changed: Apr 09 22:31:00
/packages/ Last changed: Apr 09 22:34:38
/proc/ Last changed: May 07 20:25:46
/rdm.taf Size: 498, Last changed: Apr 09 22:37:31
/root/ Last changed: Apr 10 02:19:45
/sbin/ Last changed: Apr 09 22:33:55
/staging/ Last changed: Apr 09 23:28:41
/tmp/ Last changed: Apr 11 03:14:49
/usr/ Last changed: Apr 09 22:31:34
/var/ Last changed: Apr 09 22:37:30
user@host> file list /var/?
<[Enter]> Execute this command
<path> Path to list
/var/account/ Last changed: Jul 09 2007
/var/at/ Last changed: Jul 09 2007
/var/backups/ Last changed: Jul 09 2007
/var/bin/ Last changed: Jul 09 2007
/var/crash/ Last changed: Apr 09 22:31:08
/var/cron/ Last changed: Jul 09 2007
```

```
/var/db/ Last changed: May 07 20:28:40
/var/empty/ Last changed: Jul 09 2007
/var/etc/ Last changed: Apr 16 22:35:36
/var/heimdal/ Last changed: Jul 10 2007
/var/home/ Last changed: Apr 09 22:59:18
/var/jail/ Last changed: Oct 31 2007
/var/log/ Last changed: Apr 17 02:00:10
/var/mail/ Last changed: Jul 09 2007
/var/messages/ Last changed: Jul 09 2007
/var/named/ Last changed: Jul 10 2007
/var/packages/ Last changed: Jan 18 02:38:59
/var/pdb/ Last changed: Oct 31 2007
/var/preserve/ Last changed: Jul 09 2007
/var/run/ Last changed: Apr 17 02:00:01
/var/rundb/ Last changed: Apr 17 00:46:00
/var/rwho/ Last changed: Jul 09 2007
/var/sdb/ Last changed: Apr 09 22:37:31
/var/spool/ Last changed: Jul 09 2007
/var/sw/ Last changed: Jul 09 2007
/var/tmp/ Last changed: Apr 09 23:28:41
/var/transfer/ Last changed: Jul 09 2007
/var/yp/ Last changed: Jul 09 2007
user@host> file list /var/
```

5. You can also display the contents of a file. For example:

```
user@host>file show /var/log/inventory
Jul 9 23:17:46 CHASSISD release 8.4I0 built by builder on 2007-06-12 07:58:27
UTC
Jul 9 23:18:05 CHASSISD release 8.4I0 built by builder on 2007-06-12 07:58:27
UTC
Jul 9 23:18:06 Routing Engine 0 - part number 740-003239, serial number
9000016755
Jul 9 23:18:15 Routing Engine 1 - part number 740-003239, serial number
9001018324
Jul 9 23:19:03 SSB 0 - part number 710-001951, serial number AZ8025
Jul 9 23:19:03 SSRAM bank 0 - part number 710-001385, serial number 243071
Jul 9 23:19:03 SSRAM bank 1 - part number 710-001385, serial number 410608
...
```

## Specifying Filenames and URLs

In some CLI commands and configuration statements—including **file copy**, **file archive**, **load**, **save**, **set system login user *username* authentication *load-key-file***, and **request system software add**—you can include a filename. On a routing matrix, you can include chassis information as part of the filename (for example, **lcc0**, **lcc0-re0**, or **lcc0-re1**).



You can specify a filename or URL in one of the following ways:

- **filename**—File in the user's current directory on the local flash drive. You can use wildcards to specify multiple source files or a single destination file. Wildcards are not supported in Hypertext Transfer Protocol (HTTP) or FTP.



**NOTE:** Wildcards are supported only by the **file** (**compare** | **copy** | **delete** | **list** | **rename** | **show**) commands. When you issue the **file show** command with a wildcard, it must resolve to one filename.

- **path/filename**—File on the local flash disk.
- **/var/filename** or **/var/path/filename**—File on the local hard disk. You can also specify a file on a local Routing Engine for a specific T640 router on a routing matrix:  

```
user@host> file delete lcc0-re0:/var/tmp/junk
```
- **a:filename** or **a:path/filename**—File on the local drive. The default path is / (the root-level directory). The removable media can be in MS-DOS or UNIX (UFS) format.
- **hostname:/path/filename**, **hostname:filename**, **hostname:path/filename**, or **scp://hostname/path/filename**—File on an scp/ssh client. This form is not available in the worldwide version of Junos OS. The default path is the user's home directory on the remote system. You can also specify **hostname** as **username@hostname**.
- **ftp://hostname/path/filename**—File on an FTP server. You can also specify **hostname** as **username@hostname** or **username:password@hostname**. The default path is the user's home directory. To specify an absolute path, the path must start with **%2F**; for example, **ftp://hostname/%2Fpath/filename**. To have the system prompt you for the password, specify **prompt** in place of the password. If a password is required, and you do not specify the password or **prompt**, an error message is displayed:  

```
user@host> file copy ftp://username@ftp.hostname.net//filename
file copy ftp.hostname.net: Not logged in.

user@host> file copy ftp://username:prompt@ftp.hostname.net//filename
Password for username@ftp.hostname.net:
```
- **http://hostname/path/filename**—File on an HTTP server. You can also specify **hostname** as **username@hostname** or **username:password@hostname**. If a password is required and you omit it, you are prompted for it.
- **re0:/path/filename** or **re1:/path/filename**—File on a local Routing Engine. You can also specify a file on a local Routing Engine for a specific T640 router on a routing matrix:  

```
user@host> show log lcc0-re1:chassisd
```

Related  
Documentation

- [Displaying Junos OS Information on page 177](#)

## Displaying Junos OS Information

You can display Junos OS version information and other status to determine if the version of Junos OS that you are running supports particular features or hardware.

To display Junos OS information:

1. Make sure you are in operational mode.
2. To display brief information and status for the kernel and Packet Forwarding Engine, enter the **show version brief** command. This command shows version information for Junos OS packages installed on the router. For example:

```
user@host> show version brief
Hostname: host
Model: m7i
JUNOS Base OS boot [9.1R1.8]
JUNOS Base OS Software Suite [9.1R1.8]
JUNOS Kernel Software Suite [9.1R1.8]
JUNOS Crypto Software Suite [9.1R1.8]
JUNOS Packet Forwarding Engine Support (M/T Common) [9.1R1.8]
JUNOS Packet Forwarding Engine Support (M7i/M10i) [9.1R1.8]
JUNOS Online Documentation [9.1R1.8]
JUNOS Routing Software Suite [9.1R1.8]
```

```
user@host>
```

If the **Junos Crypto Software Suite** is listed, the router has Canada and USA encrypted Junos OS. If the **Junos Crypto Software Suite** is not listed, the router is running worldwide nonencrypted Junos OS.

3. To display detailed version information, enter the **show version detail** command. This command display shows the hostname and version information for Junos OS packages installed on your router. It also includes the version information for each software process. For example:

```
user@host> show version detail

Hostname: host
Model: m20
JUNOS Base OS boot [8.4R1.13]
JUNOS Base OS Software Suite [8.4R1.13]
JUNOS Kernel Software Suite [8.4R1.13]
JUNOS Crypto Software Suite [8.4R1.13]
JUNOS Packet Forwarding Engine Support (M/T Common) [8.4R1.13]
JUNOS Packet Forwarding Engine Support (M20/M40) [8.4R1.13]
JUNOS Online Documentation [8.4R1.13]
JUNOS Routing Software Suite [8.4R1.13]
KERNEL 8.4R1.13 #0 built by builder on 2007-08-08 00:33:41 UTC
MGD release 8.4R1.13 built by builder on 2007-08-08 00:34:00 UTC
CLI release 8.4R1.13 built by builder on 2007-08-08 00:34:47 UTC
RPD release 8.4R1.13 built by builder on 2007-08-08 00:45:21 UTC
CHASSISD release 8.4R1.13 built by builder on 2007-08-08 00:36:59 UTC
DFWD release 8.4R1.13 built by builder on 2007-08-08 00:39:32 UTC
DCD release 8.4R1.13 built by builder on 2007-08-08 00:34:24 UTC
SNMPD release 8.4R1.13 built by builder on 2007-08-08 00:42:24 UTC
MIB2D release 8.4R1.13 built by builder on 2007-08-08 00:46:47 UTC
APSD release 8.4R1.13 built by builder on 2007-08-08 00:36:39 UTC
VRRPD release 8.4R1.13 built by builder on 2007-08-08 00:45:44 UTC
ALARM release 8.4R1.13 built by builder on 2007-08-08 00:34:30 UTC
PFED release 8.4R1.13 built by builder on 2007-08-08 00:41:54 UTC
CRAFTD release 8.4R1.13 built by builder on 2007-08-08 00:39:03 UTC
SAMPLED release 8.4R1.13 built by builder on 2007-08-08 00:36:05 UTC
ILMID release 8.4R1.13 built by builder on 2007-08-08 00:36:51 UTC
RMOPD release 8.4R1.13 built by builder on 2007-08-08 00:42:04 UTC
```

```

COSD release 8.4R1.13 built by builder on 2007-08-08 00:38:39 UTC
FSAD release 8.4R1.13 built by builder on 2007-08-08 00:43:01 UTC
IRSD release 8.4R1.13 built by builder on 2007-08-08 00:35:37 UTC
FUD release 8.4R1.13 built by builder on 2007-08-08 00:44:36 UTC
RTSPD release 8.4R1.13 built by builder on 2007-08-08 00:29:14 UTC
SMARTD release 8.4R1.13 built by builder on 2007-08-08 00:13:32 UTC
KSYNCD release 8.4R1.13 built by builder on 2007-08-08 00:33:17 UTC
SPD release 8.4R1.13 built by builder on 2007-08-08 00:43:50 UTC
L2TPD release 8.4R1.13 built by builder on 2007-08-08 00:43:12 UTC
HTTPD release 8.4R1.13 built by builder on 2007-08-08 00:36:27 UTC
PPPOED release 8.4R1.13 built by builder on 2007-08-08 00:36:04 UTC
RDD release 8.4R1.13 built by builder on 2007-08-08 00:33:49 UTC
PPPD release 8.4R1.13 built by builder on 2007-08-08 00:45:13 UTC
DFCD release 8.4R1.13 built by builder on 2007-08-08 00:39:11 UTC
DLSWD release 8.4R1.13 built by builder on 2007-08-08 00:42:37 UTC
LACPD release 8.4R1.13 built by builder on 2007-08-08 00:35:41 UTC
USBD release 8.4R1.13 built by builder on 2007-08-08 00:30:01 UTC
LFMD release 8.4R1.13 built by builder on 2007-08-08 00:35:52 UTC
CFMD release 8.4R1.13 built by builder on 2007-08-08 00:34:45 UTC
JDHCPD release 8.4R1.13 built by builder on 2007-08-08 00:35:40 UTC
PGCPD release 8.4R1.13 built by builder on 2007-08-08 00:46:31 UTC
SSD release 8.4R1.13 built by builder on 2007-08-08 00:36:17 UTC
MSPD release 8.4R1.13 built by builder on 2007-08-08 00:33:42 UTC
KMD release 8.4R1.13 built by builder on 2007-08-08 00:44:02 UTC
PPMD release 8.4R1.13 built by builder on 2007-08-08 00:36:03 UTC
LMPD release 8.4R1.13 built by builder on 2007-08-08 00:33:49 UTC
LRMUXD release 8.4R1.13 built by builder on 2007-08-08 00:33:55 UTC
PGMD release 8.4R1.13 built by builder on 2007-08-08 00:36:01 UTC
BFDD release 8.4R1.13 built by builder on 2007-08-08 00:44:22 UTC
SDXD release 8.4R1.13 built by builder on 2007-08-08 00:36:18 UTC
AUDITD release 8.4R1.13 built by builder on 2007-08-08 00:34:40 UTC
L2ALD release 8.4R1.13 built by builder on 2007-08-08 00:40:05 UTC
EVENTD release 8.4R1.13 built by builder on 2007-08-08 00:39:55 UTC
L2CPD release 8.4R1.13 built by builder on 2007-08-08 00:41:04 UTC
MPLSOAMD release 8.4R1.13 built by builder on 2007-08-08 00:45:11 UTC
jroute-dd release 8.4R1.13 built by builder on 2007-08-08 00:31:01 UTC
jkernel-dd release 8.4R1.13 built by builder on 2007-08-08 00:30:30 UTC
jcrypto-dd release 8.4R1.13 built by builder on 2007-08-08 00:30:12 UTC
jdocs-dd release 8.4R1.13 built by builder on 2007-08-08 00:02:52 UTC

```

```
user@host>
```

**Related Documentation** • [Managing Programs and Processes Using Junos OS Operational Mode Commands on page 179](#)

## Managing Programs and Processes Using Junos OS Operational Mode Commands

This topic shows some examples of Junos operational commands that you can use to manage programs and processes on a device running Junos OS.

Sections include:

- [Showing Software Processes on page 180](#)
- [Restarting the Junos OS Process on page 181](#)
- [Stopping Junos OS on page 182](#)
- [Rebooting Junos OS on page 183](#)

## Showing Software Processes

To verify system operation or to begin diagnosing an error condition, you may need to display information about software processes running on the device.

To show software processes:

1. Make sure you are in operational mode.
2. Type the **show system processes extensive** command. This command shows the CPU utilization on the device and lists the processes in order of CPU utilization. For example:

```
user@host> show system processes extensive
```

```
Last pid: 28689; load averages: 0.01, 0.00, 0.00 up 56+06:16:13 04:52:04
73 processes: 1 running, 72 sleeping
```

```
Mem: 101M Active, 101M Inact, 98M Wired, 159M Cache, 69M Buf, 286M Free
Swap: 1536M Total, 1536M Free
```

PID	USERNAME	PRI	NICE	SIZE	RES	STATE	TIME	WCPU	CPU	COMMAND
3365	root	2	0	21408K	4464K	select	511:23	0.00%	0.00%	chassisd
3508	root	2	0	3352K	1168K	select	32:45	0.00%	0.00%	l2ald
3525	root	2	0	3904K	1620K	select	13:40	0.00%	0.00%	dcd
5532	root	2	0	11660K	2856K	kqread	10:36	0.00%	0.00%	rpdp
3366	root	2	0	2080K	828K	select	8:33	0.00%	0.00%	alarmd
3529	root	2	0	2040K	428K	select	7:32	0.00%	0.00%	irsd
3375	root	2	0	2900K	1600K	select	6:01	0.00%	0.00%	ppmd
3506	root	2	0	5176K	2568K	select	5:38	0.00%	0.00%	mib2d
4957	root	2	0	1284K	624K	select	5:16	0.00%	0.00%	ntpd
6	root	18	0	0K	0K	syncer	4:49	0.00%	0.00%	syncer
3521	root	2	0	2312K	928K	select	2:14	0.00%	0.00%	lfmd
3526	root	2	0	5192K	1988K	select	2:04	0.00%	0.00%	snmpd
3543	root	2	0	0K	0K	peer_s	1:46	0.00%	0.00%	peer proxy
3512	root	2	0	3472K	1044K	select	1:44	0.00%	0.00%	rmopd
3537	root	2	0	0K	0K	peer_s	1:30	0.00%	0.00%	peer proxy
3527	root	2	0	3100K	1176K	select	1:14	0.00%	0.00%	pfed
3380	root	2	0	3208K	1052K	select	1:11	0.00%	0.00%	bfdd
4136	root	2	0	11252K	3668K	select	0:54	0.00%	0.00%	cli
3280	root	2	0	2248K	1420K	select	0:28	0.00%	0.00%	eventd
3528	root	2	0	2708K	672K	select	0:28	0.00%	0.00%	dfwd
7	root	-2	0	0K	0K	vlruwt	0:26	0.00%	0.00%	vnlr
3371	root	2	0	1024K	216K	sbwait	0:25	0.00%	0.00%	tnp.sntpd
13	root	-18	0	0K	0K	psleep	0:24	0.00%	0.00%	vmuncacheda
3376	root	2	0	1228K	672K	select	0:22	0.00%	0.00%	smartd
5	root	-18	0	0K	0K	psleep	0:17	0.00%	0.00%	bufdaemon
3368	root	2	0	15648K	9428K	select	0:17	0.00%	0.00%	mgd
3362	root	2	0	1020K	204K	select	0:15	0.00%	0.00%	watchdog
3381	root	2	0	2124K	808K	select	0:15	0.00%	0.00%	lacpd
3524	root	2	0	6276K	1492K	select	0:14	0.00%	0.00%	kmd
3343	root	10	0	1156K	404K	nanslp	0:14	0.00%	0.00%	cron

---(more)---

Table 10 on page 181 lists and describes the output fields included in this example. The fields are listed in alphabetical order.

**Table 10: show system process extensive Command Output Fields**

Field	Description
COMMAND	Command that is running.
CPU	Raw (unweighted) CPU usage. The value of this field is used to sort the processes in the output.
last pid	Last process identifier assigned to the process.
load averages	Three load averages, followed by the current time.
Mem	Information about physical and virtual memory allocation.
NICE	UNIX “nice” value. The nice value allows a process to change its final scheduling priority.
PID	Process identifier.
PRI	Current kernel scheduling priority of the process. A lower number indicates a higher priority.
processes	Number of existing processes and the number of processes in each state ( <b>sleeping</b> , <b>running</b> , <b>starting</b> , <b>zombies</b> , and <b>stopped</b> ).
RES	Current amount of resident memory, in KB.
SIZE	Total size of the process ( <b>text</b> , <b>data</b> , and <b>stack</b> ), in KB.
STATE	Current state of the process ( <b>sleep</b> , <b>wait</b> , <b>run</b> , <b>idle</b> , <b>zombi</b> , or <b>stop</b> ).
Swap	Information about physical and virtual memory allocation.
USERNAME	Owner of the process.
WCPU	Weighted CPU usage.

## Restarting the Junos OS Process

To correct an error condition, you might need to restart a software process running on the device. You can use the **restart** command to force a restart of a software process.



**CAUTION:** Do not restart a software process unless specifically asked to do so by your Juniper Networks customer support representative. Restarting a software process during normal operation of a device could cause interruption of packet forwarding and loss of data.

To restart a software process:

1. Make sure you are in operational mode.
2. Type the following command:

```
user@host> restart process-name < (immediately | gracefully | soft) >
```

- **process-name** is the name of the process that you want to restart. For example, **routing** or **class-of-service**. You can use the command completion feature of Junos OS to see a list of software processes that you can restart using this command.
- **gracefully** restarts the software process after performing clean-up tasks.
- **immediately** restarts the software process without performing any clean-up tasks.
- **soft** rereads and reactivates the configuration without completely restarting the software processes. For example, BGP peers stay up and the routing table stays constant.

The following example shows how to restart the routing process:

```
user@host> restart routing
Routing protocol daemon started, pid 751
```

When a process restarts, the process identifier (PID) is updated. (See [Figure 13 on page 182.](#))

Figure 13: Restarting a Process

PID	USERNAME	PRI	NICE	SIZE	RES	STATE	TIME	WCPU	CPU	COMMAND
546	root	10	0	9096K	1720K	nanslp	0:21	0.00%	0.00%	chassisd
685	root	2	0	12716K	3840K	kqread	0:01	0.00%	0.00%	rpd
553	root	2	0	8792K	1544K	select	0:01	0.00%	0.00%	mib2d

PID before restart

547	root	2	0	7732K	888K	select	0:00	0.00%	0.00%	alarmd
545	root	2	0	10292K	2268K	select	0:00	0.00%	0.00%	dcd
1	root	10	0	816K	520K	wait	0:00	0.00%	0.00%	init
550	root	2	-12	1308K	692K	select	0:00	0.00%	0.00%	ntpd
758	root	32	0	21716K	832K	RUN	0:00	0.00%	0.00%	top
560	root	2	0	8208K	1088K	select	0:00	0.00%	0.00%	rmopd
561	root	2	0	8188K	1156K	select	0:00	0.00%	0.00%	cosd
559	root	2	0	1632K	840K	select	0:00	0.00%	0.00%	ilmid
573	lab	2	0	7480K	2580K	select	0:00	0.00%	0.00%	cli
751	root	2	0	12716K	3944K	kqread	0:00	0.00%	0.00%	rpd
558	root	2	20	8708K	1880K	select	0:00	0.00%	0.00%	sampld
555	root	2	0	1856K	932K	select	0:00	0.00%	0.00%	vrrpd
686	root	2	0	7808K	940K	select	0:00	0.00%	0.00%	apsd

PID after restart

## Stopping Junos OS

To avoid damage to the file system and to prevent loss of data, you must always gracefully shut down Junos OS before powering off the device.



**NOTE:** SRX Series Services Gateway devices for the branch and EX Series Ethernet Switches support resilient dual-root partitioning.

If you are unable to shut down a device gracefully because of unexpected circumstances such as a power outage or a device failure, resilient dual-root partitioning prevents file corruption and enables a device to remain operational. In addition, it enables a device to boot transparently from the second root partition if the system fails to boot from the primary root partition.

Resilient dual-root partitioning serves as a backup mechanism for providing additional resiliency to a device when there is an abnormal shutdown. However, it is not an alternative to performing a graceful shutdown under normal circumstances.

To stop Junos OS:

1. Make sure you are in operational mode.
2. Enter the **request system halt** command. This command stops all system processes and halts the operating system. For example:

```
user@host> request system halt
Halt the system? [yes,no] (no) yes
shutdown: [pid 3110]
Shutdown NOW!
*** FINAL System shutdown message from root@host ***
System going down IMMEDIATELY
user@host> Dec 17 17:28:40 init: syslogd (PID 2514) exited with status=0 Normal
Exit
Waiting (max 60 seconds) for system process `bufdaemon' to stop...stopped
Waiting (max 60 seconds) for system process `syncer' to stop...stopped
syncing disks... 4
done
Uptime: 3h31m41s
ata0: resetting devices.. done
The operating system has halted.
Please press any key to reboot.
```

## Rebooting Junos OS

After a software upgrade or to recover (occasionally) from an error condition, you must reboot Junos OS.

To reboot Junos OS:

1. Make sure you are in operational mode.
2. Enter the **request system reboot** command. This command displays the final stages of the system shutdown and executes the reboot. Reboot requests are recorded to the system log files, which you can view with the **show log messages** command. For example:

```
user@host> request system reboot
Reboot the system? [yes,no] (no) yes
```

```
shutdown: [pid 845]
Shutdown NOW!
*** FINAL System shutdown message from root@host ***
System going down IMMEDIATELY
user@host> Dec 17 17:34:20 init: syslogd (PID 409) exited with status=0 Normal
Exit
Waiting (max 60 seconds) for system process `bufdaemon' to stop...stopped
Waiting (max 60 seconds) for system process `syncer' to stop...stopped
syncing disks... 10 6
done
Uptime: 2m45s
ata0: resetting devices.. done
Rebooting...
```

- Related Documentation**
- [Checking the Status of a Device Running Junos OS on page 36](#)
  - [Displaying Junos OS Information on page 177](#)

---

## Using the Junos OS CLI Comment Character # for Operational Mode Commands

The comment character in Junos OS enables you to copy operational mode commands that include comments from a file and paste them into the CLI. A pound sign (#) at the beginning of the command-line indicates a comment line. This is useful for describing frequently used operational mode commands; for example, a user's work instructions on how to monitor the network. To add a comment to a command file, the first character of the line must be #. When you start a command with #, the rest of the line is disregarded by Junos OS.

To add comments in operational mode, start with a # and end with a new line (carriage return):

```
user@host> # comment-string
```

*comment-string* is the text of the comment. The comment text can be any length, but each comment line must begin with a #.

- Related Documentation**
- [Example: Using Comments in Junos OS Operational Mode Commands on page 184](#)

---

## Example: Using Comments in Junos OS Operational Mode Commands

The following example shows how to use comments in a file:

```
#Command 1: Show the router version
show version
#Command 2: Show all router interfaces
show interfaces terse
```

The following example shows how to copy and paste contents of a file into the CLI:

```
user@host> #Command 1: Show the router version
user@host> show version
Hostname: myhost
Model: m5
```



```

Junos Base OS boot [6.4-20040511.0]
Junos Base OS Software Suite [6.4-20040511.0]
Junos Kernel Software Suite [6.4-20040511.0]
Junos Packet Forwarding Engine Support (M5/M10) [6.4-20040511.0] Junos Routing
 Software Suite [6.4-20040511.0] Junos Online Documentation [6.4-20040511.0] Junos
 Crypto Software Suite [6.4-20040511.0]
user@host> # Command 2: Show all router interfaces
user@host> show interfaces terse
Interface Admin Link Proto Local Remote
fe-0/0/0 up up
fe-0/0/1 up down
fe-0/0/2 up down
mo-0/1/0 up
mo-0/1/0.16383 up up inet 10.0.0.1 --> 10.0.0.17
so-0/2/0 up up
so-0/2/1 up up
dsc up up
fxp0 up up
fxp0.0 up up inet 192.168.70.62/21
fxp1 up up
fxp1.0 up up tnp 4
gre up up
ipip up up
lo0 up up
lo0.0 up up inet 127.0.0.1 --> 0/0
lo0.16385 up up inet

```

- Related Documentation**
- [Using the Junos OS CLI Comment Character # for Operational Mode Commands on page 184](#)



## CHAPTER 8

# Filtering Command Output

- [Using the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 187](#)
- [Using Regular Expressions with the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 188](#)
- [Filtering Operational Mode Command Output in a QFabric System on page 189](#)
- [Pipe \( | \) Filter Functions in the Junos OS Command-Line Interface on page 190](#)

## Using the Pipe ( | ) Symbol to Filter Junos OS Command Output

---

The Junos OS enables you to filter command output by adding the pipe ( | ) symbol when you enter a command.

For example:

```
user@host> show rip neighbor ?
Possible completions:
<[Enter]> Execute this command
<name> Name of RIP neighbor
instance Name of RIP instance
logical-system Name of logical system, or 'all'
| Pipe through a command
```

The following example lists the filters that can be used with the pipe symbol ( | ):

```
user@host> show interfaces | ?
user@host> show interfaces | ?
Possible completions:
append Append output text to file
count Count occurrences
display Show additional kinds of information
except Show only text that does not match a pattern
find Search for first occurrence of pattern
hold Hold text without exiting the --More-- prompt
last Display end of output only
match Show only text that matches a pattern
no-more Don't paginate output
refresh Refresh a continuous display of the command
request Make system-level requests
resolve Resolve IP addresses
save Save output text to file
tee Write to standard output and file
trim Trim specified number of columns from start of line
```

For the **show configuration** command only, an additional compare filter is available:

```
user@host> show configuration | ?
Possible completions:
 compare Compare configuration changes with prior version
 ...
```

You can enter any of the pipe filters in conjunction. For example:

```
user@host> command | match regular-expression | save filename
```



**NOTE:** This topic describes *only* the filters that can be used for operational mode command output. For information about filters that can be used in configuration mode, see the *Junos OS Administration Library*.

#### Related Documentation

- [Pipe \( | \) Filter Functions in the Junos OS Command-Line Interface on page 190](#)
- [Using Regular Expressions with the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 188](#)
- [Filtering Operational Mode Command Output in a QFabric System on page 189](#)

## Using Regular Expressions with the Pipe ( | ) Symbol to Filter Junos OS Command Output

The **except**, **find**, and **match** filters used with the pipe symbol employ regular expressions to filter output. Juniper Networks uses the regular expressions as defined in POSIX 1003.2. If the regular expressions contain spaces, operators, or wildcard characters, enclose the expression in quotation marks.

**Table 11: Common Regular Expression Operators in Operational Mode Commands**

Operator	Function
	Indicates that a match can be one of the two terms on either side of the pipe.
^	Used at the beginning of an expression, denotes where a match should begin.
\$	Used at the end of an expression, denotes that a term must be matched exactly up to the point of the \$ character.
[ ]	Specifies a range of letters or digits to match. To separate the start and end of a range, use a hyphen ( - ).
( )	Specifies a group of terms to match.

For example, if a command produces the following output:

```

12
22
321
4

```

a pipe filter of **| match 2** displays the following output:

```

12
22
321

```

and a pipe filter of **| except 1** displays the following output:

```

22
4

```

- Related Documentation**
- [Using the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 187](#)
  - [Pipe \( | \) Filter Functions in the Junos OS Command-Line Interface on page 190](#)

## Filtering Operational Mode Command Output in a QFabric System

When you issue an operational mode command in a QFabric system, the output generated can be fairly extensive because of the number of components contained within the system. To make the output more accessible, you can filter the output by appending the **| filter** option to the end of most Junos OS commands.

1. To filter operational mode command output and limit it to a Node group, include the **| filter node-group *node-group-name*** option at the end of your Junos OS operational mode command.

```
root@qfabric> show interfaces terse | filter node-group NW-NG-0
```

Interface	Admin	Link	Proto	Local	Remote
NW-NG-0:dsc	up	up			
NW-NG-0:em0	up	up			
NW-NG-0:em1	up	up			
NW-NG-0:gre	up	up			
NW-NG-0:ipip	up	up			
NW-NG-0:lo0	up	up			
NW-NG-0:lo0.16384	up	up	inet	127.0.0.1	--> 0/0
NW-NG-0:lo0.16385	up	up	inet		
NW-NG-0:lsi	up	up			
NW-NG-0:mtun	up	up			
NW-NG-0:pimd	up	up			
NW-NG-0:pime	up	up			
NW-NG-0:tap	up	up			
Node01:ge-0/0/10	up	up			
Node01:ge-0/0/40	up	up			
Node01:ge-0/0/41	up	up			
vlan	up	up			

2. To filter operational mode command output and limit it to a set of Node groups, include the **| filter node-group** option at the end of your Junos OS operational mode command and specify the list of Node group names in brackets.

```
root@qfabric> show ethernet-switching interfaces | filter node-group [NW-NG-0 RSNG-1]
```

Interface	State	VLAN members	Tag	Tagging	Blocking
NW-NG-0:ae0.0	up	v200	200	tagged	unblocked
		v50	50	tagged	unblocked
		v51	51	tagged	unblocked
		v52	52	tagged	unblocked
		v53	53	tagged	unblocked
RSNG-1:ae0.0	up	v200	200	untagged	unblocked
RSNG-1:ae47.0	up	v50	50	tagged	unblocked
		v51	51	tagged	unblocked
		v52	52	tagged	unblocked
		v53	53	tagged	unblocked

**Related Documentation** • [Using the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 187](#)

## Pipe ( | ) Filter Functions in the Junos OS Command-Line Interface

This topic describes the pipe ( | ) filter functions that are supported in the Junos OS command-line interface (CLI):

- [Comparing Configurations and Displaying the Differences in Text on page 190](#)
- [Comparing Configurations and Displaying the Differences in XML on page 192](#)
- [Counting the Number of Lines of Output on page 192](#)
- [Displaying Output in XML Tag Format on page 192](#)
- [Displaying Output in JSON Format on page 193](#)
- [Displaying the Configuration with YANG Translation Scripts Applied on page 193](#)
- [Displaying the RPC Tags for a Command on page 195](#)
- [Ignoring Output That Does Not Match a Regular Expression on page 195](#)
- [Displaying Output from the First Match of a Regular Expression on page 195](#)
- [Retaining Output After the Last Screen on page 196](#)
- [Displaying Output Beginning with the Last Entries on page 196](#)
- [Displaying Output That Matches a Regular Expression on page 196](#)
- [Preventing Output from Being Paginated on page 197](#)
- [Sending Command Output to Other Users on page 197](#)
- [Resolving IP Addresses on page 197](#)
- [Saving Output to a File on page 198](#)
- [Appending Output to a File on page 198](#)
- [Displaying Output on Screen and Writing to a File on page 198](#)
- [Trimming Output by Specifying the Starting Column on page 199](#)
- [Refreshing the Output of a Command on page 199](#)

### Comparing Configurations and Displaying the Differences in Text

The **compare** filter compares the candidate configuration with either the current committed configuration or a configuration file and displays the differences between

the two configurations with text characters. To compare configurations, enter **compare** after the pipe ( | ) symbol:

```
[edit]
user@host# show | compare [filename] rollback n]
```

**filename** is the full path to a configuration file.

**n** is the index into the list of previously committed configurations. The most recently saved configuration is 0. If you do not specify arguments, the candidate configuration is compared against the active configuration file (**/config/juniper.conf**).

The comparison output uses the following conventions:

- Statements that are only in the candidate configuration are prefixed with a plus sign (+).
- Statements that are only in the comparison file are prefixed with a minus sign (-).
- Statements that are unchanged are prefixed with a single blank space ( ).

For example:

```
user@host> show configuration system | compare rollback 9
[edit system]
+ host-name device;
+ backup-router 192.168.71.254;
- ports {
- console log-out-on-disconnect;
- }
[edit system name-server]
+ 172.17.28.11;
 172.17.28.101 { ... }
[edit system name-server]
 172.17.28.101 { ... }
+ 172.17.28.100;
+ 172.17.28.10;
[edit system]
- scripts {
- commit {
- allow-transients;
- }
- }
+ services {
+ ftp;
+ rlogin;
+ rsh;
+ telnet;
+ }
```

Starting with Junos OS Release 8.3, output from the **show | compare** command has been enhanced to more accurately reflect configuration changes. This includes more intelligent handling of order changes in lists. For example, consider names in a group that are reordered as follows:

```
groups { groups {
group_xmp; group_xmp;
group_cmp; group_grp:
group_grp; group_cmp;
} }
```

In previous releases, output from the **show | compare** command looked like the following:

```
[edit groups]
- group_xmp;
- group_cmp;
- group_grp;
+ group_xmp;
+ group_grp;
+ group_cmp;
```

Now, output from the **show | compare** command looks like the following:

```
[edit groups]
group_xmp {...}
! group_grp {...}
```

## Comparing Configurations and Displaying the Differences in XML

The **compare | display xml** filter compares the candidate configuration with the current committed configuration and displays the differences between the two configurations in XML. To compare configurations, enter **compare | display xml** after the pipe (|) symbol in either operational or configuration mode.

Example in operational mode:

```
user@host> show configuration | compare | display xml
```

Example in configuration mode:

```
[edit]
user@host# show | compare | display xml
```

You can enter a specific configuration hierarchy prior to **| compare**. In configuration mode, you can navigate to a hierarchy where the command is applied.

For a description of the XML output, see [“Understanding the show | compare | display xml Command Output” on page 128](#).

## Counting the Number of Lines of Output

To count the number of lines in the output from a command, enter **count** after the pipe symbol (|). For example:

```
user@host> show configuration | count
Count: 269 lines
```

## Displaying Output in XML Tag Format

To display command output in XML tag format, enter **display xml** after the pipe symbol (|).

The following example displays the **show cli directory** command output as XML tags:

```
user@host> show cli directory | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/7.5I0/junos">
 <cli>
 <working-directory>/var/home/user</working-directory>
 </cli>
```



```
<cli>
 <banner></banner>
</cli>
</rpc-reply>
```

To display the change in the candidate and active configurations in XML tag format, see [“Comparing Configurations and Displaying the Differences in XML” on page 192](#).

## Displaying Output in JSON Format

Starting in Junos OS Release 14.2, you can display the configuration or command output in JavaScript Object Notation (JSON) format by entering **display json** after the pipe symbol (**|**).

The following example displays the **show cli directory** command output in JSON format:

```
user@host> show cli directory | display json
```

```
{
 "cli" : [
 {
 "working-directory" : [
 {
 "data" : "/var/home/username"
 }
]
 }
]
}
```



**NOTE:** Starting in Junos OS Release 16.1, devices running Junos OS emit JSON-formatted configuration data using a new default implementation for serialization.



**NOTE:** Starting in Junos OS Release 17.1, integers in Junos OS configuration data emitted in JSON format are not enclosed in quotation marks. Prior to Junos OS Release 17.1, integers in JSON configuration data were treated as strings and enclosed in quotation marks.

## Displaying the Configuration with YANG Translation Scripts Applied

Starting in Junos OS Release 16.1, you can load YANG modules onto devices running Junos OS to augment the configuration hierarchy with data models that are not natively supported by Junos OS but can be supported by translation. The active and candidate configurations contain the configuration data for non-native YANG data models in the syntax defined by that model, but they do not explicitly display the corresponding translated Junos OS syntax, which is committed as a transient change.

The **| display translation-scripts** filter displays the complete post-inheritance configuration, with the translated configuration data from all enabled translation scripts explicitly included in the output. To display the configuration with all enabled YANG translation

scripts applied, append the **| display translation-scripts** filter to the **show configuration** command in operational mode or the **show** command in configuration mode. For example:

```
user@host> show configuration | display translation-scripts
```

To view just the non-native configuration data after translation, use the **| display translation-scripts translated-config** filter in either operational or configuration mode.

```
user@host> show configuration | display translation-scripts translated-config
```

In configuration mode, to display just the configuration differences in the hierarchies corresponding to non-native YANG data models before or after translation scripts are applied, append the **configured-delta** or **translated-delta** keyword, respectively, to the **show | display translation-scripts** command. In both cases, the XML output displays the deleted configuration data, followed by the new configuration data.

```
user@host# show | display-translation-scripts (configured-delta | translated-delta)
```

The following example displays a sample configuration with and without translation scripts applied. The **show** command displays the configuration, which includes the non-native configuration data in the syntax defined by the YANG data model. The **| display translation-scripts** filter displays the non-native configuration data in both the syntax defined by the YANG data model and the translated Junos OS syntax. Both commands display the entire configuration, which has been truncated for brevity in this example. However, the **show** command returns the pre-inheritance configuration, whereas the **show | display translation-scripts** command returns the post-inheritance configuration.

```
user@host# show
...
myint:intconfig {
 interfaces {
 interface ge-0/0/0 {
 config {
 description test;
 }
 }
 }
}
...

user@host# show | display translation-scripts
...
interfaces {
 ge-0/0/0 {
 description test;
 gigger-options {
 no-flow-control;
 }
 }
}
...
myint:intconfig {
 interfaces {
 interface ge-0/0/0 {
 config {
 description test;
 }
 }
 }
}
```

```
 }
 }
 ...

```

## Displaying the RPC Tags for a Command

To display the remote procedure call (RPC) XML tags for an operational mode command, enter **display xml rpc** after the pipe symbol ( | ).

The following example displays the RPC tags for the **show route** command:

```
user@host> show route | display xml rpc
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/10.1I0/junos">
 <rpc>
 <get-route-information>
 </get-route-information>
 </rpc>
 <cli>
 <banner></banner>
 </cli>
</rpc-reply>

```

## Ignoring Output That Does Not Match a Regular Expression

To ignore text that matches a regular expression, specify the **except** command after the pipe symbol ( | ). If the regular expression contains any spaces, operators, or wildcard characters, enclose it in quotation marks. For information on common regular expression operators, see [“Using Regular Expressions with the Pipe \( | \) Symbol to Filter Junos OS Command Output” on page 188](#).

The following example displays all users who are logged in to the router, except for the user **root**:

```
user@host> show system users | except root
 8:28PM up 1 day, 13:59, 2 users, load averages: 0.01, 0.01, 0.00
USER TTY FROM LOGIN@ IDLE WHAT
user p0 device1.example.com 7:25PM - cli

```

## Displaying Output from the First Match of a Regular Expression

To display output starting with the first occurrence of text matching a regular expression, enter **find** after the pipe symbol ( | ). If the regular expression contains any spaces, operators, or wildcard characters, enclose it in quotation marks. For information on common regular expression operators, see [“Using Regular Expressions with the Pipe \( | \) Symbol to Filter Junos OS Command Output” on page 188](#).

The following example displays the routes in the routing table starting at IP address **208.197.169.0**:

```
user@host> show route | find 208.197.169.0
208.197.169.0/24 *[Static/5] 1d 13:22:11
 > to 192.168.4.254 via so-3/0/0.0
224.0.0.5/32 *[OSPF/10] 1d 13:22:12, metric 1
iso.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
47.0005.80ff.f800.0000.0108.0001.1921.6800.4015.00/160

```

```
*[Direct/0] 1d 13:22:12
> via 10.0.0
```

The following example displays the first CCC entry in the forwarding table:

```
user@host> show route forwarding-table | find ccc
Routing table: ccc
MPLS:
Interface.Label Type RtRef Nexthop Type Index NhRef Netif
default perm 0 10.0.16.2 rjct 3 1
0 user 0 10.0.16.2 recv 5 2
1 user 0 10.0.16.2 recv 5 2
32769 user 0 10.0.16.2 ucst 45 1 fe-0/0/0.534
fe-0/0/0. (CCC) user 0 10.0.16.2 indr 44 2
 Push 32768, Push
```

## Retaining Output After the Last Screen

To not return immediately to the CLI prompt after viewing the last screen of output, enter **hold** after the pipe symbol ( | ). The following example prevents returning to the CLI prompt after you have viewed the last screen of output from the **show log log-file-1** command:

```
user@host> show log log-file-1 | hold
```

This filter is useful when you want to scroll or search through output.

## Displaying Output Beginning with the Last Entries

To display text starting from the end of the output, enter **last <lines>** after the pipe symbol ( | ).

The following example displays the last entries in **log-file-1** file:

```
user@host> show log log-file-1 | last
```

This filter is useful for viewing log files in which the end of the file contains the most recent entries.



**NOTE:** When the number of lines requested is less than the number of lines that the screen length setting permits you to display, Junos OS returns as many lines as permitted by the screen length setting. That is, if your screen length is set to 20 lines and you have requested only the last 10 lines, Junos OS returns the last 19 lines instead of the last 10 lines.

## Displaying Output That Matches a Regular Expression

To display output that matches a regular expression, enter **match *regular-expression*** after the pipe symbol ( | ). If the regular expression contains any spaces, operators, or wildcard characters, enclose it in quotation marks. For information on common regular expression operators, see [“Using Regular Expressions with the Pipe \( | \) Symbol to Filter Junos OS Command Output”](#) on page 188.

The following example matches all the Asynchronous Transfer Mode (ATM) interfaces in the configuration:

```
user@host> show configuration | match at-
at-2/1/0 {
at-2/1/1 {
at-2/2/0 {
at-5/2/0 {
at-5/3/0 {
```

## Preventing Output from Being Paginated

By default, if output is longer than the length of the terminal screen, you are provided with a **---(more)---** message to display the remaining output. To display the remaining output, press the Spacebar.

To prevent the output from being paginated, enter **no-more** after the pipe symbol ( | ).

The following example displays output from the **show configuration** command all at once:

```
user@host> show configuration | no-more
```

This feature is useful, for example, if you want to copy the entire output and paste it into an e-mail.

## Sending Command Output to Other Users

To display command output on the terminal of a specific user logged in to your router, or on the terminals of all users logged in to your router, enter **request message (all | user account@terminal)** after the pipe symbol ( | ).

If you are troubleshooting your router and, for example, talking with a customer service representative on the phone, you can use the **request message** command to send your representative the command output you are currently viewing on your terminal.

The following example sends the output from the **show interfaces** command you enter on your terminal to the terminal of the user **root@tty1**:

```
user@host> show interfaces | request message user root@tty1
```

The user **root@tty1** sees the following output appear on the terminal screen:

```
Message from user@host on /dev/tty0 at 10:32 PST...
Physical interface: dsc, Enabled, Physical link is Up
Interface index: 5, SNMP ifIndex: 5
Type: Software-Pseudo, MTU: Unlimited...
```

## Resolving IP Addresses

In operational mode only, if the output of a command displays an unresolved IP address, you can enter **| resolve** after the command to display the name associated with the IP address. The **resolve** filter enables the system to perform a reverse DNS lookup of the IP address. If DNS is not enabled, the lookup fails and no substitution is performed.

To perform a reverse DNS lookup of an unresolved IP address, enter **resolve <full-names>** after the pipe symbol ( | ). If you do not specify the **full-names** option, the name is truncated to fit whatever field width limitations apply to the IP address.

The following example performs a DNS lookup on any unresolved IP addresses in the output from the **show ospf neighbors** command:

```
user@host> show ospf neighbors | resolve
```

## Saving Output to a File

When command output is lengthy, when you need to store or analyze the output, or when you need to send the output in an e-mail or by FTP, you can save the output to a file. By default, the file is placed in your home directory on the router.

To save command output to a file, enter **save filename** after the pipe symbol ( | ).

The following example saves the output from the **request support information** command to a file named **my-support-info.txt**:

```
user@host> request support information | save my-support-info.txt
Wrote 1143 lines of output to 'my-support-info.txt'
user@host>
```

## Appending Output to a File

When command output is displayed, you can either save the output to a file, which overwrites the existing contents of that file or you can append the output text to a specific file.

To append the command output to the file, enter **append filename** after the pipe symbol ( | ).

The following example appends the output from the **request support information** command to a file named **my-support-info.txt**:

```
user@host> request support information | append my-support-info.txt
Wrote 2247 lines of output to 'my-support-info.txt'
user@host>
```

## Displaying Output on Screen and Writing to a File

When command output is displayed, you can also write the output to a file. To both display the output and write it to a file, enter **tee filename** after the pipe symbol ( | ).

The following example displays the output from the **show interfaces ge-\* terse** command (displaying information about the status of the Gigabit Ethernet interfaces on the device) and diverts the output to a file called **ge-interfaces.txt**:

```
user@host> show interfaces ge-* terse | tee ge-interfaces.txt
Interface Admin Link Proto Local Remote
ge-0/1/0 up down
ge-0/1/1 up up
ge-0/1/2 up down
ge-0/1/3 up up
```

Unlike the UNIX **tee** command, only an error message is displayed if the file cannot be opened (instead of displaying the output and then the error message).

```
user@host> show interfaces ge-* terse | tee /home/user/test.txt
error: tee failed: file /home/user/test.txt could not be opened

user@host>
```

## Trimming Output by Specifying the Starting Column

Output appears on the terminal screen in terms of rows and columns. The first alphanumeric character starting at the left of the screen is in column 1, the second character is in column 2, and so on. To display output starting from a specific column (thus trimming the leftmost portion of the output), enter **trim columns** after the pipe symbol (|). The **trim** filter is useful for trimming the date and time from the beginning of system log messages.

The following example displays output from the **show system storage** command, filtering out the first 10 columns:

```
user@host> show system storage | trim 11
```



**NOTE:** The **trim** command does not accept negative values.

## Refreshing the Output of a Command

You can run an operational mode command with the **| refresh** pipe option to refresh the output displayed on the screen periodically. The default refresh occurs every second. However, you can also explicitly specify a refresh interval from 1 through 604800 seconds. For example, to refresh the output of the **show interfaces** command every five seconds, you would run the following command:

```
user@host> show interfaces | refresh 5
```

**Release History Table**

Release	Description
17.1	Starting in Junos OS Release 17.1, integers in Junos OS configuration data emitted in JSON format are not enclosed in quotation marks. Prior to Junos OS Release 17.1, integers in JSON configuration data were treated as strings and enclosed in quotation marks.
16.1	Starting in Junos OS Release 16.1, devices running Junos OS emit JSON-formatted configuration data using a new default implementation for serialization.
16.1	Starting in Junos OS Release 16.1, you can load YANG modules onto devices running Junos OS to augment the configuration hierarchy with data models that are not natively supported by Junos OS but can be supported by translation.
14.2	Starting in Junos OS Release 14.2, you can display the configuration or command output in JavaScript Object Notation (JSON) format by entering <b>display json</b> after the pipe symbol (   ).

**Related Documentation**

- [Using Regular Expressions with the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 188](#)
- [Using the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 187](#)



## CHAPTER 9

# Using Shortcuts, Wildcards, and Regular Expressions in the CLI

- Using Keyboard Sequences to Move Around and Edit the Junos OS CLI on page 201
- Using Wildcard Characters in Interface Names on page 203
- Common Regular Expressions to Use with the replace Command on page 204
- Using Global Replace in the Junos OS Configuration on page 205
- Example: Using Global Replace in a Junos OS Configuration—Using the \n Back Reference on page 206
- Example: Using Global Replace in a Junos OS Configuration—Replacing an Interface Name on page 208
- Example: Using Global Replace in a Junos OS Configuration—Using the upto Option on page 210
- Using Regular Expressions to Delete Related Items from a Junos OS cConfiguration on page 211

### Using Keyboard Sequences to Move Around and Edit the Junos OS CLI

You can use keyboard sequences in the Junos OS command-line interface (CLI) to move around and edit the command line. You can also use keyboard sequences to scroll through a list of recently executed commands. [Table 12 on page 202](#) lists some of the CLI keyboard sequences. They are the same as those used in Emacs.

Table 12: CLI Keyboard Sequences

Category	Action	Keyboard Sequence
Move the Cursor	Move the cursor back one character.	Ctrl+b
	Move the cursor back one word.	Esc+b or Alt+b
	Move the cursor forward one character.	Ctrl+f
	Move the cursor forward one word.	Esc+f or Alt+f
	Move the cursor to the beginning of the command line.	Ctrl+a
	Move the cursor to the end of the command line.	Ctrl+e
Delete Characters	Delete the character before the cursor.	Ctrl+h, Delete, or Backspace
	Delete the character at the cursor.	Ctrl+d
	Delete all characters from the cursor to the end of the command line.	Ctrl+k
	Delete all characters on the command line.	Ctrl+u or Ctrl+x
	Delete the word before the cursor.	Ctrl+w, Esc+Backspace, or Alt+Backspace
	Delete the word after the cursor.	Esc+d or Alt+d
Insert Recently Deleted Text	Insert the most recently deleted text at the cursor.	Ctrl+y
Redraw the Screen	Redraw the current line.	Ctrl+l

Table 12: CLI Keyboard Sequences (*continued*)

Category	Action	Keyboard Sequence
Display Previous Command Lines	Scroll backward through the list of recently executed commands.	Ctrl+p
	Scroll forward through the list of recently executed commands.	Ctrl+n
	Search the CLI history in reverse order for lines matching the search string.	Ctrl+r
	Search the CLI history by typing some text at the prompt, followed by the keyboard sequence. The CLI attempts to expand the text into the most recent word in the history for which the text is a prefix.	Esc+/ sequence
Display Previous Command Words	Scroll backward through the list of recently entered words in a command line.	Esc+. or Alt+.
Repeat Keyboard Sequences	Specify the number of times to execute a keyboard sequence. <i>number</i> can be from 1 through 9 and <i>sequence</i> is the keyboard sequence that you want to execute.	Esc+ <i>number</i> sequence or Alt+ <i>number</i> sequence

- Related Documentation**
- [Using Wildcard Characters in Interface Names on page 203](#)
  - [Using Global Replace in the Junos OS Configuration on page 205](#)

## Using Wildcard Characters in Interface Names

You can use wildcard characters in the Junos OS operational commands to specify groups of interface names without having to type each name individually. [Table 13 on page 203](#) lists the available wildcard characters. You must enclose all wildcard characters except the asterisk (\*) in quotation marks (" ").

Table 13: Wildcard Characters for Specifying Interface Names

Wildcard Character	Description
* (asterisk)	Match any string of characters in that position in the interface name. For example, <b>so*</b> matches all SONET/SDH interfaces.
"[ <i>character</i> < <i>character</i> ...>]"	Match one or more individual characters in that position in the interface name. For example, <b>so-"[03]"*</b> matches all SONET/SDH interfaces in slots 0 and 3.

Table 13: Wildcard Characters for Specifying Interface Names (*continued*)

Wildcard Character	Description
"[!character<character...>]"	Match all characters except the ones included in the brackets. For example, <b>so- "[!03]" *</b> matches all SONET/SDH interfaces except those in slots 0 and 3.
"[character1-character2]"	Match a range of characters. For example, <b>so- "[0-3]" *</b> matches all SONET/SDH interfaces in slots 0, 1, 2, and 3.
"[!character1-character2]"	Match all characters that are not in the specified range of characters. For example, <b>so- "[!0-3]" *</b> matches all SONET/SDH interfaces in slots 4, 5, 6, and 7.

- Related Documentation**
- [Using Keyboard Sequences to Move Around and Edit the Junos OS CLI on page 201](#)
  - [Using Global Replace in the Junos OS Configuration on page 205](#)

## Common Regular Expressions to Use with the replace Command

Table 14: Common Regular Expressions to Use with the replace Command

Operator	Function
	Indicates that a match can be one of the two terms on either side of the pipe.
^	Used at the beginning of an expression, denotes where a match should begin.
\$	Used at the end of an expression, denotes that a term must be matched exactly up to the point of the \$ character.
[ ]	Specifies a range of letters or digits to match. To separate the start and end of a range, use a hyphen ( - ).
( )	Specifies a group of terms to match. Stored as numbered variables. Use for back references as \1 \2 .... \9.
*	0 or more terms.
+	One or more terms.
.	Any character except for a space ( " ").
\	A backslash escapes special characters to suppress their special meaning. For example, \. matches . (period symbol).
\n	Back reference. Matches the <i>n</i> th group.
&	Back reference. Matches the entire match.

Table 15 on page 205 lists some replacement examples.

**Table 15: Replacement Examples**

Command	Result
<code>replace pattern myrouter with router1</code>	Match: <b>myrouter</b> Result: <b>router1</b>
<code>replace pattern "192.168\.(.*)/24" with "10.2.1/28"</code>	Match: <b>192.168.3.4/24</b> Result: <b>10.2.3.4/28</b>
<code>replace pattern "1.1" with "abc&amp;def"</code>	Match: <b>1.1</b> Result: <b>abc1.1def</b>
<code>replace pattern 1.1 with "abc\&amp;def"</code>	Match: <b>1#1</b> Result: <b>abc&amp;def</b>

**Related Documentation**

- [Using Global Replace in the Junos OS Configuration on page 205](#)
- [Example: Using Global Replace in a Junos OS Configuration—Using the \n Back Reference on page 206](#)

## Using Global Replace in the Junos OS Configuration

You can make global changes to variables and identifiers in the Junos OS configuration by using the **replace** configuration mode command. This command replaces a pattern in a configuration with another pattern. For example, you can use this command to find and replace all occurrences of an interface name when a PIC is moved to another slot in the router.

```
user@host# replace pattern pattern1 with pattern2 <upto n>
```

***pattern1*** is a text string or regular expression that defines the identifiers and values you want to replace in the configuration.

***pattern2*** is a text string or regular expression that replaces the identifiers and values located with ***pattern1***.

Juniper Networks uses standard UNIX-style regular expression syntax (as defined in POSIX 1003.2). If the regular expression contains spaces, operators, or wildcard characters, enclose the expression in quotation marks. Greedy qualifiers (match as much as possible) are supported. Lazy qualifiers (match as little as possible) are not.

The **upto *n*** option specifies the number of objects replaced. The value of ***n*** controls the total number of objects that are replaced in the configuration (not the total number of times the pattern occurs). Objects at the same hierarchy level (siblings) are replaced first. Multiple occurrences of a pattern within a given object are considered a single replacement. For example, if a configuration contains a **010101** text string, the command

**replace pattern 01 with pattern 02 upto 2** replaces 010101 with 020202 (instead of 020201). Replacement of 010101 with 020202 is considered a single replacement ( $n = 1$ ), not three separate replacements ( $n = 3$ ).

If you do not specify an **upto** option, all identifiers and values in the configuration that match *pattern1* are replaced.

The **replace** command is available in configuration mode at any hierarchy level. All matches are case-sensitive.

#### Related Documentation

- [Common Regular Expressions to Use with the replace Command on page 204](#)
- [Example: Using Global Replace in a Junos OS Configuration—Using the \n Back Reference on page 206](#)
- [Example: Using Global Replace in a Junos OS Configuration—Replacing an Interface Name on page 208](#)
- [Example: Using Global Replace in a Junos OS Configuration—Using the upto Option on page 210](#)
- [Using Wildcard Characters in Interface Names on page 203](#)
- [Using Keyboard Sequences to Move Around and Edit the Junos OS CLI on page 201](#)

---

### Example: Using Global Replace in a Junos OS Configuration—Using the \n Back Reference

---

This example shows how you can use a backreference to replace a pattern.

- [Requirements on page 206](#)
- [Overview on page 207](#)
- [Configuration on page 207](#)

#### Requirements

No special configuration beyond device initiation is required before configuring this example.

Before you begin, configure the following:

```
[edit]
user@host# show interfaces
xe-0/0/0 {
 unit 0;
}
fe-3/0/1 {
 vlan-tagging;
 unit 0 {
 description "inet6 configuration. IP: 2000::c0a8::1bf5";
 vlan-id 100;
 family inet {
 address 17.10.1.1/24;
 }
 }
}
```

```

 family inet6 {
 address 2000::c0a8:1bf5/3;
 }
}

```

To quickly configure this initial configuration, copy the following commands and paste them in a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level:

```

set interfaces xe-0/0/0 unit 0
set interfaces fe-3/0/1 vlan-tagging
set interfaces fe-3/0/1 unit 0 description "inet6 configuration IP: 2000::c0a8:1bf5"
set interfaces fe-3/0/1 unit 0 vlan-id 100
set interfaces fe-3/0/1 unit 0 family inet address 17.10.1.1/24
set interfaces fe-3/0/1 unit 0 family inet6 address 2000::c0a8:1bf5/3

```

## Overview

One of the most useful features of regular expressions is the backreference. Backreferences provide a convenient way to identify a repeated character or substring within a string. Once you find the pattern, you can repeat it without writing it again. You refer to the previously captured pattern with just `\#` (where `#` is a numeral that indicates the number of times you want the pattern matched).

You can use backreferences to recall, or find, data and replace it with something else. In this way you can reformat large sets of data with a single replace command, thus saving you the time it would take to look for and replace the pattern manually.

## Configuration

### Configuring a Replacement Using a Backreference in the Command

#### Step-by-Step Procedure

To replace a pattern in a Junos OS configuration using a backreference:

- Use the **replace** command.

```

[edit]
user@host# replace pattern pattern1 with pattern2

```

In this case, we want to replace `:.1bf5` with `1bf5`.

```

[edit]
user@host# replace pattern "(.*):.1bf5" with "\11bf5"

```

Notice the backreference (`\1`), which indicates the pattern should be searched for and replaced only once.

### Results

Here is the resulting configuration:

```

[edit]
user@host# show interfaces
xe-0/0/0 {

```

```
 unit 0;
 }
 fe-3/0/1 {
 vlan-tagging;
 unit 0 {
 description "inet6 configuration. IP: 2000::c0a8:1bf5";
 vlan-id 100;
 family inet {
 address 17.10.1.1/24;
 }
 family inet6 {
 address 2000::c0a8:1bf5/3;
 }
 }
 }
}
```

In this example, the pattern 2000::c0a8:1bf5 is replaced with 2000::c0a8:1bf5 once.

**Related  
Documentation**

- [Example: Using Global Replace in a Junos OS Configuration—Replacing an Interface Name on page 208](#)
- [Using Global Replace in the Junos OS Configuration on page 205](#)

---

## Example: Using Global Replace in a Junos OS Configuration—Replacing an Interface Name

---

This example shows how to replace an interface name globally in a configuration by using the **replace** command.

Using the **replace** command can be a faster and better way to change a configuration. For example, a PIC might be moved to another slot in a router, which changes the interface name. With one command you can update the whole configuration. Or you might want to quickly extend the configuration with other similar configurations, for example, similar interfaces. By using a combination of the **copy** and **replace** commands, you can add to a configuration and then replace certain aspects of the newly copied configurations. The **replace** command works with regular expressions. Regular expressions are quick, flexible, and ubiquitous. You can fashion just about any pattern you might need to search for, and most programming languages support regular expressions.

- [Requirements on page 208](#)
- [Overview on page 209](#)
- [Configuration on page 209](#)

### Requirements

No special configuration beyond device initialization is required before configuring this example.

Before you begin, configure the following hierarchy on the router. To quickly configure this hierarchy, see “[CLI Quick Configuration](#)” on page 95 .

```
user@host# show interfaces
```



```

so-0/0/0 {
 dce;
}
user@host# show protocols
ospf {
 area 0.0.0.0 {
 interface so-0/0/0.0 {
 hello-interval 5;
 }
 }
}

```

## Overview

This example shows how to replace an interface name globally in a configuration by using the **replace** command. It is a simple example.

The previous configuration is the starting point for this configuration update. In the course of this example, you change the name of the initial interface throughout the configuration with one command.

## Configuration

**CLI Quick Configuration** To quickly configure the initial configuration for this example, copy the following commands, paste them into a text file, remove any line breaks and change any details necessary to match your network configuration, copy and paste these commands into the CLI at the **[edit]** hierarchy level, and then enter **commit** from configuration mode.:

```

set interfaces so-0/0/0 dce
set protocols ospf area 0.0.0.0 interface so-0/0/0.0 hello-interval 5

```

### Configuring an Interface Name Change

**Step-by-Step Procedure** To change an interface name:

1. Make sure that you are at the top of the configuration mode hierarchy.  

```
user@host# top
```
2. Replace so-0/0/0 with so-1/1/0 using the **replace** command, which uses the **pattern** keyword.  

```
user@host# replace pattern so-0/0/0 with so-1/1/0
```

### Results

After making the required changes, verify the configuration by using the **show interfaces** and **show protocols** configuration mode commands.

```

[edit]
user@host# show interfaces
so-1/1/0 {
 dce;
}
user@host# show protocols

```

```
ospf {
 area 0.0.0.0 {
 interface so-1/1/0.0 {
 hello-interval 5;
 }
 }
}
```

After you have confirmed that the configuration is correct, enter the **commit** command.

#### Related Documentation

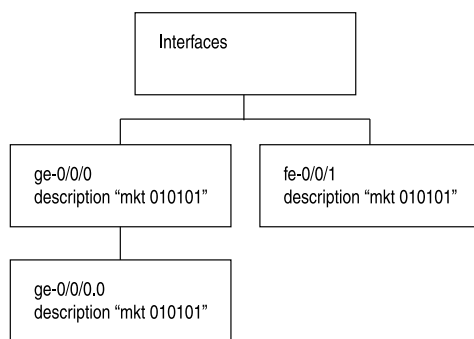
- [Example: Using Global Replace in a Junos OS Configuration—Using the upto Option on page 210](#)
- [Using Global Replace in the Junos OS Configuration on page 205](#)
- [Examples: Re-Using Configuration on page 76](#)

## Example: Using Global Replace in a Junos OS Configuration—Using the upto Option

Consider the hierarchy shown in [Figure 14 on page 210](#). The text string **010101** appears in three places: the description sections of **ge-0/0/0**, **ge-0/0/0.0**, and **fe-0/0/1**. These three instances are three objects. The following example shows how you can use the **upto** option to perform replacements in a JUNOS configuration:

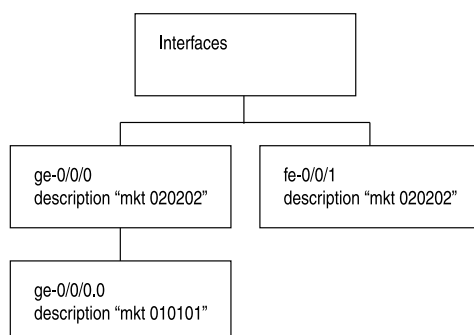
**Figure 14: Replacement by Object**

Current Configuration:



user@host > **replace pattern 01 with pattern 02 upto 2**

Resulting Configuration:



g017228

An **upto 2** option in the **replace** command converts **01** to **02** for two object instances. The objects under the main interfaces **ge-0/0/0** and **fe-0/0/1** will be replaced first (since these are siblings in the hierarchy level). Because of the **upto 2** restriction, the **replace** command replaces patterns in the first and second instance in the hierarchy (siblings), but not the third instance (child of the first instance).

```

user@host# show interfaces
ge-0/0/0 {
 description "mkt 010101"; #First instance in the hierarchy
 unit 0 {
 description "mkt 010101"; #Third instance in the hierarchy (child of the first
 instance)
 }
}
fe-0/0/1 {
 description "mkt 010101"; #second instance in the hierarchy (sibling of the first
 instance)
 unit 0 {
 family inet {
 address 200.200.20.2/24;
 }
 }
}
[edit]
user@host# replace pattern 01 with 02 upto 2
[edit]
user@host# commit
commit complete

[edit]
user@host# show interfaces
ge-0/0/0 {
 description "mkt 020202"; #First instance in the hierarchy
 unit 0 {
 description "mkt 010101"; #Third instance in the hierarchy (child of the first
 instance)
 }
}
fe-0/0/1 {
 description "mkt 020202"; #second instance in the hierarchy (sibling of the first
 instance)
 unit 0 {
 family inet {
 address 200.200.20.2/24;
 }
 }
}

```

**Related Documentation** • [Using Global Replace in the Junos OS Configuration on page 205](#)

## Using Regular Expressions to Delete Related Items from a Junos OS cConfiguration

The Junos OS command-line interface (CLI) enables you to delete related configuration items simultaneously, such as channelized interfaces or static routes, by using a single

command and regular expressions. Deleting a statement or an identifier effectively “unconfigures” the functionality associated with that statement or identifier, returning that functionality to its default condition.

You can only delete certain parts of the configuration where you normally put multiple items, for example, interfaces. However, you cannot delete “groups” of different items; for example:

```
user@host# show system services
ftp;
rlogin;
rsh;
ssh {
 root-login allow;
}
telnet;
[edit]
user@host# wildcard delete system services *
syntax error.
```

When you delete a statement, the statement and all its subordinate statements and identifiers are removed from the configuration.

To delete related configuration items, issue the **wildcard** configuration mode command with the **delete** option and specify the statement path, the items to be summarized with a regular expression, and the regular expression.

```
user@host# wildcard delete <statement-path> <identifier> <regular-expression>
```



**NOTE:** When you use the **wildcard** command to delete related configuration items, the regular expression must be the final statement.

If the Junos OS matches more than eight related items, the CLI displays only the first eight items.

### Deleting Interfaces from the Configuration

Delete multiple T1 interfaces in the range from t1-0/0/0:0 through t1-0/0/0:23:

```
user@host# wildcard delete interfaces t1-0/0/0:.*
matched: t1-0/0/0:0
matched: t1-0/0/0:1
matched: t1-0/0/0:2
Delete 3 objects? [yes,no] (no) no
```

**Deleting Routes from  
the Configuration**

Delete static routes in the range from 172.0.0.0 to 172.255.0.0:

```
user@host# wildcard delete routing-options static route 172.*
matched: 172.16.0.0/12
matched: 172.16.14.0/24
matched: 172.16.100.0/24
matched: 172.16.128.0/19
matched: 172.16.160.0/24
matched: 172.17.12.0/23
matched: 172.17.24.0/23
matched: 172.17.28.0/23
...
Delete 13 objects? [yes,no] (no)
```

**Related  
Documentation**

- [Disabling Inheritance of a Junos OS Configuration Group on page 222](#)



## CHAPTER 10

# Using Configuration Groups to Quickly Configure Devices

- [Understanding Junos OS Configuration Groups on page 216](#)
- [Creating the Junos OS Configuration Group on page 217](#)
- [Applying the Junos OS Configuration Group on page 219](#)
- [Example: Configuring and Applying Junos OS Configuration Groups on page 220](#)
- [Example: Creating and Applying Configuration Groups on a TX Matrix Router on page 221](#)
- [Disabling Inheritance of a Junos OS Configuration Group on page 222](#)
- [Using Wildcards with Configuration Groups on page 224](#)
- [Example: Configuring Sets of Statements with Configuration Groups on page 227](#)
- [Example: Configuring Interfaces Using Junos OS Configuration Groups on page 228](#)
- [Example: Configuring a Consistent IP Address for the Management Interface on page 230](#)
- [Example: Configuring Peer Entities on page 232](#)
- [Establishing Regional Configurations on page 234](#)
- [Configuring Wildcard Configuration Group Names on page 235](#)
- [Example: Referencing the Preset Statement From the Junos OS defaults Group on page 236](#)
- [Example: Viewing Default Statements That Have Been Applied to the Configuration on page 237](#)
- [Using Conditions to Apply Configuration Groups Overview on page 238](#)
- [Example: Configuring Conditions for Applying Configuration Groups on page 238](#)
- [Improving Commit Time When Using Configuration Groups on page 240](#)
- [Example: Improving Commit Time When Using Configuration Groups on page 241](#)
- [Using Junos OS Defaults Groups on page 242](#)
- [Setting Up Routing Engine Configuration Groups on page 244](#)

## Understanding Junos OS Configuration Groups

---

This topic provides an overview of the configuration groups feature and the inheritance model in Junos OS, and contains the following sections:

- [Configuration Groups Overview on page 216](#)
- [Inheritance Model on page 216](#)
- [Configuring Configuration Groups on page 216](#)

### Configuration Groups Overview

The configuration groups feature in Junos OS enables you to create a group containing configuration statements and to direct the inheritance of that group's statements in the rest of the configuration. The same group can be applied to different sections of the configuration, and different sections of one group's configuration statements can be inherited in different places in the configuration.

Configuration groups enable you to create smaller, more logically constructed configuration files, making it easier to configure and maintain Junos OS. For example, you can group statements that are repeated in many places in the configuration, such as when configuring interfaces, and thereby limit updates to just the group.

You can also use wildcards in a configuration group to allow configuration data to be inherited by any object that matches a wildcard expression.

The configuration group mechanism is separate from the grouping mechanisms used elsewhere in the configuration, such as BGP groups. Configuration groups provide a generic mechanism that can be used throughout the configuration but that are known only to the Junos OS CLI. The individual software processes that perform the actions directed by the configuration receive the expanded form of the configuration—they have no knowledge of configuration groups.

### Inheritance Model

Configuration groups use true inheritance, which involves a dynamic, ongoing relationship between the source of the configuration data and the target of that data. Data values changed in the configuration group are automatically inherited by the target. The target does not need to contain the inherited information, although the inherited values can be overridden in the target without affecting the source from which they were inherited.

This inheritance model allows you to see only the instance-specific information without seeing the inherited details. A command pipe in configuration mode allows you to display the inherited data.

### Configuring Configuration Groups

For areas of your configuration to inherit configuration statements, you must first put the statements into a configuration group and then apply that group to the levels in the configuration hierarchy that require the statements.



To configure configuration groups and inheritance, you can include the **groups** statement at the **[edit]** hierarchy level:

```
[edit]
groups {
 group-name {
 configuration-data;
 }
}
```

Include the **apply-groups [group-names]** statement anywhere in the configuration where the configuration statements contained in a configuration group are needed.

**Related Documentation**

- [Creating the Junos OS Configuration Group on page 217](#)

## Creating the Junos OS Configuration Group

To create a configuration group, include the **groups** statement at the **[edit]** hierarchy level:

```
[edit]
groups {
 group-name {
 configuration-data;
 }
 lccn-re0 {
 configuration-data;
 }
 lccn-re1 {
 configuration-data;
 }
}
```

**group-name** is the name of a configuration group. You can configure more than one configuration group by specifying multiple **group-name** statements. However, you cannot use the prefix **junos-** in a group name because it is reserved for use by Junos OS. Similarly, the configuration group **juniper-ais** is reserved exclusively for Juniper Advanced Insight Solutions (AIS)-related configuration. For more information on the **juniper-ais** configuration group, see the [Juniper Networks Advanced Insight Solutions Guide](#).

One reason for the naming restriction is a configuration group called **junos-defaults**. This preset configuration group is applied to the configuration automatically. You cannot modify or remove the **junos-defaults** configuration group. For more information about the Junos default configuration group, see [“Using Junos OS Defaults Groups” on page 242](#).

On routers that support multiple Routing Engines, you can also specify two special group names:

- **re0**—Configuration statements applied to the Routing Engine in slot 0.
- **re1**—Configuration statements applied to the Routing Engine in slot 1.

The configuration specified in group **re0** is only applied if the current Routing Engine is in slot 0; likewise, the configuration specified in group **re1** is only applied if the current Routing Engine is in slot 1. Therefore, both Routing Engines can use the same configuration file, each using only the configuration statements that apply to it. Each **re0** or **re1** group contains at a minimum the configuration for the hostname and the management interface (**fxp0**). If each Routing Engine uses a different management interface, the group also should contain the configuration for the backup router and static routes.

In addition, the TX Matrix router supports group names for the Routing Engines in each T640 router attached to the routing matrix. Providing special group names for all Routing Engines in the routing matrix allows you to configure the individual Routing Engines in each T640 router differently. Parameters that are not configured at the **[edit groups]** hierarchy level apply to all Routing Engines in the routing matrix.

**configuration-data** contains the configuration statements applied elsewhere in the configuration with the **apply-groups** statement. To have a configuration inherit the statements in a configuration group, include the **apply-groups** statement. For information about the **apply-groups** statement, see [“Applying the Junos OS Configuration Group” on page 219](#).

The group names for Routing Engines on the TX Matrix router have the following formats:

- **lccn-re0**—Configuration statements applied to the Routing Engine in slot 0 in a specified T640 router.
- **lccn-re1**—Configuration statements applied to the Routing Engine in slot 1 in a specified T640 router.

*n* identifies the T640 router and can be from 0 through 3. For example, to configure Routing Engine 1 properties for **lcc3**, you include statements at the **[edit groups lcc3-re1]** hierarchy level. For information about the TX Matrix router and routing matrix, see the *Administration Guide for Security Devices*.



**NOTE:** The management Ethernet interface used for the TX Matrix Plus router, T1600 or T4000 routers in a routing matrix, and PTX Series Packet Transport Routers, is **em0**. Junos OS automatically creates the router's management Ethernet interface, **em0**.

---

**Related  
Documentation**

- [Applying the Junos OS Configuration Group on page 219](#)
- [Using Junos OS Defaults Groups on page 242](#)
- [Understanding Junos OS Configuration Groups on page 216](#)
- [Disabling Inheritance of a Junos OS Configuration Group on page 222](#)
- [Using Wildcards with Configuration Groups on page 224](#)
- [Example: Configuring Sets of Statements with Configuration Groups on page 227](#)

## Applying the Junos OS Configuration Group

To have the Junos OS configuration inherit the statements from a configuration group, include the **apply-groups** statement:

```
apply-groups [group-names];
```

If you specify more than one group name, list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.

For routers that support multiple Routing Engines, you can specify **re0** and **re1** group names. The configuration specified in group **re0** is only applied if the current Routing Engine is in slot 0; likewise, the configuration specified in group **re1** is only applied if the current Routing Engine is in slot 1. Therefore, both Routing Engines can use the same configuration file, each using only the configuration statements that apply to it. Each **re0** or **re1** group contains at a minimum the configuration for the hostname and the management interface (**fxp0**). If each Routing Engine uses a different management interface, the group also should contain the configuration for the backup router and static routes.



**NOTE:** The management Ethernet interface used for the TX Matrix Plus router, T1600 routers in a routing matrix, and PTX Series Packet Transport Switches, is **em0**.

You can include only one **apply-groups** statement at each specific level of the configuration hierarchy. The **apply-groups** statement at a specific hierarchy level lists the configuration groups to be added to the containing statement's list of configuration groups.

Values specified at the specific hierarchy level override values inherited from the configuration group.

Groups listed in nested **apply-groups** statements take priority over groups in outer statements. In the following example, the BGP neighbor **10.0.0.1** inherits configuration data from group **one** first, then from groups **two** and **three**. Configuration data in group **one** overrides data in any other group. Data from group **ten** is used only if a statement is not contained in any other group.

```
apply-groups [eight nine ten];
protocols {
 apply-groups seven;
 bgp {
 apply-groups [five six];
 group some-bgp-group {
 apply-groups four;
 neighbor 10.0.0.1 {
 apply-groups [one two three];
 }
 }
 }
}
```

When you configure a group defined for the root level—that is, in the default logical system—you cannot successfully apply that group to a nondefault logical system under the `[edit logical-systems logical-system-name]` hierarchy level. Although the router accepts the commit if you apply the group, the configuration group does not take effect for the nondefault logical system. You can instead create an additional configuration group at the root level and apply it within the logical system. Alternatively, you can modify the original group so that it includes configuration for both the default and nondefault logical system hierarchy levels.

**Related Documentation**

- [Example: Configuring and Applying Junos OS Configuration Groups on page 220](#)
- [Disabling Inheritance of a Junos OS Configuration Group on page 222](#)
- [Creating the Junos OS Configuration Group on page 217](#)
- [Using Wildcards with Configuration Groups on page 224](#)
- [Example: Configuring Sets of Statements with Configuration Groups on page 227](#)

---

## Example: Configuring and Applying Junos OS Configuration Groups

---

In this example, the SNMP configuration is divided between the group **basic** and the normal configuration hierarchy.

There are a number of advantages to placing the system-specific configuration (SNMP contact) into a configuration group and thus separating it from the normal configuration hierarchy—the user can replace (using the **load replace** command) either section without discarding data from the other.

In addition, setting a contact for a specific box is now possible because the group data would be hidden by the router-specific data.

```
[edit]
groups {
 basic { # User-defined group name
 snmp { # This group contains some SNMP data
 contact "My Engineering Group";
 community BasicAccess {
 authorization read-only;
 }
 }
 }
}
apply-groups basic; # Enable inheritance from group "basic"
snmp { # Some normal (non-group) configuration
 location "West of Nowhere";
}
```

This configuration is equivalent to the following:

```
[edit]
snmp {
 location "West of Nowhere";
 contact "My Engineering Group";
}
```

```

community BasicAccess {
 authorization read-only;
}

```

For information about how to disable inheritance of a configuration group, see [“Disabling Inheritance of a Junos OS Configuration Group” on page 222](#).

#### Related Documentation

- [Example: Creating and Applying Configuration Groups on a TX Matrix Router on page 221](#)
- [Example: Configuring Interfaces Using Junos OS Configuration Groups on page 228](#)
- [Example: Configuring Peer Entities on page 232](#)
- [Example: Referencing the Preset Statement From the Junos OS defaults Group on page 236](#)
- [Example: Viewing Default Statements That Have Been Applied to the Configuration on page 237](#)
- [Example: Configuring Sets of Statements with Configuration Groups on page 227](#)
- [Example: Configuring a Consistent IP Address for the Management Interface on page 230](#)
- [Creating the Junos OS Configuration Group on page 217](#)

## Example: Creating and Applying Configuration Groups on a TX Matrix Router

The following example shows how to configure and apply configuration groups on a TX Matrix Router:

```

[edit]
groups {
 re0 { # Routing Engine 0 on TX Matrix router
 system {
 host-name hostname;
 backup-router ip-address;
 }
 interfaces {
 fxp0 {
 unit 0 {
 family inet {
 address ip-address;
 }
 }
 }
 }
 }
 re1 { # Routing Engine 1 on TX Matrix router
 system {
 host-name hostname;
 backup-router ip-address;
 }
 interfaces {
 fxp0 {
 unit 0 {

```

```
 family inet {
 address ip-address;
 }
 }
}
}
lcc0-re0 { # Routing Engine 0 on T640 router numbered 0
 system {
 host-name hostname;
 backup-router ip-address;
 }
 interfaces {
 fxp0 {
 unit 0 {
 family inet {
 address ip-address;
 }
 }
 }
 }
}
lcc0-re1 { # Routing Engine 1 on T640 router numbered 0
 system {
 host-name hostname;
 backup-router ip-address;
 }
 interfaces {
 fxp0 {
 unit 0 {
 family inet {
 address ip-address;
 }
 }
 }
 }
}
}
apply-groups [re0 re1 lcc0-re0 lcc0-re1];
```

- Related Documentation**
- [Example: Configuring and Applying Junos OS Configuration Groups on page 220](#)
  - [Creating the Junos OS Configuration Group on page 217](#)

---

## Disabling Inheritance of a Junos OS Configuration Group

To disable inheritance of a configuration group at any level except the top level of the hierarchy, include the **apply-groups-except** statement:

```
apply-groups-except [group-names];
```

This statement is useful when you use the **apply-group** statement at a specific hierarchy level but also want to override the values inherited from the configuration group for a specific parameter.

**Example: Disabling Inheritance on Interface so-1/1/0**

In the following example, the **apply-groups** statement is applied globally at the interfaces level. The **apply-groups-except** statement is also applied at interface **so-1/1/0** so that it uses the default values for the **hold-time** and **link-mode** statements.

```
[edit]
groups { # "groups" is a top-level statement
 global { # User-defined group name
 interfaces {
 <*> {
 hold-time down 640;
 link-mode full-duplex;
 }
 }
 }
}
apply-groups global;
interfaces {
 so-1/1/0 {
 apply-groups-except global; # Disables inheritance from group "global"
 # so-1/1/0 uses default value for "hold-time"
 # and "link-mode"
 }
}
```

For information about applying a configuration group, see [“Applying the Junos OS Configuration Group” on page 219](#).

Configuration groups can add some confusion regarding the actual values used by the router, because configuration data can be inherited from configuration groups. To view the actual values used by the router, use the **display inheritance** command after the pipe (|) in a **show** command. This command displays the inherited statements at the level at which they are inherited and the group from which they have been inherited.

```
[edit]
user@host# show | display inheritance
snmp {
 location "West of Nowhere";
 ##
 ## 'My Engineering Group' was inherited from group 'basic'
 ##
 contact "My Engineering Group";
 ##
 ## 'BasicAccess' was inherited from group 'basic'
 ##
 community BasicAccess {
 ##
 ## 'read-only' was inherited from group 'basic'
 ##
 authorization read-only;
 }
}
```

To display the expanded configuration (the configuration, including the inherited statements) without the ## lines, use the **except** command after the pipe in a **show** command:

```
[edit]
user@host# show | display inheritance | except ##
snmp {
 location "West of Nowhere";
 contact "My Engineering Group";
 community BasicAccess {
 authorization read-only;
 }
}
```



**NOTE:** Using the `display inheritance | except ##` option removes all the lines with `##`. Therefore, you might also not be able to view information about passwords and other important data where `##` is used. To view the complete configuration details with all the information without just the comments marked with `##`, use the `no-comments` option with the `display inheritance` command:

```
[edit]
user@host# show | display inheritance no-comments
snmp {
 location "West of Nowhere";
 contact "My Engineering Group";
 community BasicAccess {
 authorization read-only;
 }
}
```

**Related  
Documentation**

- [Applying the Junos OS Configuration Group on page 219](#)
- [Understanding Junos OS Configuration Groups on page 216](#)

---

## Using Wildcards with Configuration Groups

---

You can use wildcards to identify names and allow one statement to provide data for a variety of statements. For example, grouping the configuration of the **sonet-options** statement over all SONET/SDH interfaces or the dead interval for OSPF over all Asynchronous Transfer Mode (ATM) interfaces simplifies configuration files and eases their maintenance.

Using wildcards in normal configuration data is done in a style that is consistent with that used with traditional UNIX shell wildcards. In this style, you can use the following metacharacters:

- Asterisk ( `*` )—Matches any string of characters.
- Question mark ( `?` )—Matches any single character.
- Open bracket ( `[` )—Introduces a character class.
- Close bracket ( `]` )—Indicates the end of a character class. If the close bracket is missing, the open bracket matches a `[` rather than introduce a character class.



- A character class matches any of the characters between the square brackets. Within a configuration group, an interface name that includes a character class must be enclosed in quotation marks.
- Hyphen ( - )—Specifies a range of characters.
- Exclamation point ( ! )—The character class can be complemented by making an exclamation point the first character of the character class. To include a close bracket ( **]** ) in a character class, make it the first character listed (after the **!**, if any). To include a minus sign, make it the first or last character listed.



**NOTE:** If used inside the **groups** hierarchy, an identifier name cannot start with **<** unless you are defining a wildcard statement, in which case the wildcard statement must have a closing **>**.

Wildcarding in configuration groups follows the same rules, but **<** and **>** have a special meaning when used under the **groups** hierarchy. In the **groups** hierarchy, any term using a wildcard pattern must be enclosed in angle brackets *<pattern>* to differentiate it from other wildcarding in the configuration file.

```
[edit]
groups {
 sonet-default {
 interfaces {
 <so-*> {
 sonet-options {
 payload-scrambler;
 rfc-2615;
 }
 }
 }
 }
}
```

Wildcard expressions match (and provide configuration data for) existing statements in the configuration that match their expression only. In the previous example, the expression **<so-\*>** passes its **sonet-options** statement to any interface that matches the expression **so-\***.

The following example shows how to specify a range of interfaces:

```
[edit]
groups {
 gigabit-ethernet-interfaces {
 interfaces {
 "<ge-1/2/[5-8]>" {
 description "These interfaces reserved for Customer ABC";
 }
 }
 }
}
```

Angle brackets allow you to pass normal wildcarding through without modification. In any matching within the configuration, whether it is done with or without wildcards, the first item encountered in the configuration that matches is used. In the following example, data from the wildcarded BGP groups is inherited in the order in which the groups are listed. The preference value from `<*a*>` overrides the preference in `<*b*>`, just as the `p` value from `<*c*>` overrides the one from `<*d*>`. Data values from any of these groups override the data values from `abcd`.

```
[edit]
user@host# show
groups {
 one {
 protocols {
 bgp {
 group <*a*> {
 preference 1;
 }
 group <*b*> {
 preference 2;
 }
 group <*c*> {
 out-delay 3;
 }
 group <*d*> {
 out-delay 4;
 }
 group abcd {
 preference 10;
 hold-time 10;
 out-delay 10;
 }
 }
 }
 }
}
protocols {
 bgp {
 group abcd {
 apply-groups one;
 }
 }
}
[edit]
user@host# show | display inheritance
protocols {
 bgp {
 group abcd {
 ##
 ## '1' was inherited from group 'one'
 ##
 preference 1;
 ##
 ## '10' was inherited from group 'one'
 ##
 hold-time 10;
```

```

 ##
 ## '3' was inherited from group 'one'
 ##
 out-delay 3;
 }
}

```

#### Related Documentation

- [Configuring Wildcard Configuration Group Names on page 235](#)
- [Applying the Junos OS Configuration Group on page 219](#)
- [Creating the Junos OS Configuration Group on page 217](#)
- [Understanding Junos OS Configuration Groups on page 216](#)

## Example: Configuring Sets of Statements with Configuration Groups

When sets of statements exist in configuration groups, all values are inherited. For example:

```

[edit]
user@host# show
groups {
 basic {
 snmp {
 interface so-1/1/1.0;
 }
 }
}
apply-groups basic;
snmp {
 interface so-0/0/0.0;
}
[edit]
user@host# show | display inheritance
snmp {
 ##
 ## 'so-1/1/1.0' was inherited from group 'basic'
 ##
 interface [so-0/0/0.0 so-1/1/1.0];
}

```

For sets that are not displayed within brackets, all values are also inherited. For example:

```

[edit]
user@host# show
groups {
 worldwide {
 system {
 name-server {
 10.0.0.100;
 10.0.0.200;
 }
 }
 }
}

```

```
 }
 }
 apply-groups worldwide;
 system {
 name-server {
 10.0.0.1;
 10.0.0.2;
 }
 }
[edit]
user@host# show | display inheritance
system {
 name-server {
 ##
 ## '10.0.0.100' was inherited from group 'worldwide'
 ##
 10.0.0.100;
 ##
 ## '10.0.0.200' was inherited from group 'worldwide'
 ##
 10.0.0.200;
 10.0.0.1;
 10.0.0.2;
 }
}
```

**Related  
Documentation**

- [Understanding Junos OS Configuration Groups on page 216](#)
- [Creating the Junos OS Configuration Group on page 217](#)
- [Applying the Junos OS Configuration Group on page 219](#)

---

## Example: Configuring Interfaces Using Junos OS Configuration Groups

---

You can use configuration groups to separate the common interface media parameters from the interface-specific addressing information. The following example places configuration data for ATM interfaces into a group called **atm-options**:

```
[edit]
user@host# show
groups {
 atm-options {
 interfaces {
 <at-*> {
 atm-options {
 vpi 0 maximum-vcs 1024;
 }
 unit <*> {
 encapsulation atm-snap;
 point-to-point;
 family iso;
 }
 }
 }
 }
}
```

```

}
apply-groups atm-options;
interfaces {
 at-0/0/0 {
 unit 100 {
 vci 0.100;
 family inet {
 address 10.0.0.100/30;
 }
 }
 unit 200 {
 vci 0.200;
 family inet {
 address 10.0.0.200/30;
 }
 }
 }
}
[edit]
user@host# show | display inheritance
interfaces {
 at-0/0/0 {
 ##
 ## "atm-options" was inherited from group "atm-options"
 ##
 atm-options {
 ##
 ## "1024" was inherited from group "atm-options"
 ##
 vpi 0 maximum-vcs 1024;
 }
 unit 100 {
 ##
 ## "atm-snap" was inherited from group "atm-options"
 ##
 encapsulation atm-snap;
 ##
 ## "point-to-point" was inherited from group "atm-options"
 ##
 point-to-point;
 vci 0.100;
 family inet {
 address 10.0.0.100/30;
 }
 ##
 ## "iso" was inherited from group "atm-options"
 ##
 family iso;
 }
 unit 200 {
 ##
 ## "atm-snap" was inherited from group "atm-options"
 ##
 encapsulation atm-snap;
 ##
 ## "point-to-point" was inherited from group "atm-options"

```

```
##
point-to-point;
vci 0.200;
family inet {
 address 10.0.0.200/30;
}
##
"iso" was inherited from group "atm-options"
##
family iso;
}
}
[edit]
user@host# show | display inheritance | except ##
interfaces {
 at-0/0/0 {
 atm-options {
 vpi 0 maximum-vcs 1024;
 }
 unit 100 {
 encapsulation atm-snap;
 point-to-point;
 vci 0.100;
 family inet {
 address 10.0.0.100/30;
 }
 family iso;
 }
 unit 200 {
 encapsulation atm-snap;
 point-to-point;
 vci 0.200;
 family inet {
 address 10.0.0.200/30;
 }
 family iso;
 }
 }
}
```

**Related  
Documentation**

- [Understanding Junos OS Configuration Groups on page 216](#)
- [Creating the Junos OS Configuration Group on page 217](#)
- [Interface Naming Conventions Used in the Junos OS Operational Commands on page 172](#)
- [Example: Configuring a Consistent IP Address for the Management Interface on page 230](#)

---

## Example: Configuring a Consistent IP Address for the Management Interface

On routers with multiple Routing Engines, each Routing Engine is configured with a separate IP address for the management interface (**fxp0**). To access the master Routing Engine, you must know which Routing Engine is active and use the appropriate IP address.

Optionally, for consistent access to the master Routing Engine, you can configure an additional IP address and use this address for the management interface regardless of which Routing Engine is active. This additional IP address is active only on the management interface for the master Routing Engine. During switchover, the address moves to the new master Routing Engine.

In the following example, address **10.17.40.131** is configured for both Routing Engines and includes a **master-only** statement. With this configuration, the **10.17.40.131** address is active only on the master Routing Engine. The address remains consistent regardless of which Routing Engine is active. Address **10.17.40.132** is assigned to **fxp0** on **re0**, and **10.17.40.133** is assigned to **fxp0** on **re1**.

```
[edit groups re0 interfaces fxp0]
unit 0 {
 family inet {
 address 10.17.40.131/25 {
 master-only;
 }
 address 10.17.40.132/25;
 }
}
[edit groups re1 interfaces fxp0]
unit 0 {
 family inet {
 address 10.17.40.131/25 {
 master-only;
 }
 address 10.17.40.133/25;
 }
}
```

This feature is available on all routers that include dual Routing Engines. On a routing matrix composed of the TX Matrix router, this feature is applicable to the switch-card chassis (SCC) only. Likewise, on a routing matrix composed of a TX Matrix Plus router, this feature is applicable to the switch-fabric chassis (SFC) only.



#### NOTE:

- If you configure the same IP address for a management interface or internal interface such as **fxp0** and an external physical interface such as **ge-0/0/1**, when graceful Routing Engine switchover (GRES) is enabled, the CLI displays an appropriate commit error message that identical addresses have been found on the private and public interfaces. In such cases, you must assign unique IP addresses for the two interfaces that have duplicate addresses.
- The management Ethernet interface used for the TX Matrix Plus router, T1600 routers in a routing matrix, and PTX Series Packet Transport Routers, is **em0**. Junos OS automatically creates the router's management Ethernet interface, **em0**.

**Related Documentation**

- [Understanding Junos OS Configuration Groups on page 216](#)
- [Creating the Junos OS Configuration Group on page 217](#)
- [Example: Configuring Interfaces Using Junos OS Configuration Groups on page 228](#)

---

## Example: Configuring Peer Entities

In this example, we create a group **some-isp** that contains configuration data relating to another Internet service provider (ISP). We can then insert **apply-group** statements at any point to allow any location in the configuration hierarchy to inherit this data.

```
[edit]
user@host# show
groups {
 some-isp {
 interfaces {
 <xe-*> {
 gigether-options {
 flow-control;
 }
 }
 }
 protocols {
 bgp {
 group <*> {
 neighbor <*> {
 remove-private;
 }
 }
 }
 pim {
 interface <*> {
 version 1;
 }
 }
 }
 }
}
interfaces {
 xe-0/0/0 {
 apply-groups some-isp;
 unit 0 {
 family inet {
 address 10.0.0.1/24;
 }
 }
 }
}
protocols {
 bgp {
 group main {
 neighbor 10.254.0.1 {
 apply-groups some-isp;
 }
 }
 }
}
```



```

 }
 }
}
pim {
 interface xe-0/0/0.0 {
 apply-groups some-isp;
 }
}
}
[edit]
user@host# show | display inheritance
interfaces {
 xe-0/0/0 {
 ##
 ## "igether-options" was inherited from group "some-isp"
 ##
 igether-options {
 ##
 ## "flow-control" was inherited from group "some-isp"
 ##
 flow-control;
 }
 unit 0 {
 family inet {
 address 10.0.0.1/24;
 }
 }
 }
}
protocols {
 bgp {
 group main {
 neighbor 10.254.0.1 {
 ##
 ## "remove-private" was inherited from group "some-isp"
 ##
 remove-private;
 }
 }
 }
}
pim {
 interface xe-0/0/0.0 {
 ##
 ## "1" was inherited from group "some-isp"
 ##
 version 1;
 }
}
}

```

#### Related Documentation

- [Understanding Junos OS Configuration Groups on page 216](#)
- [Creating the Junos OS Configuration Group on page 217](#)
- [Establishing Regional Configurations on page 234](#)

## Establishing Regional Configurations

---

In this example, one group is populated with configuration data that is standard throughout the company, while another group contains regional deviations from this standard:

```
[edit]
user@host# show
groups {
 standard {
 interfaces {
 <t3-*> {
 t3-options {
 compatibility-mode larscom subrate 10;
 idle-cycle-flag ones;
 }
 }
 }
 }
 northwest {
 interfaces {
 <t3-*> {
 t3-options {
 long-buildout;
 compatibility-mode kentrox;
 }
 }
 }
 }
}
apply-groups standard;
interfaces {
 t3-0/0/0 {
 apply-groups northwest;
 }
}
[edit]
user@host# show | display inheritance
interfaces {
 t3-0/0/0 {
 ##
 ## "t3-options" was inherited from group "northwest"
 ##
 t3-options {
 ##
 ## "long-buildout" was inherited from group "northwest"
 ##
 long-buildout;
 ##
 ## "kentrox" was inherited from group "northwest"
 ##
 compatibility-mode kentrox;
 ##
 ## "ones" was inherited from group "standard"
```

```

 ##
 idle-cycle-flag ones;
 }
}
}

```

- Related Documentation**
- [Understanding Junos OS Configuration Groups on page 216](#)
  - [Example: Configuring Peer Entities on page 232](#)

## Configuring Wildcard Configuration Group Names

Wildcards are configuration group names that use special characters to create a pattern that can be applied to multiple statements. Wildcards are useful for copying one set of configuration options to a large number of different configuration groups. It is important to set up your wildcard name properly to ensure that the wildcard configuration options get copied to the appropriate configuration groups.

In this example, you configure different values for the `<*-major>` and `<*-minor>` wildcard groups under the `label-switched-path` statement. The asterisk (\*) character represents a section of the wildcard name that can match any string of characters. For example the configuration options under `label-switched-path <*-major>` are passed onto `label-switched-path metro-major` and any other `label-switched-path` configuration group containing `-major` in its name.

```

[edit]
user@host# show
groups {
 mpls-conf {
 protocols {
 mpls {
 label-switched-path <*-major> {
 retry-timer 5;
 bandwidth 155m;
 optimize-timer 60;
 }
 label-switched-path <*-minor> {
 retry-timer 15;
 bandwidth 64k;
 optimize-timer 120;
 }
 }
 }
 }
}
apply-groups mpls-conf;
protocols {
 mpls {
 label-switched-path metro-major {
 to 10.0.0.10;
 }
 label-switched-path remote-minor {
 to 10.0.0.20;
 }
 }
}

```

```
 }
 }
}
[edit]
user@host# show | display inheritance
protocols {
 mpls {
 label-switched-path metro-major {
 to 10.0.0.10;
 ##
 ## "5" was inherited from group "mpls-conf"
 ##
 retry-timer 5;
 ## "155m" was inherited from group "mpls-conf"
 ##
 bandwidth 155m;
 ##
 ## "60" was inherited from group "mpls-conf"
 ##
 optimize-timer 60;
 }
 label-switched-path remote-minor {
 to 10.0.0.20;
 ##
 ## "15" was inherited from group "mpls-conf"
 ##
 retry-timer 15;
 ##
 ## "64k" was inherited from group "mpls-conf"
 ##
 bandwidth 64k;
 ##
 ## "120" was inherited from group "mpls-conf"
 ##
 optimize-timer 120;
 }
 }
}
```

**Related Documentation**

- [Using Wildcards with Configuration Groups on page 224](#)

---

## Example: Referencing the Preset Statement From the Junos OS defaults Group

The following example is a preset statement from the Junos defaults group that is available for FTP in a stateful firewall:

```
[edit]
groups {
 junos-defaults {
 applications {
 application junos-ftp {# Use FTP default configuration
 application-protocol ftp;
 protocol tcp;
 destination-port 21;
 }
 }
}
```

```

 }
 }
}

```

To reference a preset Junos default statement from the Junos defaults group, include the **junos-default-name** statement at the applicable hierarchy level. For example, to reference the Junos default statement for FTP in a stateful firewall, include the **junos-ftp** statement at the **[edit services stateful-firewall rule my-rule term my-term from applications]** hierarchy level:

```

[edit]
services {
 stateful-firewall {
 rule my-rule {
 term my-term {
 from {
 applications junos-ftp; #Reference predefined statement, junos-ftp
 }
 }
 }
 }
}

```

#### Related Documentation

- [Example: Viewing Default Statements That Have Been Applied to the Configuration on page 237](#)
- [Using Junos OS Defaults Groups on page 242](#)
- [Understanding Junos OS Configuration Groups on page 216](#)
- [Creating the Junos OS Configuration Group on page 217](#)

## Example: Viewing Default Statements That Have Been Applied to the Configuration

To view the Junos defaults that have been applied to the configuration, issue the **show | display inheritance defaults** command. For example, to view the inherited Junos defaults at the **[edit system ports]** hierarchy level:

```

user@host# show system ports | display inheritance defaults
'console' was inherited from group 'junos-defaults'
'vt100' was inherited from group 'junos-defaults'
console type vt100;

```

If you choose not to use existing Junos default statements, you can create your own configuration groups manually.

To view the complete configuration information without the comments marked with **##**, use the **no-comments** option with the **display inheritance** command.

#### Related Documentation

- [Creating the Junos OS Configuration Group on page 217](#)
- [Configuring Configuration Groups on page 216](#)

## Using Conditions to Apply Configuration Groups Overview

---

You can use the **when** statement at the **[edit groups group-name]** hierarchy level to define conditions under which a configuration group should be applied.

You can configure a group to be applied based on the type of chassis, model, or Routing Engine, virtual chassis member, cluster node, and start and optional end time of day or date.

For example, you could use the **when** statement to create a generic configuration group for each type of node and then apply the configuration based on certain node properties, such as chassis or model.

### Related Documentation

- [Example: Configuring Conditions for Applying Configuration Groups on page 238](#)

## Example: Configuring Conditions for Applying Configuration Groups

---

This example shows how to configure conditions under which a specified configuration group is to be applied.

- [Requirements on page 238](#)
- [Overview on page 238](#)
- [Configuration on page 239](#)

### Requirements

No special configuration beyond device initialization is required before you configure this example.

### Overview

You can configure your group configuration data at the **[edit groups group-name]** hierarchy level, then use the **when** statement to have the group applied based on conditions including: type of chassis, model, routing-engine, virtual chassis member, cluster node, and start and optional end time of day or date.

If you specify multiple conditions in a single configuration group, all conditions must be met before the configuration group is applied.

You can specify the start time or the time duration for the configuration group to be applied. If only the start time is specified, the configuration group is applied at the specified time and it remains in effect until the time is changed. If the end time is specified, then on each day, the applied configuration group is started and stopped at the specified times.

This example sets conditions in a configuration group, **test1**, such that this group is applied only when all of the following conditions are met: the router is a model MX240 router with chassis type LCC0, with a Routing Engine operating as RE0, is member0 of the virtual

chassis on node0, and the configuration group will only be in effect from 9:00 a.m. until 5:00 p.m. each day.

## Configuration

**CLI Quick Configuration** To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

```
set groups test1 when model mx240
set groups test1 when chassis lcc0
set groups test1 when routing-engine re0
set groups test1 when member member0
set groups test1 when node node0
set groups test1 when time 9 to 5
```

**Step-by-Step Procedure** To configure conditions for configuration group **test1**:

1. Set the condition that identifies the model MX240 router.  

```
[edit groups test1 when]
user@host# set model mx240
```
2. Set the condition that identifies the chassis type as LCC0.  

```
[edit groups test1 when]
user@host# set chassis lcc0
```
3. Set the condition that identifies the Routing Engine operating as RE0.  

```
[edit groups test1 when]
user@host# set routing-engine re0
```
4. Set the condition that identifies the virtual chassis **member0**.  

```
[edit groups test1 when]
user@host# set member member0
```
5. Set the condition that identifies the cluster **node0**.  

```
[edit groups test1 when]
user@host# set node node0
```
6. Set the condition that applies the group only between the hours of 9:00 a.m. and 5:00 p.m. daily.  

```
[edit groups test1 when]
user@host# set time 9 to 5
```



**NOTE:** The syntax for specifying the time is: `time <start-time> [to <end-time>]` using the time format `yyyy-mm-dd.hh:mm`, `hh:mm`, or `hh`.

7. Commit the configuration.  

```
user@host# commit
```

**Results** From configuration mode, confirm your configuration by entering the **show groups test1** command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show groups test1
when {
 time 9 to 5;
 chassis lcc0;
 model mx240;
 routing-engine re0;
 member member0;
 node node0;
}
```

---

### Verification

Confirm that the configuration is working properly.

- [Checking Group Inheritance with Conditional Data on page 240](#)

#### *Checking Group Inheritance with Conditional Data*

**Purpose** Verify that conditional data from a configuration group is inherited when applied.

**Action** The **show | display inheritance** operational command can be issued with the **when** data to display the conditional inheritance. Using this example, you could issue one of these commands to determine that the conditional data was inherited:

```
user@host> show | display inheritance when model mx240
user@host> show | display inheritance when chassis lcc0
user@host> show | display inheritance when routing-engine re0
user@host> show | display inheritance when member member0
user@host> show | display inheritance when node node0
user@host> show | display inheritance when time 9 to 5
```

**Related  
Documentation**

- [Understanding Junos OS Configuration Groups on page 216](#)
- [Creating the Junos OS Configuration Group on page 217](#)
- [Applying the Junos OS Configuration Group on page 219](#)
- [Using Conditions to Apply Configuration Groups Overview on page 238](#)

---

## Improving Commit Time When Using Configuration Groups

Configuration groups are used for applying configurations across other hierarchies without re-entering configuration data. Some configuration groups specify every configuration detail. Other configuration groups make use of wildcards to configure ranges of data, without detailing each configuration line. Some configurations have an inheritance path that includes a long string of configurations to be applied.

When a configuration that uses configuration groups is committed, the commit process expands and reads all of the configuration data of the group into memory in order to



apply the configurations as intended. The commit performance can be negatively impacted if many configuration groups are being applied, especially if the configuration groups use wildcards extensively.

If your system uses many configuration groups that use wildcards, you can configure the **persist-groups-inheritance** statement at the **[edit system commit]** hierarchy level to improve commit time performance.

Using this option allows the system to build the inheritance path for each configuration group inside the database, instead of in the process memory. This can improve commit time performance. However, it can also increase the database size by up to 22 percent.

**Related  
Documentation**

- [Example: Improving Commit Time When Using Configuration Groups on page 241](#)
- *persist-groups-inheritance*

---

## Example: Improving Commit Time When Using Configuration Groups

---

This example shows how to use the **persist-groups-inheritance** configuration statement to improve commit time performance when committing configurations that make use of many configuration groups that are created using wildcards.

- [Requirements on page 241](#)
- [Overview on page 241](#)
- [Configuration on page 242](#)
- [Verification on page 242](#)

### Requirements

This example uses the following hardware and software components:

- One Juniper Networks M Series, MX Series, or T Series router that uses a number of configuration groups created with wildcards.
- Junos OS Release 13.2 or later.

### Overview

When committing a configuration that uses configuration groups, at the time of commit, all of the inheritance paths of the configuration groups must be fully expanded into memory to apply the configurations as intended. This can negatively impact commit performance if there are many configuration groups and they are configured using wildcards.

To improve commit performance, you can configure **persist-groups-inheritance** at the **[edit system commit]** hierarchy level. Configuring this option causes the configuration groups to be expanded into the database instead of into the process memory at commit time.

## Configuration

### Configuring Persist Groups Inheritance

---

#### Step-by-Step Procedure

To configure **persist-groups-inheritance**:

1. Set the **persist-groups-inheritance** option.  

```
[edit system commit]
user@host# set persist-groups-inheritance
```
2. Commit the configuration.  

```
[edit system commit]
user@host# commit
```

## Verification

### Verifying the Configuration

---

**Purpose** Verify that **persist-groups-inheritance** is configured.

**Action** To confirm the configuration, use the **show system commit** command.

```
[edit]
user@host# show system commit
persist-groups-inheritance
```

**Related Documentation**

- [Improving Commit Time When Using Configuration Groups on page 240](#)
- *persist-groups-inheritance*

## Using Junos OS Defaults Groups

---

Junos OS provides a hidden and immutable configuration group called **junos-defaults** that is automatically applied to the configuration of your router. The **junos-defaults** group contains preconfigured statements that contain predefined values for common applications. Some of the statements must be referenced to take effect, such as definitions for applications (for example, FTP or telnet settings). Other statements are applied automatically, such as terminal settings.



**NOTE:** Many identifiers included in the **junos-defaults** configuration group begin with the name **junos-**. Because identifiers beginning with the name **junos-** are reserved for use by Juniper Networks, you cannot define any configuration objects using this name.

You cannot include **junos-defaults** as a configuration group name in an **apply-groups** statement.

---

To view the full set of available preset statements from the Junos defaults group, issue the **show groups junos-defaults** configuration mode command at the top level of the configuration. The following example displays a partial list of Junos defaults groups:

```
user@host# show groups junos-defaults
Make vt100 the default for the console port
system {
 ports {
 console type vt100;
 }
}
applications {
 # File Transfer Protocol
 application junos-ftp {
 application-protocol ftp;
 protocol tcp;
 destination-port 21;
 }
 # Trivial File Transfer Protocol
 application junos-tftp {
 application-protocol tftp;
 protocol udp;
 destination-port 69;
 }
 # RPC port mapper on TCP
 application junos-rpc-portmap-tcp {
 application-protocol rpc-portmap;
 protocol tcp;
 destination-port 111;
 }
 # RPC port mapper on UDP
}
```

To reference statements available from the **junos-defaults** group, include the selected **junos- *default-name*** statement at the applicable hierarchy level.

#### Related Documentation

- [Creating the Junos OS Configuration Group on page 217](#)
- [Example: Referencing the Preset Statement From the Junos OS defaults Group on page 236](#)
- [Example: Viewing Default Statements That Have Been Applied to the Configuration on page 237](#)

## Setting Up Routing Engine Configuration Groups

In a router with two Routing Engines, one configuration should be shared between both Routing Engines. This ensures that both Routing Engine configurations are identical. Within this configuration, create two Routing Engine groups, one for each Routing Engine. Within these groups, you specify the Routing Engine–specific parameters.

For more information about creating configuration groups, see the *CLI User Guide*.

For more information about the initial configuration for redundant Routing Engine systems and the **re0** group, see *Junos OS High Availability Library for Routing Devices*.

1. Create the configuration group **re0**. The **re0** group is a special group designator that is only used by **RE0** in a redundant routing platform.

```
[edit]
root# set groups re0
```

2. Navigate to the **groups re0** level of the configuration hierarchy.

```
[edit]
root# edit groups re0
```

3. Specify the router hostname.

```
[edit groups re0]
root# set system host-name host-name
```



**NOTE:** The hostname specified in the router configuration is not used by the DNS server to resolve to the correct IP address. This hostname is used to display the name of the Routing Engine in the CLI. For example, the hostname appears at the command-line prompt when the user is logged in to the CLI:

```
user-name@host-name>
```

4. Configure the IP address and prefix length for the router Ethernet interface.
  - For all devices *except* the TX Matrix Plus router, T1600 or T4000 routers in a routing matrix, and PTX Series Packet Transport Routers:

```
[edit]
root@# set interfaces fxp0 unit 0 family inet address address/prefix-length
```

- For the TX Matrix Plus router, T1600 or T4000 routers in a routing matrix only, and PTX Series Packet Transport Routers:

```
[edit]
root@# set interfaces em0 unit 0 family inet address address/prefix-length
```

To use **em0** as an out-of-band management Ethernet interface, you must configure its logical port, **em0.0**, with a valid IP address.

- For a T1600 standalone router (not connected to a TX Matrix Plus router and not in a routing matrix):

```
[edit]
root@# set interfaces fxp0 unit 0 family inet address address/prefix-length
```

5. Return to the top level of the hierarchy.

```
[edit groups re0]
root# top
```

6. Create the configuration group **re1**.

```
[edit]
root# set groups re1
```

7. Navigate to the **groups re1** level of the configuration hierarchy.

```
[edit]
root# edit groups re1
```

8. Specify the router hostname.

```
[edit groups re1]
root# set system host-name host-name
```

9. Configure the IP address and prefix length for the router Ethernet interface.

- For all devices *except* the TX Matrix Plus router, T1600 or T4000 routers in a routing matrix, and PTX Series Packet Transport Routers:

```
[edit]
root@# set interfaces fxp0 unit 0 family inet address address/prefix-length
```

- For the TX Matrix Plus router and T1600 or T4000 routers in a routing matrix only:

```
[edit]
root@# set interfaces em0 unit 0 family inet address address/prefix-length
```

To use **em0** as an out-of-band management Ethernet interface, you must configure its logical port, **em0.0**, with a valid IP address.

- For a T1600 standalone router (not connected to a TX Matrix Plus router, and not in a routing matrix):

```
[edit]
root@# set interfaces fxp0 unit 0 family inet address address/prefix-length
```

10. Return to the top level of the hierarchy.

```
[edit groups re0]
root# top
```

11. Specify the group application order.

```
[edit]
root# set apply-groups [re0 re1]
```



# Controlling the CLI Environment

- [Controlling the Junos OS CLI Environment on page 247](#)
- [Setting the Junos OS CLI Screen Length and Width on page 249](#)
- [Example: Controlling the CLI Environment on page 250](#)
- [Example: Enabling Configuration Breadcrumbs on page 256](#)

## Controlling the Junos OS CLI Environment

---

In operational mode, you can control the Junos OS command-line interface (CLI) environment. For example, you can specify the number of lines that are displayed on the screen or your terminal type. The following output lists the options that you can use to control the CLI environment:

```
user@host>set cli ?
Possible completions:
complete-on-space Set whether typing space completes current word
directory Set working directory
idle-timeout Set maximum idle time before login session ends
logical-system Set default logical system
prompt Set CLI command prompt string
restart-on-upgrade Set whether CLI prompts to restart after software upgrade

screen-length Set number of lines on screen
screen-width Set number of characters on a line
terminal Set terminal type
timestamp Timestamp CLI output
```



**NOTE:** When you use SSH to log in to the router or log in from the console when its terminal type is already configured, your terminal type, screen length, and screen width are already set.

This chapter discusses the following topics:

- [Setting the Terminal Type on page 248](#)
- [Setting the CLI Prompt on page 248](#)
- [Setting the CLI Directory on page 248](#)
- [Setting the CLI Timestamp on page 248](#)

- [Setting the Idle Timeout on page 248](#)
- [Setting the CLI to Prompt After a Software Upgrade on page 248](#)
- [Setting Command Completion on page 249](#)
- [Displaying CLI Settings on page 249](#)

## Setting the Terminal Type

To set the terminal type, use the **set cli terminal** command:

```
user@host> set cli terminal terminal-type
```

The terminal type can be one of the following: **ansi**, **vt100**, **small-xterm**, or **xterm**.

## Setting the CLI Prompt

The default CLI prompt is **user@host>**. To change this prompt, use the **set cli prompt** command. If the prompt string contains spaces, enclose the string in quotation marks ( " ").

```
user@host> set cli prompt string
```

## Setting the CLI Directory

To set the current working directory, use the **set cli directory** command:

```
user@host> set cli directory directory
```

**directory** is the pathname of working directory.

## Setting the CLI Timestamp

By default, CLI output does not include a timestamp. To include a timestamp in CLI output, use the **set cli timestamp** command:

```
user@host> set cli timestamp [format time-date-format | disable]
```

If you do not specify a timestamp format, the default format is **Mmm dd hh:mm:ss** (for example, Feb 08 17:20:49). Enclose the format in single quotation marks ( ' ).

## Setting the Idle Timeout

By default, an individual CLI session never times out after extended times, unless the **idle-timeout** statement has been included in the user's login class configuration. To set the maximum time an individual session can be idle before the user is logged off the router, use the **set cli idle-timeout** command:

```
user@host> set cli idle-timeout timeout
```

**timeout** can be 0 through 100,000 minutes. Setting **timeout** to 0 disables the timeout.

## Setting the CLI to Prompt After a Software Upgrade

By default, the CLI prompts you to restart after a software upgrade. To disable the prompt for an individual session, use the **set cli restart-on-upgrade off** command:

```
user@host> set cli restart-on-upgrade off
```



To reenable the prompt, use the **set cli restart-on-upgrade on** command:

```
user@host> set cli restart-on-upgrade on
```

Setting Command Completion

By default, you can press Tab or the Spacebar to have the CLI complete a command.

To have the CLI allow only a tab to complete a command, use the **set cli complete-on-space off** command:

```
user@host> set cli complete-on-space off
Disabling complete-on-space
user@host>
```

To reenable the use of both spaces and tabs for command completion, use the **set cli complete-on-space on** command:

```
user@host> set cli complete-on-space on
Enabling complete-on-space
user@host>
```

Displaying CLI Settings

To display the current CLI settings, use the **show cli** command:

```
user@host> show cli
CLI screen length set to 24
CLI screen width set to 80
CLI complete-on-space set to on
```



**NOTE:** In Junos OS Release 13.3 and later, the value of screen width is 0 or in the range of 40 through 1024.

Release History Table

Release	Description
13.3	In Junos OS Release 13.3 and later, the value of <b>screen width</b> is <b>0</b> or in the range of <b>40</b> through <b>1024</b> .

- Related Documentation
- [Example: Controlling the CLI Environment on page 250](#)

Setting the Junos OS CLI Screen Length and Width

You can set the Junos OS command-line interface (CLI) screen length and width according to your specific requirements. This topic contains the following sections:

- [Setting the Screen Length on page 250](#)
- [Setting the Screen Width on page 250](#)

## Setting the Screen Length

The default CLI screen length is 24 lines. To change the length, use the **set cli screen-length** command:

```
user@host> set cli screen-length length
```

Setting the screen length to 0 lines disables the display of output one screen at a time. Disabling this UNIX **more**-type interface can be useful when you are issuing CLI commands from scripts.

## Setting the Screen Width

The value of CLI screen width is 0 or in the range of 40 through 1024. The default CLI screen width is 80 characters. To change the width, use the **set cli screen-width** command:

```
user@host> set cli screen-width width
```



**NOTE:** In Junos OS Release 13.2 and earlier, the value of *width* is in the range of 0 through 1024.

Release History Table

Release	Description
13.2	In Junos OS Release 13.2 and earlier, the value of <i>width</i> is in the range of 0 through 1024.

### Related Documentation

- [Example: Controlling the CLI Environment on page 250](#)
- [Controlling the Junos OS CLI Environment on page 247](#)

## Example: Controlling the CLI Environment

The following example shows you how to change the default CLI environment.

Changing the CLI environment is all about customizing the CLI window to fit your personal preferences. Use the settings discussed in this topic to make the CLI window look and behave according to what you find most convenient and efficient.

- [Requirements on page 250](#)
- [Overview on page 251](#)
- [Configuration on page 251](#)

## Requirements

No special configuration beyond device initialization is required before configuring this example.

Before starting this example, check what the default settings are. Use the **show cli** operational mode command.

```
user@host> show cli
CLI complete-on-space set to on
CLI idle-timeout disabled
CLI restart-on-upgrade set to on
CLI screen length set to 66
CLI screen width set to 80
CLI terminal is 'xterm'
```

Is the prompt set to your *username@routername*? If not, exit the CLI and enter the operational mode again.

Is the CLI screen length set to 66 and the CLI screen width set to 80? If so, you can start the example. Otherwise, make these changes to the CLI settings:

```
user@host> set cli screen-length 66
Screen length set to 66 lines long
user@host> set cli screen-width 80
Screen width set to 80 columns wide
```

## Overview

To see a list of CLI environmental settings that you can change, use the **set cli ?** command.

```
user@host> set cli ?
Possible completions:
complete-on-space Set whether typing space completes current word
directory Set working directory
idle-timeout Set maximum idle time before login session ends
logical-system Set default logical system
prompt Set CLI command prompt string
restart-on-upgrade Set whether CLI prompts to restart after software upgrade
screen-length Set number of lines on screen
screen-width Set number of characters on a line
terminal Set terminal type
timestamp Timestamp CLI output
```

This example focuses on three of these commands: **set cli screen-length**, **set cli screen-width**, and **set cli prompt**.

## Configuration

This configuration example has the following sections:

- [Configuring the CLI Prompt on page 252](#)
- [Configuring CLI Width on page 252](#)
- [Configuring CLI Length on page 253](#)
- [Return to the Default CLI Prompt on page 255](#)

### CLI Quick Configuration

To quickly configure this example, copy the following commands and paste them in a text file, remove any line breaks, change the values used to match your network configuration, and then copy and paste the commands into the CLI at the operational command prompt.

```
set cli prompt "router1-san-jose> "
```

```
set cli screen-width 110
set cli screen-length 45
```



**NOTE:** In Junos OS Release 13.3 and later, the value of screen width is 0 or in the range of 40 through 1024.

---

### Configuring the CLI Prompt

---

#### Step-by-Step Procedure

The default CLI prompt is your *username@hostname*. But you can have any prompt you find useful.

To configure a different CLI prompt:

- Use the following operational mode command where *string* is the exact text you want to see at the command line.

```
set cli prompt "string"
```

For example, if "*string*" is "router1-san-jose> ", the command is as follows:

```
set cli prompt "router1-san-jose> "
router1-san-jose>
```

---

### Configuring CLI Width

---

#### Step-by-Step Procedure

How do you know what width works best for you? This example discusses how CLI width can affect what you see.

To configure a new default CLI width:

1. See what the current defaults are for the CLI environment.

```
router1-san-jose> show cli
CLI complete-on-space set to on
CLI idle-timeout disabled
CLI restart-on-upgrade set to on
CLI screen length set to 66
CLI screen width set to 80
CLI terminal is 'xterm'
router1-san-jose>
```



**NOTE:** In Junos OS Release 13.3 and later, the value of *width* is 0 or in the range of 40 through 1024.

2. Look at the following output for the operational command **show class-of-service forwarding-class**.

The output from this command is wider than some and so illustrates a common problem with viewing output. If, for example, you have a relatively narrow window, command output might show up in overrun lines.

```

router1-san-jose> show class-of-service forwarding-class
Forwarding class ID Queue Restricted queue Fabric
priority Policing priority SPU priority
premium-rate 0 0 0 low
normal
medium-rate 1 1 1 low
normal
low-rate 2 2 2 low
normal
NC 3 3 3 low
normal
tunnel-rate 4 4 0 low
normal

```

The lines look to be intermingled and it is hard to read across to find the information you might be seeking.

3. Change the window width to 110 columns.

Notice how the output of this command is much easier to read in the wider format:

```
router1-san-jose> set cli screen-width 110
```

```

router1-san-jose> show class-of-service forwarding-class
Forwarding class ID Queue Restricted queue Fabric priority Policing priority SPU priority
premium-rate 0 0 0 low normal low
medium-rate 1 1 1 low normal low
low-rate 2 2 2 low normal low
NC 3 3 3 low normal low
tunnel-rate 4 4 0 low normal low

```

## Configuring CLI Length

**Step-by-Step Procedure** You can configure the length of the CLI screen in a similar fashion as you did the width.

To configure a new default CLI length:

1. See what the current defaults are for the CLI environment.

```

router1-san-jose> show cli
CLI complete-on-space set to on
CLI idle-timeout disabled
CLI restart-on-upgrade set to on
CLI screen length set to 66
CLI screen width set to 80
CLI terminal is 'xterm'
router1-san-jose>

```

2. Look at the following output for the operational command **show version**.

```

Makefile sync-dpm-sb.manifest
build sync-equilibrium-sb.manifest
etc sync-equilibrium2-sb.manifest
include sync-hellopics-sb.manifest
jexample sync-ipprobe-mt-sb.manifest
jnx-cc-routeservice-sb.manifest sync-ipprobe-sb.manifest
jnx-example-sb.manifest sync-ipsnooper-sb.manifest
jnx-flow-sb.manifest sync-monitube-sb.manifest

```

```

jnx-gateway-sb.manifest sync-monitube2-plugin-sb.manifest
jnx-ifinfo-sb.manifest sync-packetproc-sb.manifest
jnx-mspexamp1ed-sb.manifest sync-passthru-sb.manifest
jnx-msprsm-sb.manifest sync-policy-manager-sb.manifest
jnx-routeservice-sb.manifest sync-reassembler-sb.manifest
lib sync-route-manager-sb.manifest
JnprFirewall-Proto.html Makefile.depend.octeon dfw_filter.proto
JnprFirewall.html Makefile.depend.powerpc dfw_ifattach.proto
Makefile Makefile.depend.xlr dfw_policer.proto
Makefile.depend.arm dfw.jsdl dfw_stats.proto
Makefile.depend.host dfw_bulk.proto
Makefile.depend.i386 dfw_common.proto

```

```

Trying 192.168.184.75...
Connected to spot-fxp0.englab.juniper.net.
Escape character is '^]'.
Unauthorized use is prohibited.

```

```

router1-san-jose> show version
Hostname: spot
Model: mx240
Junos: 14.2-20140710_ib_14_2_psd.1
JUNOS Base OS boot [14.2-20140710_ib_14_2_psd.1]
JUNOS Base OS Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Kernel Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Crypto Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Packet Forwarding Engine Support (M/T/EX Common)
[14.2-20140710_ib_14_2_psd.1]
JUNOS Packet Forwarding Engine Support (MX Common)
[14.2-20140710_ib_14_2_psd.1]
JUNOS Online Documentation [14.2-20140710_ib_14_2_psd.1]
JUNOS Services AACL Container package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Application Level Gateways [14.2-20140710_ib_14_2_psd.1]
JUNOS AppId Services [14.2-20140710_ib_14_2_psd.1]
JUNOS Border Gateway Function package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Captive Portal and Content Delivery Container package
[14.2-20140710_ib_14_2_psd.1]
JUNOS Services HTTP Content Management package [14.2-20140710_ib_14_2_psd.1]
JUNOS IDP Services [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Jflow Container package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services LL-PDF Container package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services MobileNext Software package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Mobile Subscriber Service Container package
[14.2-20140710_ib_14_2_psd.1]
JUNOS Services NAT [14.2-20140710_ib_14_2_psd.1]
JUNOS Services PTSP Container package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services RPM [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Stateful Firewall [14.2-20140710_ib_14_2_psd.1]
JUNOS Voice Services Container package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Crypto [14.2-20140710_ib_14_2_psd.1]
JUNOS Services SSL [14.2-20140710_ib_14_2_psd.1]
JUNOS Services IPSec [14.2-20140710_ib_14_2_psd.1]
JUNOS platform Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Routing Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Runtime Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Web Management [14.2-20140710_ib_14_2_psd.1]
JUNOS py-base-i386 [14.2-20140710_ib_14_2_psd.1]

```

```
router1-san-jose>
```

The current length is 66 lines, which is close to the length of a typical monitor. But even though the output is fairly long, it hardly needs all that space to be clearly seen in its entirety. In fact, it is harder to pick out just where the output starts in a screen this long.

3. Change the window width to 45 lines.

```
router1-san-jose> set cli screen-length 45
```

4. Now look at the output again.

```
router1-san-jose> show version
Hostname: spot
Model: mx240
Junos: 14.2-20140710_ib_14_2_psd.1
JUNOS Base OS boot [14.2-20140710_ib_14_2_psd.1]
JUNOS Base OS Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Kernel Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Crypto Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Packet Forwarding Engine Support (M/T/EX Common)
[14.2-20140710_ib_14_2_psd.1]
JUNOS Packet Forwarding Engine Support (MX Common)
[14.2-20140710_ib_14_2_psd.1]
JUNOS Online Documentation [14.2-20140710_ib_14_2_psd.1]
JUNOS Services AACL Container package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Application Level Gateways [14.2-20140710_ib_14_2_psd.1]
JUNOS AppId Services [14.2-20140710_ib_14_2_psd.1]
JUNOS Border Gateway Function package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Captive Portal and Content Delivery Container package
[14.2-20140710_ib_14_2_psd.1]
JUNOS Services HTTP Content Management package [14.2-20140710_ib_14_2_psd.1]
JUNOS IDP Services [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Jflow Container package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services LL-PDF Container package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services MobileNext Software package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Mobile Subscriber Service Container package
[14.2-20140710_ib_14_2_psd.1]
JUNOS Services NAT [14.2-20140710_ib_14_2_psd.1]
JUNOS Services PTSP Container package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services RPM [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Stateful Firewall [14.2-20140710_ib_14_2_psd.1]
JUNOS Voice Services Container package [14.2-20140710_ib_14_2_psd.1]
JUNOS Services Crypto [14.2-20140710_ib_14_2_psd.1]
JUNOS Services SSL [14.2-20140710_ib_14_2_psd.1]
JUNOS Services IPSec [14.2-20140710_ib_14_2_psd.1]
JUNOS platform Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Routing Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Runtime Software Suite [14.2-20140710_ib_14_2_psd.1]
JUNOS Web Management [14.2-20140710_ib_14_2_psd.1]
JUNOS py-base-i386 [14.2-20140710_ib_14_2_psd.1]
```

```
router1-san-jose>
```

With a shorter screen, you can easily see where the current output begins and ends.

### [Return to the Default CLI Prompt](#)

#### Step-by-Step Procedure

To go back to the default prompt:

1. Exit the CLI.

```
router1-san-jose> exit
%
```

2. Enter the CLI operational mode again.

```
% cli
user@host>
```

**Related  
Documentation**

- [Setting the Junos OS CLI Screen Length and Width on page 249](#)
- [Controlling the Junos OS CLI Environment on page 247](#)

---

## Example: Enabling Configuration Breadcrumbs

The output of **show configuration** operational mode command and **show** configuration mode commands can be configured to display configuration breadcrumbs that indicate the exact location in the hierarchy of the output being viewed.

Before enabling the configuration breadcrumbs feature, check the output of the **show configuration** command.

```
user@host> show configuration
```

```
...
 }
 }
}
fe-4/1/2 {
 description "FA4/1/2: mxvj1-mr6 (64.12.137.160/27) (T=bb1an, bbmail,
bbowmtc)";
 unit 0 {
 family inet {
 filter {
 output 151mj;
 }
 address 64.12.137.187/27 {
 vrrp-group 1 {
 virtual-address 64.12.137.189;
 }
 }
 }
 }
}
---(more 18%)-----
```

In the output, there is no clear indication about the section of the configuration being viewed.

To enable the configuration breadcrumbs feature:

1. Define a class at the **[edit system login]** hierarchy level.

```
[edit system login]
user@host# set class breadclass idle-timeout 10
```

2. Add a user to the defined login class to enable the breadcrumbs output view when this user enters the **show configuration** operational mode command.

```
[edit system login user user1]
user@host# set class breadclass
```



3. Configure the **configuration-breadcrumbs** statement at the **[edit system login class <class name>]** hierarchy level.

```
[edit system login class breadclass]
user@host# set configuration-breadcrumbs
```

4. Confirm the configuration.

```
[edit]
user@host# commit
```

On enabling configuration breadcrumbs in the CLI, User1 (the user added to the login class) can verify the feature in the output by entering the **show configuration** command.

```
user1@host> show configuration
```

```
...
 }
 }
}
fe-4/1/2 {
 description "FA4/1/2: mxxj1-mr6 (64.12.137.160/27) (T=bb1an, bbmail,
bbowmtc)";
 unit 0 {
 family inet {
 filter {
 output 151mj;
 }
 address 64.12.137.187/27 {
 vrrp-group 1 {
 virtual-address 64.12.137.189;
---(more 18%)---[groups main interfaces fe-4/1/2 unit 0 family inet address
64.12.137.187/27 vrrp-group 1]---
```

The new output indicates the exact location of the configuration hierarchy being viewed. User1 is currently viewing the interface configuration of a group.



**NOTE:** If you are enabling configuration breadcrumbs for your own user account, you should log out and log in again to see the changes.

- Related Documentation**
- [class](#)
  - [configuration-breadcrumbs on page 270](#)



## CHAPTER 12

# Junos OS Configuration Statements and Commands

- [apply-groups on page 260](#)
- [apply-groups-except on page 261](#)
- [activate](#)
- [annotate](#)
- [commit](#)
- [commit-interval \(Batch Commits\) on page 269](#)
- [configuration-breadcrumbs on page 270](#)
- [copy](#)
- [days-to-keep-error-logs \(Batch Commits\) on page 271](#)
- [deactivate](#)
- [delete](#)
- [edit](#)
- [exit](#)
- [export-format on page 276](#)
- [groups on page 277](#)
- [help](#)
- [insert](#)
- [load](#)
- [maximum-aggregate-pool \(Batch Commits\) on page 283](#)
- [maximum-entries \(Batch Commits\) on page 283](#)
- [no-hidden-commands on page 284](#)
- [protect](#)
- [quit](#)
- [rename](#)
- [replace](#)
- [rollback](#)

- [run](#)
- [save](#)
- [server \(Batch Commits\) on page 292](#)
- [set](#)
- [show](#)
- [show configuration](#)
- [show | display inheritance](#)
- [show | display omit](#)
- [show | display set](#)
- [show | display set relative](#)
- [show groups junos-defaults](#)
- [status](#)
- [top](#)
- [traceoptions \(Batch Commits\) on page 305](#)
- [unprotect](#)
- [up](#)
- [update](#)
- [when on page 309](#)
- [wildcard delete](#)

---

## [apply-groups](#)

---

<b>Syntax</b>	<code>apply-groups [ <i>group-names</i> ];</code>
<b>Hierarchy Level</b>	All hierarchy levels
<b>Release Information</b>	Statement introduced before Junos OS Release 7.4.
<b>Description</b>	<p>Apply a configuration group to a specific hierarchy level in a configuration, to have a configuration inherit the statements in the configuration group.</p> <p>You can specify more than one group name. You must list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.</p>
<b>Options</b>	<i>group-names</i> —One or more names specified in the <b>groups</b> statement.
<b>Required Privilege Level</b>	<code>configure</code> —To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Applying the Junos OS Configuration Group on page 219</a></li><li>• <a href="#">groups on page 277</a></li></ul>

## **apply-groups-except**

---

<b>Syntax</b>	<code>apply-groups-except [ <i>group-names</i> ];</code>
<b>Hierarchy Level</b>	All hierarchy levels except the top level
<b>Release Information</b>	Statement introduced before Junos OS Release 7.4.
<b>Description</b>	Disable inheritance of a configuration group.
<b>Options</b>	<i>group-names</i> —One or more names specified in the <b>groups</b> statement.
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">groups on page 277</a></li><li>• <a href="#">Disabling Inheritance of a Junos OS Configuration Group on page 222</a></li></ul>

## activate

---

<b>Syntax</b>	activate ( <i>statement</i>   <i>identifier</i> )
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Remove the <b>inactive:</b> tag from a statement, effectively adding the statement or identifier back to the configuration. Statements or identifiers that have been activated take effect when you next issue the <b>commit</b> command.
<b>Options</b>	<p><b>identifier</b>—Identifier from which you are removing the <b>inactive</b> tag. It must be an identifier at the current hierarchy level.</p> <p><b>statement</b>—Statement from which you are removing the <b>inactive</b> tag. It must be a statement at the current hierarchy level.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">deactivate on page 272</a></li><li>• <a href="#">Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration on page 89</a></li></ul>

## annotate

**Syntax** `annotate statement "comment-string"`

**Release Information** Command introduced before Junos OS Release 7.4.

**Description** Add comments to a configuration. You can add comments only at the current hierarchy level.

Any comments you add appear only when you view the configuration by entering the [show](#) command in configuration mode or the **show configuration** command in operational mode.



**NOTE:** The Junos OS supports annotation up to the last level in the configuration hierarchy, including oneliners. However, annotation of parts (child statements or identifiers within a oneliner) of the oneliner is not supported. For example, in the following sample configuration hierarchy, annotation is supported up to the oneliner level 1, but not supported for the metric child statement and its attribute *10*:

```
[edit protocols]
 isis {
 interface ge-0/0/0.0 {
 level 1 metric 10;
 }
 }
}
```

**Options** *comment-string*—Text of the comment. You must enclose it in quotation marks. In the comment string, you can include the comment delimiters `/* */` or `#`. If you do not specify any, the comment string is enclosed with the `/* */` comment delimiters. If a comment for the specified *statement* already exists, it is deleted and replaced with the new comment.

*statement*—Statement to which you are attaching the comment.

**Required Privilege Level** `configure`—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.

**Related Documentation**

- [Adding Comments in a Junos OS Configuration on page 92](#)

## commit

**Syntax** commit <<*at* <"string">> <and-quit> <check> <comment <"comment-string">>  
<confirmed> <display detail> <fast-synchronize> <minutes> <peers-synchronize >  
<synchronize <force> <scripts>>

**Release Information** Command introduced before Junos OS Release 7.4.  
Command introduced in Junos OS Release 11.1 for the QFX Series.  
Option **fast-synchronize** added in Junos OS Release 12.2.  
Option **synchronize scripts** introduced in Junos OS Release 13.2.  
Command introduced in Junos OS Release 14.1X53-D20 for the OCX Series.  
Option **peers-synchronize** introduced in Junos OS Release 14.2R6.

**Description** Commit the set of changes to the database and cause the changes to take operational effect.



**NOTE:** The **fast-synchronize** option is not supported in a QFX Series Virtual Chassis.



**NOTE:** Beginning in Junos OS 12.3, it is possible that FPCs brought offline using the **request chassis fpc slot *fpc-slot* offline** operational-mode CLI command can come online during a configuration commit or power-supply replacement procedure. As an alternative, use the **set fpc *fpc-slot* power off** configuration-mode command at the [edit chassis] hierarchy level to ensure that the FPCs remain offline.

**Options** **at** <"string">—(Optional) Save software configuration changes and activate the configuration at a future time, or upon reboot.

**string** is **reboot** or the future time to activate the configuration changes. Enclose the **string** value (including **reboot**) in quotation marks (" "). You can specify time in two formats:



- A time value in the form **hh:mm[:ss]** (hours, minutes, and optionally seconds)—Commit the configuration at the specified time, which must be in the future but before 11:59:59 PM on the day the **commit at** configuration command is issued. Use 24-hour time for the **hh** value; for example, **04:30:00** is 4:30:00 AM, and **20:00** is 8:00 PM. The time is interpreted with respect to the clock and time zone settings on the router.
- A date and time value in the form **yyyy-mm-dd hh:mm[:ss]** (year, month, date, hours, minutes, and, optionally, seconds)—Commit the configuration at the specified day and time, which must be after the **commit at** command is issued. Use 24-hour time for the **hh** value. For example, **2003-08-21 12:30:00** is 12:30 PM on August 21, 2003. The time is interpreted with respect to the clock and time zone settings on the router.

For example, **commit at "18:00:00"**. For date and time, include both values in the same set of quotation marks. For example, **commit at "2005-03-10 14:00:00"**.

A *commit check* is performed when you issue the **commit at** configuration mode command. If the result of the check is successful, then the current user is logged out of configuration mode, and the configuration data is left in a read-only state. No other commit can be performed until the scheduled commit is completed.



**NOTE:** If Junos OS fails before the configuration changes become active, all configuration changes are lost.

You cannot enter the **commit at** configuration command when there is a pending reboot.

You cannot enter the **request system reboot** command once you schedule a commit operation for a specific time in the future.

You cannot commit a configuration when a scheduled commit is pending. For information about how to use the **clear** command to cancel a scheduled configuration, see the [CLI Explorer](#).

**and-quit**—(Optional) Commit the configuration and, if the configuration contains no errors and the commit succeeds, exit from configuration mode.

**check**—(Optional) Verify the syntax of the configuration, but do not activate it.

**comment** <*comment-string*>—(Optional) Add a comment that describes the committed configuration. The comment can be as long as 512 bytes and must be typed on a single line. You cannot include a comment with the **commit check** command. Enclose *comment-string* in quotation marks (" "). For example, **commit comment "Includes changes recommended by SW Lab"**.

**confirmed** <*minutes*>—(Optional) Require that the commit be confirmed within the specified amount of time. To confirm a commit, enter either a **commit** or **commit check** command. If the commit is not confirmed within the time limit, the configuration rolls back automatically to the precommit configuration and a broadcast message is sent to

all logged-in users. To show when a rollback is scheduled, enter the **show system commit** command. The allowed range is 1 through 65,535 minutes, and the default is 10 minutes.

In Junos OS Release 11.4 and later, you can also use the **commit confirmed** command in the **[edit private]** configuration mode.

**display detail**—(Optional) Monitors the commit process.



**NOTE:** In Junos OS Release 10.4 and later, if the number of commit details or messages exceeds a page when used with the **| display detail** pipe option, the more pagination option on the screen is no longer available. Instead, the messages roll up on the screen by default, just like using the **commit** command with the **| no more** pipe option.

**fast-synchronize**—(Optional) Configure the commits to run in parallel on both the master and backup Routing Engines to reduce the time taken for commit synchronization.



**NOTE:** The **fast-synchronize** statement is not supported on QFX Series devices when used in a Virtual Chassis.

**peers-synchronize**—(Optional) Automatically synchronizes and commits MC-LAG configurations across the peers. The local peer (the requesting peer) on which you enable the **peers-synchronize** statement copies and loads its configuration to the remote (the responding) peer. Each peer then performs a syntax check on the configuration file being committed. If no errors are found, the configuration is activated and becomes the current operational configuration on both peers.

**synchronize <force> <scripts>**—(Optional) If your router has two Routing Engines, you can manually direct one Routing Engine to synchronize its configuration with the other by issuing the **commit synchronize** command. The Routing Engine on which you execute this command (the request Routing Engine) copies and loads its candidate configuration to the other Routing Engine (the responding Routing Engine). Both Routing Engines then perform a syntax check on the candidate configuration file being committed. If no errors are found, the configuration is activated and becomes the current operational configuration on both Routing Engines.

It can happen that the **commit synchronize** command is initiated at the same time from both Routing Engines, which causes the process to hang. As of Junos OS Release 15.1, this is a temporary (20 seconds) anomaly, after which the user can try the **commit synchronize** command again.

The **commit synchronize** command does not work if the responding Routing Engine has uncommitted configuration changes. However, you can enforce commit synchronization on the Routing Engines by using the **force** option. When you issue the **commit synchronize** command with the **force** option from one Routing Engine, the configuration sessions on

the other Routing Engine are terminated and its configuration synchronized with that on the Routing Engine from which you issued the command.

When you issue the **commit synchronize** command with the **scripts** option, the device synchronizes all commit, event, lib, op, and SNMP scripts from the requesting Routing Engine to the responding Routing Engine and also commits and synchronizes the configuration. If the commit check operation fails for the requesting Routing Engine, the process stops, and the scripts are not copied to the responding Routing Engine. If the commit check or commit operation fails for the responding Routing Engine, the scripts are still synchronized, since the synchronization occurs prior to the commit check operation on the responding Routing Engine.

If the **load-scripts-from-flash** statement is configured for the requesting Routing Engine, the device synchronizes the scripts from flash memory on the requesting Routing Engine to flash memory on the responding Routing Engine. Otherwise, the device synchronizes the scripts from the hard disk on the requesting Routing Engine to the hard disk on the responding Routing Engine. The device synchronizes all scripts regardless of whether they are enabled in the configuration or have been updated since the last synchronization.



**NOTE:** When you issue the **commit synchronize** command, you must use the **apply-groups re0** and **re1** commands. For information about how to use groups, see [“Disabling Inheritance of a Junos OS Configuration Group” on page 222](#).

The responding Routing Engine must use Junos OS Release 5.0 or later.

#### Required Privilege Level

configure—To enter configuration mode.



**NOTE:** If you are using Junos OS in a Common Criteria environment, system log messages are created whenever a secret attribute is changed (for example, password changes or changes to the RADIUS shared secret). These changes are logged during the following configuration load operations:

```
load merge
load replace
load override
load update
```

For more information, see the *Secure Configuration Guide for Common Criteria and Junos-FIPS*

#### Related Documentation

- [Verifying a Junos OS Configuration on page 104, Committing a Junos OS Configuration on page 108](#)
- [Scheduling a Junos OS Commit Operation on page 113](#)
- [Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration on page 89](#)

- [Monitoring the Junos OS Commit Process on page 114](#)
- [Adding a Comment to Describe the Committed Configuration on page 115](#)
- [Committing Configurations on a Routing Matrix with a TX Matrix Plus Router](#)

## Sample Output

### commit | display detail

```

user@host> commit | display detail

2011-08-24 01:08:08.00691 PDT: begin creating snapshots
2011-08-24 01:08:09.00210 PDT: end creating snapshots
2011-08-24 01:08:09.00211 PDT: begin preparing metadata
2011-08-24 01:08:09.00228 PDT: end preparing metadata
2011-08-24 01:08:09.00229 PDT: begin computing dcf root changes
2011-08-24 01:08:09.00236 PDT: end computing dcf root changes
2011-08-24 01:08:09.00244 PDT: begin computing additions
2011-08-24 01:08:09.00251 PDT: end computing additions
2011-08-24 01:08:09.00251 PDT: begin local object validation
2011-08-24 01:08:09.00251 PDT: end local object validation
2011-08-24 01:08:09.00252 PDT: begin update instances
2011-08-24 01:08:09.00252 PDT: end update instances
2011-08-24 01:08:09.00252 PDT: begin adjust metadata
2011-08-24 01:08:09.00252 PDT: end adjust metadata
2011-08-24 01:08:09.00253 PDT: begin validate metadata
2011-08-24 01:08:09.00253 PDT: end validate metadata
2011-08-24 01:08:09.00253 PDT: begin adjust allocations
2011-08-24 01:08:09.00254 PDT: end adjust allocations
2011-08-24 01:08:09.00254 PDT: begin adjust dependencies
2011-08-24 01:08:09.00254 PDT: end adjust dependencies
2011-08-24 01:08:09.00255 PDT: begin instance validation
2011-08-24 01:08:09.00255 PDT: end instance validation
2011-08-24 01:08:09.00255 PDT: begin opening all sessions eagerly
2011-08-24 01:08:09.00277 PDT: begin request #1 [login]
2011-08-24 01:08:09.00278 PDT: end request #1 [login]
2011-08-24 01:08:09.00325 PDT: begin processing globals
2011-08-24 01:08:09.00330 PDT: begin waiting for stamp check
(qfabric-default---node0)
2011-08-24 01:08:09.00334 PDT: end reply #1 [login]
2011-08-24 01:08:09.00351 PDT: end reply #1 [login]
2011-08-24 01:08:09.00451 PDT: begin request #2 [open]
2011-08-24 01:08:09.00451 PDT: end request #2 [open]
2011-08-24 01:08:09.00451 PDT: begin request #3 [get commit history]
2011-08-24 01:08:09.00452 PDT: end request #3 [get commit history]
2011-08-24 01:08:09.00452 PDT: begin request #4 [load]
2011-08-24 01:08:09.00453 PDT: end request #4 [load]
2011-08-24 01:08:09.00453 PDT: begin request #5 [load]
2011-08-24 01:08:09.00454 PDT: begin reply #2 [open]
2011-08-24 01:08:09.00456 PDT: end reply #2 [open]
2011-08-24 01:08:09.00457 PDT: begin reply #3 [get commit history]
2011-08-24 01:08:09.00475 PDT: end reply #3 [get commit history]
2011-08-24 01:08:09.00476 PDT: begin reply #4 [load]
2011-08-24 01:08:09.00499 PDT: begin reply #5 [load]
2011-08-24 01:08:09.00501 PDT: end waiting for stamp check
(qfabric-default---node0)
2011-08-24 01:08:09.00501 PDT: begin waiting for open (qfabric-default---node0)
2011-08-24 01:08:09.00502 PDT: end waiting for open (qfabric-default---node0)
2011-08-24 01:08:09.00504 PDT: end processing globals

```

```

2011-08-24 01:08:09.00617 PDT: end request #5 [load]
2011-08-24 01:08:09.00617 PDT: begin request #6 [check]
2011-08-24 01:08:09.00617 PDT: end request #6 [check]
2011-08-24 01:08:09.00619 PDT: end reply #5 [load]
2011-08-24 01:08:09.00619 PDT: begin reply #6 [check]
2011-08-24 01:08:09.00730 PDT: end session
2011-08-24 01:08:09.00752 PDT: end request #5 [load]
2011-08-24 01:08:09.00754 PDT: begin request #6 [check]
2011-08-24 01:08:09.00755 PDT: end request #6 [check]
2011-08-24 01:08:09.00881 PDT: end request #5 [load]
2011-08-24 01:08:09.00961 PDT: begin commit to devices
2011-08-24 01:08:10.00668 PDT: begin request #8 [get commit history]
2011-08-24 01:08:10.00669 PDT: end request #8 [get commit history]
2011-08-24 01:08:10.00721 PDT: end session
2011-08-24 01:08:10.00727 PDT: end commit to devices
2011-08-24 01:08:10.00733 PDT: begin committing metadata
2011-08-24 01:08:10.00772 PDT: end committing metadata
2011-08-24 01:08:10.00772 PDT: begin calling commit callbacks
2011-08-24 01:08:10.00773 PDT: end calling commit callbacks
commit complete

```

## commit-interval (Batch Commits)

<b>Syntax</b>	<code>commit-interval <i>number-of-seconds-between-commits</i>;</code>
<b>Hierarchy Level</b>	[edit system commit server], [edit system commit synchronize server]
<b>Release Information</b>	Statement introduced in Junos OS Release 12.1.
<b>Description</b>	For Junos OS batch commits, specify the time interval (in seconds) between two commit operations.
<b>Options</b>	<p><i>number-of-seconds-between-commits</i>—Time interval (in seconds) between two commit operations.</p> <p><b>Range:</b> 1 through 30 seconds.</p> <p><b>Default:</b> 5 seconds.</p>
<b>Required Privilege Level</b>	<p>system—To view this statement in the configuration.</p> <p>system-control—To add this statement to the configuration.</p>
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Example: Configuring Batch Commit Server Properties on page 117</a></li> </ul>

## configuration-breadcrumbs

---

<b>Syntax</b>	configuration-breadcrumbs;
<b>Hierarchy Level</b>	[edit system login class]
<b>Release Information</b>	Statement introduced in Junos OS Release 12.2.
<b>Description</b>	Enable the configuration breadcrumbs view in the CLI to display the location in the configuration hierarchy.
<b>Required Privilege Level</b>	admin—To view this statement in the configuration. admin-control—To add this statement to the configuration.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Example: Enabling Configuration Breadcrumbs on page 256</a></li><li>• <i>Defining Junos OS Login Classes</i></li><li>• <i>class</i></li><li>• <i>login</i></li></ul>

## copy

---

<b>Syntax</b>	<code>copy <i>existing-statement</i> to <i>new-statement</i></code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Make a copy of an existing statement in the configuration.
<b>Options</b>	<p><i>existing-statement</i>—Statement to copy.</p> <p><i>new-statement</i>—Copy of the statement.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Copying a Junos OS Statement in the Configuration on page 73</a></li> </ul>

## days-to-keep-error-logs (Batch Commits)

---

<b>Syntax</b>	<code>days-to-keep-error-logs <i>days-to-keep-error-log-entries</i>;</code>
<b>Hierarchy Level</b>	<p>[edit system commit server],</p> <p>[edit system commit synchronize server]</p>
<b>Release Information</b>	Statement introduced in Junos OS Release 12.1.
<b>Description</b>	For Junos OS batch commits, specify the number of days to keep the error logs.
<b>Options</b>	<p><i>days-to-keep-error-log-entries</i>—Number of days to keep the error logs.</p> <p><b>Range:</b> 1 through 366 days</p> <p><b>Default:</b> 1 day</p>
<b>Required Privilege Level</b>	<p>system—To view this statement in the configuration.</p> <p>system-control—To add this statement to the configuration.</p>
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Example: Configuring Batch Commit Server Properties on page 117</a></li> </ul>

## deactivate

---

<b>Syntax</b>	deactivate ( <i>statement</i>   <i>identifier</i> )
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Add the <b>inactive:</b> tag to a statement, effectively commenting out the statement or identifier from the configuration. Statements or identifiers marked as inactive do not take effect when you issue the <b>commit</b> command.
<b>Options</b>	<p><b>identifier</b>—Identifier to which you are adding the <b>inactive:</b> tag. It must be an identifier at the current hierarchy level.</p> <p><b>statement</b>—Statement to which you are adding the <b>inactive:</b> tag. It must be a statement at the current hierarchy level.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">activate on page 262</a></li><li>• <a href="#">delete on page 273</a></li><li>• <a href="#">Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration on page 89.</a></li></ul>



## delete

---

<b>Syntax</b>	<code>delete &lt;statement-path&gt; &lt;identifier&gt;</code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	<p>Delete a statement or identifier. All subordinate statements and identifiers contained within the specified statement path are deleted with it.</p> <p>Deleting a statement or an identifier effectively “unconfigures” or disables the functionality associated with that statement or identifier.</p> <p>If you do not specify <i>statement-path</i> or <i>identifier</i>, the entire hierarchy, starting at the current hierarchy level, is removed.</p>
<b>Options</b>	<p><i>statement-path</i>—(Optional) Path to an existing statement or identifier. Include this if the statement or identifier to be deleted is not at the current hierarchy level.</p> <p><i>identifier</i>—(Optional) Name of the statement or identifier to delete.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">deactivate on page 272</a></li><li>• <a href="#">Deleting a Statement from a Junos OS Configuration on page 70</a></li></ul>

## edit

---

<b>Syntax</b>	<code>edit <i>statement-path</i></code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	<p>Move inside the specified statement hierarchy. If the statement does not exist, it is created.</p> <p>You cannot use the <b>edit</b> command to change the value of identifiers. You must use the <b>set</b> command.</p>
<b>Options</b>	<i>statement-path</i> —Path to the statement.
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">set on page 293</a></li><li>• <a href="#">Displaying the Current Junos OS Configuration on page 97</a></li></ul>

## exit

---

<b>Syntax</b>	exit <configuration-mode>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Exit the current level of the statement hierarchy, returning to the level prior to the last <b>edit</b> command, or exit from configuration mode. The <b>quit</b> and <b>exit</b> commands are synonyms.
<b>Options</b>	<p>none—Return to the previous edit level. If you are at the top of the statement hierarchy, exit configuration mode.</p> <p><b>configuration-mode</b>—(Optional) Exit from configuration mode.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">top on page 304</a></li><li>• <a href="#">up on page 307</a></li><li>• <a href="#">Displaying the Current Junos OS Configuration on page 97</a></li></ul>

## export-format

---

<b>Syntax</b>	<pre>export-format {   json {     ietf;   } }</pre>
<b>Hierarchy Level</b>	[edit system]
<b>Release Information</b>	Statement introduced in Junos OS Release 16.1.
<b>Description</b>	Specify the default implementation of the serialization to use for exported data in the given format. This statement only affects Junos OS configuration data that is displayed in the requested format.
<b>Options</b>	<p><b>json</b>—Define which implementation of the serialization to use for configuration data emitted in JavaScript Object Notation (JSON) format.</p> <p>Acceptable values include:</p> <ul style="list-style-type: none"><li>• <b>ietf</b>—JSON data is emitted according to the encoding rules defined in Internet drafts draft-ietf-netmod-yang-json-09, <a href="#">JSON Encoding of Data Modeled with YANG</a>, and draft-ietf-netmod-yang-metadata-06, <a href="#">Defining and Using Metadata with YANG</a>.</li></ul> <p><b>Default:</b> ietf</p>
<b>Required Privilege Level</b>	<p>maintenance—To view this statement in the configuration.</p> <p>maintenance-control—To add this statement to the configuration.</p>
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <i>Mapping Junos OS Command Output to JSON Using the CLI</i></li><li>• <i>Mapping Junos OS Configuration Statements to JSON</i></li></ul>

## groups

```
Syntax groups {
 group-name {
 configuration-data;
 when {
 chassis chassis-id;
 member member-id;
 model model-id;
 node node-id;
 peers [names of peers]
 routing-engine routing-engine-id;
 time <start-time> [to <end-time>];
 }
 conditional-data;
 }
 lccn-re0 {
 configuration-data;
 }
 lccn-re1 {
 configuration-data;
 }
 }
```

Hierarchy Level [edit]

Release Information Statement introduced before Junos OS Release 7.4.

Description Create a configuration group.

Options —

**group-name**—Name of the configuration group. To configure multiple groups, specify more than one **group-name**.

**configuration-data**—The configuration statements that are to be applied elsewhere in the configuration with the **apply-groups** statement, to have the target configuration inherit the statements in the group.

**when conditional-data**—Option introduced in Junos 11.3. The conditional statements that are to be applied when this configuration group is applied.

On routers that support multiple Routing Engines, you can also specify two special group names:

**re0**—Configuration statements that are to be applied to the Routing Engine in slot 0.

**re1**—Configuration statements that are to be applied to the Routing Engine in slot 1.

The configuration specified in group **re0** is applied only if the current Routing Engine is in slot 0; likewise, the configuration specified in group **re1** is applied only if the current Routing Engine is in slot 1. Therefore, both Routing Engines can use the same configuration file, each using only the configuration statements that apply to it. Each

**re0** or **re1** group contains at a minimum the configuration for the hostname and the management interface (**fxp0**). If each Routing Engine uses a different management interface, the group also should contain the configuration for the backup router and static routes.

(Routing matrix only) The TX Matrix router supports group names for the Routing Engines in each connected T640 router in the following formats:



**NOTE:** The management Ethernet interface used for the TX Matrix Plus router, T1600 routers in a routing matrix, and PTX Series Packet Transport Routers, is **em0**. Junos OS automatically creates the router's management Ethernet interface, **em0**.

- **lccn-re0**—Configuration statements applied to the Routing Engine in slot 0 of the specified T640 router that is connected to a TX Matrix router.
  - **lccn-re1**—Configuration statements applied to the specified to the Routing Engine in slot 1 of the specified T640 router that is connected to a TX Matrix router.
- n* identifies the T640 router and can be from 0 through 3.

The remaining statements are explained separately.

**Required Privilege Level**      configure—To enter configuration mode.

- Related Documentation**
- [Creating the Junos OS Configuration Group on page 217](#)
  - [apply-groups on page 260](#)
  - [apply-groups-except on page 261](#)

## help

---

<b>Syntax</b>	<code>help &lt;(apropos <i>string</i>   reference &lt;<i>statement-name</i>&gt;   syslog &lt;<i>syslog-tag</i>&gt;   tip cli <i>number</i>   topic &lt;<i>word</i>&gt;)&gt;</code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Display help about available configuration statements or general information about getting help.
<b>Options</b>	<p><b>apropos <i>string</i></b>—(Optional) Display statement names and help text that matches the string specified. If the string contains spaces, enclose it in quotation marks (" "). You can also specify a regular expression for the string, using standard UNIX-style regular expression syntax.</p> <p><b>reference &lt;<i>statement-name</i>&gt;</b>—(Optional) Display summary information for the statement. This information is based on summary descriptions that appear in the Junos configuration guides.</p> <p><b>syslog &lt;<i>syslog-tag</i>&gt;</b>—(Optional) Display information about system log messages.</p> <p><b>tip cli <i>number</i></b>—(Optional) Display a tip about using the CLI. Specify the number of the tip you want to view.</p> <p><b>topic &lt;<i>word</i>&gt;</b>—(Optional) Display usage guidelines for a topic or configuration statement. This information is based on subjects that appear in the Junos configuration guides.</p> <p>Entering the <b>help</b> command without an option provides introductory information about how to use the <b>help</b> command.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode.
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Getting Online Help from the Junos OS Command-Line Interface on page 47</a></li> </ul>

## insert

---

<b>Syntax</b>	insert < <i>statement-path</i> > <i>identifier1</i> (before   after) <i>identifier2</i>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Insert an identifier in to an existing hierarchy.
<b>Options</b>	<p><b>after</b>—Place <i>identifier1</i> after <i>identifier2</i>.</p> <p><b>before</b>—Place <i>identifier1</i> before <i>identifier2</i>.</p> <p><i>identifier1</i>—Existing identifier.</p> <p><i>identifier2</i>—New identifier to insert.</p> <p><i>statement-path</i>—(Optional) Path to the existing identifier.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Inserting a New Identifier in a Junos OS Configuration on page 81</a></li></ul>



## load

<b>Syntax</b>	<code>load (factory-default   merge   override   patch   replace   set   update) (<i>filename</i>   terminal) &lt;json&gt; &lt;relative&gt;</code>
<b>QFX Series</b>	<code>load (dhcp-snooping <i>filename</i>)</code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. Command introduced in Junos OS Release 11.1 for the QFX Series. Command introduced in Junos OS Release 14.1X53-D20 for the OCX Series. <code>json</code> option introduced in Junos OS Release 16.1.
<b>Description</b>	Load a configuration from an ASCII configuration file, from terminal input, or from the factory default. Your current location in the configuration hierarchy is ignored when the load operation occurs.

For information on valid filename and URL formats, see *Format for Specifying Filenames and URLs in Junos OS CLI Commands*.



**NOTE:** `load` can be run from configuration mode only.

**Options** `dhcp-snooping`—(QFX Series switches) Loads DHCP snooping entries.

**factory-default**—Loads the factory configuration. The factory configuration contains the manufacturer's suggested configuration settings. The factory configuration is the router or switch's first configuration and is loaded when the router or switch is first installed and powered on. The **factory-default** option cannot be combined with other options.



**NOTE:** To load the factory default configuration, you must first **unprotect** any protected hierarchies in the configuration.

**filename**—Name of the file to load. For information about specifying the filename, see *"Specifying Filenames and URLs" on page 176*.

**json**—(Optional) Load configuration data that uses JavaScript Object Notation (JSON) format. This option can be used with the **merge**, **override**, or **update** options.

**merge**—Combine the configuration that is currently shown in the CLI with the configuration.

**override**—Discard the entire configuration that is currently shown in the CLI and load the entire configuration. Marks every object as changed.

**patch**—Change part of the configuration and mark only those parts as changed.

**relative**—(Optional) Load the new configuration data relative to the current edit point in the configuration hierarchy.

**replace**—Look for a **replace** tag in *filename*, delete the existing statement of the same name, and replace it with the configuration.

**set**—Merge a set of commands with an existing configuration. This option executes the configuration instructions line by line as they are stored in a file or from a terminal. The instructions can contain any configuration mode command, such as **set**, **edit**, **exit**, and **top**.

**terminal**—Use the text you type at the terminal as input to the configuration. Type Ctrl+d to end terminal input.

**update**—Discard the entire configuration that is currently shown in the CLI, and load the entire configuration. Marks changed objects only.



**NOTE:** If you are using Junos OS in a Common Criteria environment, system log messages are created whenever a secret attribute is changed (for example, password changes or changes to the RADIUS shared secret). These changes are logged during the following configuration load operations:

```
load merge
load replace
load override
load update
```

For more information, see the *Secure Configuration Guide for Common Criteria and Junos-FIPS*.

---

<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Loading a Configuration from a File or the Terminal on page 144</a></li></ul>

## maximum-aggregate-pool (Batch Commits)

<b>Syntax</b>	<code>maximum-aggregate-pool <i>maximum-number-of-commits-to-aggregate</i>;</code>
<b>Hierarchy Level</b>	[edit system commit server], [edit system commit synchronize server]
<b>Release Information</b>	Statement introduced in Junos OS Release 12.1.
<b>Description</b>	For Junos OS batch commits, specify the maximum number of individual commit operations that are aggregated or merged into a single commit operation.
<b>Options</b>	<p><i>maximum-number-of-commits-to-aggregate</i>—Maximum number of individual commit operations that are aggregated or merged into a single commit operation.</p> <p><b>Range:</b> 1 through 4294967295</p> <p><b>Default:</b> 5</p>
<b>Required Privilege Level</b>	<p>system—To view this statement in the configuration.</p> <p>system-control—To add this statement to the configuration.</p>
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Example: Configuring Batch Commit Server Properties on page 117</a></li> </ul>

## maximum-entries (Batch Commits)

<b>Syntax</b>	<code>maximum-entries <i>number-of-entries</i>;</code>
<b>Hierarchy Level</b>	[edit system commit server], [edit system commit synchronize server]
<b>Release Information</b>	Statement introduced in Junos OS Release 12.1.
<b>Description</b>	For Junos OS batch commits, specify the maximum number of commit jobs that are included in the commit queue.
<b>Options</b>	<i>number-of-entries</i> —Maximum number of commit jobs that are included in the commit queue.
<b>Required Privilege Level</b>	<p>system—To view this statement in the configuration.</p> <p>system-control—To add this statement to the configuration.</p>
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Example: Configuring Batch Commit Server Properties on page 117</a></li> </ul>

## no-hidden-commands

---

<b>Syntax</b>	no-hidden-commands;
<b>Hierarchy Level</b>	[edit system]
<b>Release Information</b>	Statement introduced in Junos OS Release 16.1 EX Series, M Series and MX Series.
<b>Description</b>	Hidden commands are Junos OS commands that are not published but could be run on a router. Hidden command serve a specific purpose but for most part are not expected to be used by the customers and are not supported. The <b>no-hidden-commands</b> statement allows the user to block all hidden commands to all users except root.
<b>Default</b>	Hidden commands are enabled by default.
<b>Required Privilege Level</b>	admin—To view this statement in the configuration. admin-control—To add this statement to the configuration.

## protect

---

<b>Syntax</b>	<code>protect (hierarchy   statement   identifier)</code>
<b>Release Information</b>	Command introduced in Junos OS Release 11.2.
<b>Description</b>	Protect a hierarchy, statement, or identifier from modification or deletion.
<b>Options</b>	none
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Example: Protecting the Junos OS Configuration from Modification or Deletion on page 151</a></li></ul>

## quit

---

<b>Syntax</b>	quit <configuration-mode>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Exit the current level of the statement hierarchy, returning to the level prior to the last <b>edit</b> command, or exit from configuration mode. The <b>quit</b> and <b>exit</b> commands are synonyms.
<b>Options</b>	<p>none—Return to the previous edit level. If you are at the top of the statement hierarchy, exit configuration mode.</p> <p><b>configuration-mode</b>—(Optional) Exit from configuration mode.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">top on page 304</a></li><li>• <a href="#">up on page 307</a></li><li>• <a href="#">Displaying the Current Junos OS Configuration on page 97</a></li></ul>

## rename

**Syntax** `rename <statement-path> identifier1 to identifier2`

**Release Information** Command introduced before Junos OS Release 7.4.

**Description** Rename an existing configuration statement or identifier.

**Options** *identifier1*—Existing identifier to rename.

*identifier2*—New name of identifier.

*statement-path*—(Optional) Path to an existing statement or identifier.



**NOTE:** For example, to rename interface `ge-0/0/0.0` to `ge-0/0/10.0` at the following hierarchy level:

```
logical-systems {
 logical-system-abc {
 (...)
 protocols {
 ospf {
 area 0.0.0.0 {
 interface ge-0/1/0.0;
```

Issue the following command:

```
rename logical-systems logical-system-abc protocols ospf area 0.0.0.0 interface
ge-0/1/0.0.0 to interface ge-0/1/10.0
```

**Required Privilege Level** `configure`—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.

**Related Documentation**

- [Renaming an Identifier in a Junos OS Configuration on page 76](#)

## replace

---

<b>Syntax</b>	replace pattern <i>pattern1</i> with <i>pattern2</i> <upto <i>n</i> >
<b>Release Information</b>	Command introduced in Junos OS Release 7.6.
<b>Description</b>	Replace identifiers or values in a configuration. For more information, refer to <a href="#">KB30332</a> .
<b>Options</b>	<p><i>pattern1</i>—Text string or regular expression that defines the identifiers or values you want to match.</p> <p><i>pattern2</i>—Text string or regular expression that replaces the identifiers and values located with <i>pattern1</i>.</p> <p>Juniper Networks uses standard UNIX-style regular expression syntax (as defined in POSIX 1003.2). If the regular expression contains spaces, operators, or wildcard characters, enclose the expression in quotation marks. Greedy qualifiers (match as much as possible) are supported. Lazy qualifiers (match as little as possible) are not.</p> <p><b>upto <i>n</i></b>—Number of objects replaced. The value of <i>n</i> controls the total number of objects that are replaced in the configuration (not the total number of times the pattern occurs). Objects at the same hierarchy level (siblings) are replaced first. Multiple occurrences of a pattern within a given object are considered a single replacement. If you do not specify an <b>upto</b> option, all identifiers and values in the configuration that match <i>pattern1</i> are replaced.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Using Global Replace in the Junos OS Configuration on page 205</a></li></ul>



## rollback

<b>Syntax</b>	<code>rollback &lt;number   rescue&gt;</code>
<b>Release Information</b>	<p>Command introduced before Junos OS Release 7.4.</p> <p>Command introduced in Junos OS Release 11.1 for the QFX Series.</p>
<b>Description</b>	<p>Return to a previously committed configuration. The software saves the last 50 committed configurations, including the rollback number, date, time, and name of the user who issued the <b>commit</b> configuration command.</p> <p>The currently operational Junos OS configuration is stored in the file <b>juniper.conf</b>, and the last three committed configurations are stored in the files <b>juniper.conf.1</b>, <b>juniper.conf.2</b>, and <b>juniper.conf.3</b>. These four files are located in the directory <b>/config</b>, which is on the router's flash drive. The remaining 46 previous committed configurations, the files <b>juniper.conf.4</b> through <b>juniper.conf.49</b>, are stored in the directory <b>/var/db/config</b>, which is on the router's hard disk.</p> <p>During rollback, the configuration you specify is loaded from the associated file. Only objects in the rollback configuration that differ from the previously loaded configuration are marked as changed (equivalent to <b>load update</b>).</p>
<b>Options</b>	<p><b>none</b> (Optional)—Return to the most recently saved configuration.</p> <p><b>number</b>—(Optional) Configuration to return to. The range of values is from <b>0</b> through <b>49</b>. The most recently saved configuration is number <b>0</b>, and the oldest saved configuration is number <b>49</b>. The default is <b>0</b>.</p> <p><b>rescue</b>—(Optional) Return to the rescue configuration.</p>
<b>Required Privilege Level</b>	rollback—To roll back to configurations other than the one most recently committed.
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Returning to a Previously Committed Junos OS Configuration on page 135</a></li> <li>• <a href="#">Creating and Returning to a Rescue Configuration on page 138</a></li> </ul>

## run

---

<b>Syntax</b>	<code>run <i>command</i></code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Run a top-level CLI command without exiting from configuration mode.
<b>Options</b>	<i>command</i> —CLI top-level command.
<b>Required Privilege Level</b>	configure—To enter configuration mode.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Understanding Junos OS CLI Configuration Mode on page 56</a></li></ul>

## save

<b>Syntax</b>	<code>save <i>filename</i></code>
<b>QFX Series</b>	<code>save (dhcp-snooping <i>filename</i>)</code>
<b>Release Information</b>	<p>Command introduced before Junos OS Release 7.4.</p> <p>Command introduced in Junos OS Release 11.1 for the QFX Series.</p> <p>Command introduced in Junos OS Release 14.1X53-D20 for the OCX Series.</p>
<b>Description</b>	<p>Save the configuration to an ASCII file. The contents of the current level of the statement hierarchy (and below) are saved, along with the statement hierarchy containing it. This allows a section of the configuration to be saved, while fully specifying the statement hierarchy.</p> <p>For information on valid filename and URL formats, see <i>Format for Specifying Filenames and URLs in Junos OS CLI Commands</i>.</p> <p>When saving a file to a remote system, the software uses the <b>scp/ssh</b> protocol.</p>
<b>Options</b>	<p><b><i>filename</i></b>—Name of the saved file. You can specify a filename in one of the following ways:</p> <ul style="list-style-type: none"> <li>• <b><i>filename</i></b>—File in the user's home directory (the current directory) on the local flash drive.</li> <li>• <b><i>path/filename</i></b>—File on the local flash drive.</li> <li>• <b><i>/var/filename</i></b> or <b><i>/var/path/filename</i></b>—File on the local hard disk.</li> <li>• <b><i>a:filename</i></b> or <b><i>a:path/filename</i></b>—File on the local drive. The default path is <b>/</b> (the root-level directory). The removable media can be in MS-DOS or UNIX (UFS) format.</li> <li>• <b><i>hostname:/path/filename</i></b>, <b><i>hostname:filename</i></b>, <b><i>hostname:path/filename</i></b>, or <b><i>scp://hostname/path/filename</i></b>—File on an <b>scp/ssh</b> client. This form is not available in the worldwide version of Junos OS. The default path is the user's home directory on the remote system. You can also specify <b><i>hostname</i></b> as <b><i>username@hostname</i></b>.</li> <li>• <b><i>ftp://hostname/path/filename</i></b>—File on an FTP server. You can also specify <b><i>hostname</i></b> as <b><i>username @hostname</i></b> or <b><i>username:password @hostname</i></b>. The default path is the user's home directory. To specify an absolute path, the path must start with the string <b>%2F</b>; for example, <b><i>ftp://hostname/%2Fpath/filename</i></b>. To have the system prompt you for the password, specify <b><i>prompt</i></b> in place of the password. If a password is required, and you do not specify the password or <b><i>prompt</i></b>, an error message is displayed:           <pre>user@host&gt; file copy ftp://username@ftp.hostname.net//filename file copy ftp.hostname.net: Not logged in. user@host&gt; file copy ftp://username:prompt@ftphostname.net//filename</pre> <p>Password for <b><i>username@ftp.hostname.net</i></b>:</p> </li> <li>• <b><i>http://hostname/path/filename</i></b>—File on a Hypertext Transfer Protocol (HTTP) server. You can also specify <b><i>hostname</i></b> as <b><i>username@hostname</i></b> or</li> </ul>

*username:password@hostname*. If a password is required and you omit it, you are prompted for it.

- *re0:/path/filename* or *re1:/path/filename*—File on a local Routing Engine.

**Required Privilege Level**      configure—To enter configuration mode.

**Related Documentation**      • [Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration on page 89](#)

---

## server (Batch Commits)

---

**Syntax**      server {  
                commit-interval <number-of-seconds-between-commits>;  
                days-to-keep-error-logs <days-to-keep-error-log-entries>;  
                maximum-aggregate-pool <maximum-number-of-commits-to-aggregate>;  
                maximum-entries <number-of-entries>;  
                traceoptions {  
                    file *filename*;  
                    files *number*;  
                    flag (all | batch | commit-server | configuration);  
                    size *maximum-file-size*;  
                    (world-readable | no-world-readable);  
                }  
            }

**Hierarchy Level**      [edit system commit]

**Release Information**      Statement introduced in Junos OS Release 12.1.

**Description**      Configure the system commit to occur in batches. Configure parameters for aggregating and saving batch commits.

**Options**      **commit-interval**—Configure the interval between commits.

**days-to-keep-error-logs**—Configure the number of days to keep log entries.

**maximum-aggregate-pool**—Configure the maximum number of commits to aggregate together.

**maximum-entries** —Configure the maximum number of commit entries.

**Required Privilege Level**      system—To view this statement in the configuration.  
                                        system-control—To add this statement to the configuration.

**Related Documentation**      • [Example: Configuring Batch Commit Server Properties on page 117](#)

## set

---

<b>Syntax</b>	<code>set &lt;<i>statement-path</i>&gt; <i>identifier</i></code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Create a statement hierarchy and set identifier values. This is similar to <b>edit</b> except that your current level in the hierarchy does not change.
<b>Options</b>	<p><i>identifier</i>—Name of the statement or identifier to set.</p> <p><i>statement-path</i>—(Optional) Path to an existing statement hierarchy level. If that hierarchy level does not exist, it is created.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">edit on page 274</a></li><li>• <a href="#">Displaying the Current Junos OS Configuration on page 97</a></li></ul>

## show

---

<b>Syntax</b>	<code>show &lt;statement-path&gt; &lt;identifier&gt;</code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Display the current configuration.
<b>Options</b>	<p><code>none</code>—Display the entire configuration at the current hierarchy level.</p> <p><i>identifier</i>—(Optional) Display the configuration for the specified identifier.</p> <p><i>statement-path</i>—(Optional) Display the configuration for the specified statement hierarchy path.</p>
<b>Required Privilege Level</b>	<code>configure</code> —To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">show   display inheritance on page 298</a></li><li>• <a href="#">show   display omit on page 299</a></li><li>• <a href="#">show   display set on page 300</a></li><li>• <a href="#">show   display set relative on page 301</a></li><li>• <a href="#">show groups junos-defaults on page 302</a></li><li>• <a href="#">Displaying the Current Junos OS Configuration on page 97</a></li></ul>

## show configuration

---

<b>Syntax</b>	show configuration <statement-path>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. Command introduced in Junos OS Release 9.0 for EX Series switches.
<b>Description</b>	Display the configuration that currently is running on the router or switch, which is the last committed configuration.
<b>Options</b>	<p><b>none</b>—Display the entire configuration.</p> <p><b>statement-path</b>—(Optional) Display one of the following hierarchies in a configuration. (Each <b>statement-path</b> option has additional suboptions not described here. See the appropriate feature guide or EX Series switch documentation for more information.)</p> <ul style="list-style-type: none"> <li>• <b>access</b>—Network access configuration.</li> <li>• <b>access-profile</b>—Access profile configuration.</li> <li>• <b>accounting-options</b>—Accounting data configuration.</li> <li>• <b>applications</b>—Applications defined by protocol characteristics.</li> <li>• <b>apply-groups</b>—Groups from which configuration data is inherited.</li> <li>• <b>chassis</b>—Chassis configuration.</li> <li>• <b>chassis network-services</b>—Current running mode.</li> <li>• <b>class-of-service</b>—Class-of-service configuration.</li> <li>• <b>diameter</b>—Diameter base protocol layer configuration.</li> <li>• <b>ethernet-switching-options</b>—(EX Series switch only) Ethernet switching configuration.</li> <li>• <b>event-options</b>—Event processing configuration.</li> <li>• <b>firewall</b>—Firewall configuration.</li> <li>• <b>forwarding-options</b>—Options that control packet sampling.</li> <li>• <b>groups</b>—Configuration groups.</li> <li>• <b>interfaces</b>—Interface configuration.</li> <li>• <b>jsrc</b>—JSRC partition configuration.</li> <li>• <b>jsrc-partition</b>—JSRC partition configuration.</li> <li>• <b>logical-systems</b>—Logical system configuration.</li> <li>• <b>poe</b>—(EX Series switch only) Power over Ethernet configuration.</li> <li>• <b>policy-options</b>—Routing policy option configuration.</li> <li>• <b>protocols</b>—Routing protocol configuration.</li> </ul>

- **routing-instances**—Routing instance configuration.
- **routing-options**—Protocol-independent routing option configuration.
- **security**—Security configuration.
- **services**—Service PIC applications configuration.
- **snmp**—Simple Network Management Protocol configuration.
- **system**—System parameters configuration.
- **virtual-chassis**—(EX Series switch only) Virtual Chassis configuration.
- **vlan**—(EX Series switch only) VLAN configuration.

**Additional Information** The portions of the configuration that you can view depend on the user class that you belong to and the corresponding permissions. If you do not have permission to view a portion of the configuration, the text **ACCESS-DENIED** is substituted for that portion of the configuration. If you do not have permission to view authentication keys and passwords in the configuration, because the **secret** permission bit is not set for your user account, the text **SECRET-DATA** is substituted for that portion of the configuration. If an identifier in the configuration contains a space, the identifier is displayed in quotation marks.

Likewise, when you issue the **show configuration** command with the **| display set** pipe option to view the configuration as **set** commands, those portions of the configuration that you do not have permissions to view are substituted with the text **ACCESS-DENIED**.

**Required Privilege Level** view

**Related Documentation**

- [Displaying the Current Junos OS Configuration on page 97](#)
- [Overview of Junos OS CLI Operational Mode Commands on page 163](#)

**List of Sample Output** [show configuration on page 296](#)  
[show configuration policy-options on page 297](#)

**Output Fields** This command displays information about the current running configuration.

## Sample Output

### show configuration

```
user@host> show configuration
Last commit: 2006-10-31 14:13:00 PST by user1 version "8.2I0 [userb]"; ## last
changed: 2006-10-31 14:05:53 PST
system {
 host-name exhost;
 domain-name ex1.net;
 backup-router 198.51.100.254;
 time-zone America/Los_Angeles;
 default-address-selection;
 name-server {
 192.0.2.254;
 192.0.2.249;
```



```

 192.0.2.176;
 }
 services {
 telnet;
 }
 tacplus-server {
 10.2.3.4 {
 secret /* SECRET-DATA */;
 ...
 }
 }
}
interfaces {
 ...
}
protocols {
 isis {
 export "direct routes";
 }
}
policy-options {
 policy-statement "direct routes" {
 from protocol direct;
 then accept;
 }
}

```

#### show configuration policy-options

```

user@host> show configuration policy-options
policy-options {
 policy-statement "direct routes" {
 from protocol direct;
 then accept;
 }
}

```

## show | display inheritance

---

**Syntax**    show | display inheritance <brief | defaults | no-comments | terse>

**Release Information**    Command introduced before Junos OS Release 7.4.

**Description**    Show the inherited configuration data and information about the source group from which the configuration has been inherited. Show interface ranges configuration data in expanded format and information about the source interface-range from which the configuration has been expanded

```
user@host# show system ports | display inheritance defaults
'console' was inherited from group 'junos-defaults'
'vt100' was inherited from group 'junos-defaults'
console type vt100;
```

```
user@host# show system login class readonly | display inheritance
'interface' was inherited from group global'
'network' was inherited from group global'
'routing' was inherited from group global'
'system' was inherited from group global'
'trace' was inherited from group global'
'view' was inherited from group global'
##
permissions [interface network routing system trace view];
```

```
user@host# show system login class readonly | display inheritance no-comments
permissions [interface network routing system trace view];
```

- Options**
- **brief**—Display brief output for the command.
  - **defaults**—Display the Junos OS defaults that have been applied to the configuration.
  - **no-comments**—Display configuration information without inline comments marked with ##.
  - **terse**—Display terse output with inheritance details as inline comment.

**Required Privilege Level**    view

**Related Documentation**

- [Using Junos OS Defaults Groups on page 242](#)

## show | display omit

---

**Syntax**    show | display omit

**Release Information**    Command introduced in Junos OS Release 8.2.

**Description**    Display configuration statements (including those marked as hidden by the **apply-flags omit** configuration statement).

```
user@host# show | display omit
system {
 apply-flags omit;
 login {
 message lengthy-login-message;
 }
}
```

**Required Privilege Level**    view

**Related Documentation**    • [show on page 294](#)

## show | display set

---

<b>Syntax</b>	show   display set
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	<p>Display the configuration as a series of configuration mode commands required to re-create the configuration from the top level of the hierarchy as <b>set</b> commands</p> <pre>user@host# show   display set set interfaces fe-0/0/0 unit 0 family inet address 192.168.1.230/24 set interfaces fe-0/0/0 unit 0 family iso set interfaces fe-0/0/0 unit 0 family mpls set interfaces fe-0/0/0 unit 1 family inet address 10.0.0.1/8 deactivate interfaces fe-0/0/0 unit 1</pre>
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">show on page 294</a></li><li>• <a href="#">Displaying set Commands from the Junos OS Configuration on page 101</a></li></ul>

## show | display set relative

<b>Syntax</b>	show   display set relative
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	<p>Display the configuration as a series of configuration mode commands required to re-create the configuration from the current hierarchy level.</p> <pre>[edit interfaces fe-0/0/0] user@host# show unit 0 {   family inet {     address 192.107.1.230/24;   }   family iso;   family mpls; } inactive: unit 1 {   family inet {     address 10.0.0.1/8;   } } user@host# show   display set relative set unit 0 family inet address 192.107.1.230/24 set unit 0 family iso set unit 0 family mpls set unit 1 family inet address 10.0.0.1/8 deactivate unit 1</pre>
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Displaying set Commands from the Junos OS Configuration on page 101</a></li> </ul>

## show groups junos-defaults

---

**Syntax**    show groups junos-defaults

**Release Information**    Command introduced before Junos OS Release 7.4.

**Description**    Display the full set of available preset statements from the Junos OS defaults group.

```
user@host# show groups junos-defaults
groups {
 junos-defaults {
 applications {
 # File Transfer Protocol
 application junos-ftp {
 application-protocol ftp;
 protocol tcp;
 destination-port 21;
 }
 # Trivial File Transfer Protocol
 application junos-tftp {
 application-protocol tftp;
 protocol udp;
 destination-port 69;
 }
 # RPC port mapper on TCP
 application junos-rpc-portmap-tcp {
 application-protocol rpc-portmap;
 protocol tcp;
 destination-port 111;
 }
 # RPC port mapper on UDP
 }
 }
}
```

**Required Privilege Level**    view

**Related Documentation**    • [Using Junos OS Defaults Groups.](#)

## status

---

<b>Syntax</b>	status
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Display the users currently editing the configuration.
<b>Required Privilege Level</b>	configure—To enter configuration mode. <ul style="list-style-type: none"><li>• <a href="#">“Displaying Users Currently Editing the Junos OS Configuration” on page 104.</a></li></ul>

## top

---

<b>Syntax</b>	<code>top &lt;configuration-command&gt;</code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Return to the top level of configuration command mode, which is indicated by the <b>[edit]</b> banner.
<b>Options</b>	<i>configuration-command</i> —(Optional) Issue configuration mode commands from the top of the hierarchy.
<b>Required Privilege Level</b>	configure—To enter configuration mode.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Displaying the Current Junos OS Configuration on page 97</a></li><li>• <a href="#">exit on page 275</a></li><li>• <a href="#">up on page 307</a></li></ul>



## traceoptions (Batch Commits)

<b>Syntax</b>	<pre> traceoptions {     file <i>filename</i>;     files <i>number</i>;     flag (all   batch   commit-server   configuration);     size <i>maximum-file-size</i>;     (world-readable   no-world-readable); } </pre>
<b>Hierarchy Level</b>	<pre> [edit system commit server], [edit system commit synchronize server] </pre>
<b>Release Information</b>	Statement introduced in Junos OS Release 12.1.
<b>Description</b>	For Junos OS batch commits, configure tracing operations.
<b>Options</b>	<b>file <i>name</i></b> —Name of the file to receive the output of the tracing operation.



**NOTE:** If you configure traceoptions and do not explicitly specify a filename for logging the events, the batch commit events are logged in the commitd file (var/log/commitd) by default.

**files *number***—Maximum number of trace files.

**flag *flag***—Tracing operation to perform. To specify more than one tracing operation, include multiple **flag** statements. You can include the following flags:

- **all**—All tracing operations flags.
- **batch**—Tracing operations for batch events.
- **commit-server**—Tracing operations for commit server events.
- **configuration**—Tracing operations for the reading of configuration.

**size**—Maximum size of each trace file, in kilobytes (KB), megabytes (MB), or gigabytes (GB).

**world-readable | no-world-readable**—**readable**—Grant all users permission to read archived log files, or restrict the permission only to the root user and users who have the Junos OS maintenance permission.

<b>Required Privilege Level</b>	system—To view this statement in the configuration. system-control—To add this statement to the configuration.
---------------------------------	-------------------------------------------------------------------------------------------------------------------

<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Example: Configuring Batch Commit Server Properties on page 117</a></li> </ul>
------------------------------	-------------------------------------------------------------------------------------------------------------------------------------

## unprotect

---

<b>Syntax</b>	unprotect ( <i>hierarchy</i>   <i>statement</i>   <i>identifier</i> )
<b>Release Information</b>	Command introduced in Junos OS Release 11.2.
<b>Description</b>	Unprotect a protected hierarchy, configuration statement, or an identifier.
<b>Options</b>	none
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">top on page 304</a></li><li>• <a href="#">up on page 307</a></li><li>• <a href="#">Displaying the Current Junos OS Configuration on page 97</a></li></ul>

## up

---

<b>Syntax</b>	<code>up &lt;number&gt; &lt;configuration-command&gt;</code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Move up one level in the statement hierarchy.
<b>Options</b>	<p>none—Move up one level in the configuration hierarchy.</p> <p><i>configuration-command</i>—(Optional) Issue configuration mode commands from a location higher in the hierarchy.</p> <p><i>number</i>—(Optional) Move up the specified number of levels in the configuration hierarchy.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Displaying the Current Junos OS Configuration on page 97</a></li><li>• <a href="#">exit on page 275</a></li><li>• <a href="#">top on page 304</a></li></ul>

## update

---

**Syntax**    update

**Release Information**    Command introduced in Junos OS Release 7.5.

**Description**    Update private candidate configuration with a copy of the most recently committed configuration, including your private changes.



**NOTE:** The `update` command is available only when you are in configure private mode.

---

**Required Privilege Level**    configure—To enter configuration mode.

**Related Documentation**

- [Updating the configure private Configuration on page 96.](#)

## when

<b>Syntax</b>	<pre> when {   chassis <i>chassis-id</i>;   member <i>member-id</i>;   model <i>model-id</i>;   node [ <i>names of peers</i> ]<i>node-id</i>;   peers [ <i>names of peers</i> ];   routing-engine <i>routing-engine-id</i>;   time &lt;<i>start-time</i>&gt; [to &lt;<i>end-time</i>&gt;]; } </pre>
<b>Hierarchy Level</b>	[edit groups <i>group-name</i> ]
<b>Release Information</b>	<p>Statement introduced in Junos OS Release 11.3.</p> <p><b>peers</b> option added in Junos OS Release 14.2R6 for the MX Series.</p> <p><b>peers</b> option added in Junos OS Release 16.1R1 for the EX Series.</p>
<b>Description</b>	<p>Define conditions under which the configuration group should be applied. Conditions include the type of chassis, model, or Routing Engine, virtual chassis member, cluster node, and start and optional end time of day. If you specify multiple conditions in a single configuration group, all conditions must be met before the configuration group is applied.</p>
<b>Options</b>	<p><b>chassis</b> <i>chassis-id</i>—Specify the chassis type of the router. Valid types include SCC0, SCC1, LCC0, LCC1 ... LCC3.</p> <p><b>member</b> <i>member-id</i>—Specify the name of the member of the virtual chassis.</p> <p><b>model</b> <i>model-id</i>—Specify the model name of the router, such as m7i or tx100.</p> <p><b>node</b> <i>node-id</i>—Specify the cluster node.</p> <p><b>peers</b> [ <i>names of peers</i> ]—Specify the names of the MC-LAG peers participating in commit synchronization.</p> <p><b>routing-engine</b> <i>routing-engine-id</i>—Specify the type of Routing Engine, re0 or re1.</p> <p><b>time</b> &lt;<i>start-time</i>&gt; [to &lt;<i>end-time</i>&gt;]—Specify the start time or time duration for this configuration group to be applied. If only the start time is specified, the configuration group is applied at the specified time and remains in effect until the time is changed. If the end time is specified, then on each day, the applied configuration group is started and stopped at the specified times. The syntax for specifying the time is: <b>time</b> &lt;<i>start-time</i>&gt; [to &lt;<i>end-time</i>&gt;] using the time format yyyy-mm-dd.hh:mm, hh:mm, or hh.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode.
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Creating the Junos OS Configuration Group on page 217</a></li> <li>• <a href="#">apply-groups on page 260</a></li> </ul>

- [apply-groups-except on page 261](#)
- [groups on page 277](#)

## wildcard delete

---

<b>Syntax</b>	<code>wildcard delete &lt;statement-path&gt; &lt;identifier&gt; &lt;regular-expression&gt;</code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	<p>Delete a statement or identifier. All subordinate statements and identifiers contained within the specified statement path are deleted with it.</p> <p>Deleting a statement or an identifier effectively “unconfigures” or disables the functionality associated with that statement or identifier.</p> <p>If you do not specify <i>statement-path</i> or <i>identifier</i>, the entire hierarchy starting at the current hierarchy level is removed.</p>
<b>Options</b>	<p><i>identifier</i>—(Optional) Name of the statement or identifier to delete.</p> <p><i>regular-expression</i>—(Optional) The pattern based on which you want to delete multiple items. When you use the <b>wildcard</b> command to delete related configuration items, the <i>regular-expression</i> must be the final statement.</p> <p><i>statement-path</i>—(Optional) Path to an existing statement or identifier. Include this if the statement or identifier to be deleted is not at the current hierarchy level.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode. Other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Example: Using Global Replace in a Junos OS Configuration—Using the upto Option on page 210.</a></li></ul>





## CHAPTER 13

# Junos OS CLI Environment Commands

- `set cli complete-on-space`
- `set cli directory`
- `set cli idle-timeout`
- `set cli prompt`
- `set cli restart-on-upgrade`
- `set cli screen-length`
- `set cli screen-width`
- `set cli terminal`
- `set cli timestamp`
- `set date`
- `show cli`
- `show cli`
- `show cli authorization`
- `show cli directory`
- `show cli history`

## set cli complete-on-space

---

<b>Syntax</b>	set cli complete-on-space (off   on)
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. Command introduced in Junos OS Release 9.0 for EX Series switches.
<b>Description</b>	Set the command-line interface (CLI) to complete a partial command entry when you type a space or a tab. This is the default behavior of the CLI.
<b>Options</b>	<b>off</b> —Turn off command completion. <b>on</b> —Allow either a space or a tab to be used for command completion.
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <i>CLI User Interface Overview</i></li><li>• <a href="#">show cli on page 324</a></li></ul>
<b>List of Sample Output</b>	<a href="#">set cli complete-on-space on page 314</a>
<b>Output Fields</b>	When you enter this command, you are provided feedback on the status of your request.

## Sample Output

### set cli complete-on-space

In the following example, pressing the Spacebar changes the partial command entry from **com** to **complete-on-space**. The example shows how adding the keyword **off** at the end of the command disables command completion.

```
user@host> set cli com<Space>
user@host>set cli complete-on-space off
Disabling complete-on-space
```

---

## set cli directory

---

<b>Syntax</b>	set cli directory <i>directory</i>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. Command introduced in Junos OS Release 9.0 for EX Series switches.
<b>Description</b>	Set the current working directory.
<b>Options</b>	<i>directory</i> —Pathname of the working directory.
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <i>CLI User Interface Overview</i></li><li>• <i>show cli directory</i></li></ul>
<b>List of Sample Output</b>	<a href="#">set cli directory on page 315</a>
<b>Output Fields</b>	When you enter this command, you are provided feedback on the status of your request.

### Sample Output

#### set cli directory

```
user@host> set cli directory /var/tmp
Current directory: /var/tmp
```

## set cli idle-timeout

---

<b>Syntax</b>	set cli idle-timeout < <i>minutes</i> >
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. Command introduced in Junos OS Release 9.0 for EX Series switches.
<b>Description</b>	Set the maximum time that an individual session can be idle before the user is logged off the router or switch.
<b>Options</b>	<i>minutes</i> —(Optional) Maximum idle time. The range of values, in minutes, is 0 through 100,000. If you do not issue this command, and the user's login class does not specify this value, the user is never forced off the system after extended idle times. Setting the value to 0 disables the timeout.
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <i>CLI User Interface Overview</i></li><li>• <a href="#">show cli on page 324</a></li></ul>
<b>List of Sample Output</b>	<a href="#">set cli idle-timeout on page 316</a>
<b>Output Fields</b>	When you enter this command, you are provided feedback on the status of your request.

### Sample Output

#### set cli idle-timeout

```
user@host> set cli idle-timeout 60
Idle timeout set to 60 minutes
```

---

## set cli prompt

---

<b>Syntax</b>	set cli prompt <i>string</i>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. Command introduced in Junos OS Release 9.0 for EX Series switches.
<b>Description</b>	Set the prompt so that it is displayed within the CLI.
<b>Options</b>	<b>string</b> —CLI prompt string. To include spaces in the prompt, enclose the string in quotation marks. By default, the string is <i>username@hostname</i> .
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <i>CLI User Interface Overview</i></li><li>• <a href="#">show cli on page 324</a></li></ul>
<b>List of Sample Output</b>	<a href="#">set cli prompt on page 317</a>
<b>Output Fields</b>	When you enter this command, the new CLI prompt is displayed.

### Sample Output

#### set cli prompt

```
user@host> set cli prompt lab1-router>
lab1-router>
```

## set cli restart-on-upgrade

---

<b>Syntax</b>	set cli restart-on-upgrade string (off   on)
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. Command introduced in Junos OS Release 9.0 for EX Series switches.
<b>Description</b>	For an individual session, set the CLI to prompt you to restart the router or switch after upgrading the software.
<b>Options</b>	<b>off</b> —Disables the prompt. <b>on</b> —Enables the prompt.
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <i>CLI User Interface Overview</i></li><li>• <a href="#">show cli on page 324</a></li></ul>
<b>List of Sample Output</b>	<a href="#">set cli restart-on-upgrade on page 318</a>
<b>Output Fields</b>	When you enter this command, you are provided feedback on the status of your request.

### Sample Output

#### set cli restart-on-upgrade

```
user@host> set cli restart-on-upgrade on
Enabling restart-on-upgrade
```

---

## set cli screen-length

---

<b>Syntax</b>	set cli screen-length <i>length</i>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	<p>Set terminal screen length.</p> <pre>user@host&gt; set cli screen-length 75 Screen Length set to 75</pre>
<b>Options</b>	<p><b><i>length</i></b>—Number of lines of text that the terminal screen displays. The range of values, in number of lines, is 24 through 100,000. The default is 24.</p> <p>The point at which the ---(<b>more</b>)--- prompt appears on the screen is a function of this setting and the settings for the <b>set cli screen-width</b> and <b>set cli terminal</b> commands.</p>
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Setting the Screen Length on page 250</a></li><li>• <a href="#">Setting the Junos OS CLI Screen Length and Width on page 249</a></li><li>• <a href="#">set cli screen-width on page 320</a></li><li>• <a href="#">set cli terminal on page 321</a></li><li>• <a href="#">show cli on page 326</a></li></ul>

## set cli screen-width

---

**Syntax**    `set cli screen-width width`

**Release Information**    Command introduced before Junos OS Release 7.4.  
Command introduced in Junos OS Release 9.0 for EX Series switches.

**Description**    Set the terminal screen width.

```
user@host> set cli screen-width
Screen width set to 132
```

**Options**    *width*—Number of characters in a line. The value is **0** or in the range of **40** through **1024**.  
The default value is **80**.



**NOTE:** In Junos OS Release 13.2 and earlier, the value of *width* is in the range of **0** through **1024**.

---

**Required Privilege Level**    view

**Related Documentation**

- [Setting the Screen Width on page 250](#)
- [set cli screen-length on page 319](#)
- [set cli terminal on page 321](#)
- [show cli on page 326](#)




## set cli terminal

---

<b>Syntax</b>	set cli terminal <i>terminal-type</i>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	<p>Set the terminal type.</p> <pre>user@host&gt; set cli terminal xterm</pre>
<b>Options</b>	<p><i>terminal-type</i>—Type of terminal that is connected to the Ethernet management port:</p> <ul style="list-style-type: none"><li>• <b>ansi</b>—ANSI-compatible terminal (80 characters by 24 lines)</li><li>• <b>small-xterm</b>—Small xterm window (80 characters by 24 lines)</li><li>• <b>vt100</b>—VT100-compatible terminal (80 characters by 24 lines)</li><li>• <b>xterm</b>—Large xterm window (80 characters by 65 lines)</li></ul>
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Setting the Terminal Type on page 248</a></li></ul>

## set cli timestamp

---

<b>Syntax</b>	set cli timestamp (format <i>timestamp-format</i>   disable)
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. Command introduced in Junos OS Release 9.0 for EX Series switches.
<b>Description</b>	Set a timestamp for CLI output.
<b>Options</b>	<p><b>format <i>timestamp-format</i></b>—Set the date and time format for the timestamp. The timestamp format you specify can include the following placeholders in any order:</p> <ul style="list-style-type: none"><li>• <b>%m</b>—Two-digit month</li><li>• <b>%d</b>—Two-digit date</li><li>• <b>%T</b>—Six-digit hour, minute, and seconds</li></ul> <p><b>disable</b>—Remove the timestamp from the CLI.</p>
<div> <b>NOTE:</b> A timestamp is displayed by default when no command output is generated.</div>	
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <i>CLI User Interface Overview</i></li><li>• <a href="#">show cli on page 324</a></li></ul>
<b>List of Sample Output</b>	<a href="#">set cli timestamp on page 322</a>
<b>Output Fields</b>	When you enter this command, you are provided feedback on the status of your request.

### Sample Output

#### set cli timestamp

```
user@host> set cli timestamp format '%m-%d-%T'
'04-21-17:39:13'
CLI timestamp set to: '%m-%d-%T'
```

## set date

---

<b>Syntax</b>	set date ( <i>date-time</i>   ntp < <i>ntp-server</i> > <source-address <i>source-address</i> >)
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Set the date and time.  user@host> set date ntp 21 Apr 17:22:02 ntpdate[3867]: step time server 172.17.27.46 offset 8.759252 sec
<b>Options</b>	<ul style="list-style-type: none"> <li>• <i>date-time</i>—Specify date and time in one of the following formats: <ul style="list-style-type: none"> <li>• <i>YYYYMMDDHHMM.SS</i></li> <li>• “<i>month DD, YYYY HH:MM(am   pm)</i>”</li> </ul> </li> <li>• <i>ntp</i>—Configure the router to synchronize the current date and time setting with a Network Time Protocol (NTP) server.</li> <li>• <i>ntp-server</i>—(Optional) Specify the IP address of one or more NTP servers.</li> <li>• <i>source-address source-address</i>—(Optional) Specify the source address that is used by the router to contact the remote NTP server.</li> </ul>
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <i>Setting the Date and Time Locally</i></li> </ul>

## show cli

<b>List of Syntax</b>	<a href="#">Syntax on page 324</a> <a href="#">Syntax (QFX Series and OCX Series) on page 324</a>
<b>Syntax</b>	show cli
<b>Syntax (QFX Series and OCX Series)</b>	show cli <authorization> <directory> <history <i>count</i> >
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. Command introduced in Junos OS Release 9.0 for EX Series switches. Command introduced in Junos OS Release 11.1 for the QFX Series. Command introduced in Junos OS Release 14.1X53-D20 for the OCX Series.
<b>Description</b>	Display configured CLI settings.
<b>Options</b>	This command has no options.
<b>Required Privilege Level</b>	view
<b>List of Sample Output</b>	<a href="#">show cli on page 325</a>
<b>Output Fields</b>	<a href="#">Table 16 on page 324</a> lists the output fields for the <b>show cli</b> command. Output fields are listed in the approximate order in which they appear.

**Table 16: show cli Output Fields**

Field Name	Field Description
CLI complete-on-space	Capability to complete a partial command entry when you type a space or a tab: <b>on</b> or <b>off</b> .
CLI idle-timeout	Maximum time that an individual session can be idle before the user is logged out from the router or switch. When this feature is enabled, the number of minutes is displayed. Otherwise, the state is <b>disabled</b> .
CLI restart-on-upgrade	CLI is set to prompt you to restart the router or switch after upgrading the software: <b>on</b> or <b>off</b> .
CLI screen-length	Number of lines of text that the terminal screen displays.
CLI screen-width	Number of characters in a line on the terminal screen.
CLI terminal	Terminal type.
CLI is operating in	Mode: <b>enhanced</b> .
CLI timestamp	Date and time format for the timestamp. If the timestamp is not set, the state is <b>disabled</b> .
CLI working directory	Pathname of the working directory.

## Sample Output

show cli

```
user@host> show cli
CLI complete-on-space set to on
CLI idle-timeout disabled
CLI restart-on-upgrade set to on
CLI screen-length set to 47
CLI screen-width set to 132
CLI terminal is 'vt100'
CLI is operating in enhanced mode
CLI timestamp disabled
CLI working directory is '/var/tmp'
```

## show cli

---

<b>Syntax</b>	show cli
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	<p>Display configured CLI settings.</p> <pre>user@host&gt; show cli CLI complete-on-space set to on CLI idle-timeout disabled CLI restart-on-upgrade set to on CLI screen-length set to 47 CLI screen-width set to 132 CLI terminal is 'vt100' CLI is operating in enhanced mode CLI timestamp disabled CLI working directory is '/var/tmp'</pre>
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">show cli authorization on page 327</a></li><li>• <a href="#">show cli directory on page 328</a></li></ul>

## show cli authorization

**Syntax** show cli authorization

**Release Information** Command introduced before Junos OS Release 7.4.

**Description** Display the permissions for the current user.

```
user@host> show cli authorization
Current user: 'root' login: 'boojum' class '(root)'
Permissions:
Permissions:
 admin -- Can view user accounts
 admin-control-- Can modify user accounts
 clear -- Can clear learned network info
 configure -- Can enter configuration mode
 control -- Can modify any config
 edit -- Can edit full files
 field -- Can use field debug commands
 floppy -- Can read and write the floppy
 interface -- Can view interface configuration
 interface-control-- Can modify interface configuration
 network -- Can access the network
 reset -- Can reset/restart interfaces and daemons
 routing -- Can view routing configuration
 routing-control-- Can modify routing configuration
 shell -- Can start a local shell
 snmp -- Can view SNMP configuration
 snmp-control-- Can modify SNMP configuration
 system -- Can view system configuration
 system-control-- Can modify system configuration
 trace -- Can view trace file settings
 trace-control-- Can modify trace file settings
 view -- Can view current values and statistics
 maintenance -- Can become the super-user
 firewall -- Can view firewall configuration
 firewall-control-- Can modify firewall configuration
 secret -- Can view secret statements
 secret-control-- Can modify secret statements
 rollback -- Can rollback to previous configurations
 security -- Can view security configuration
 security-control-- Can modify security configuration
 access -- Can view access configuration
 access-control-- Can modify access configuration
 view-configuration-- Can view all configuration (not including secrets)
 flow-tap -- Can view flow-tap configuration
 flow-tap-control-- Can modify flow-tap configuration
 idp-profiler-operation-- Can Profiler data
 pgcp-session-mirroring-- Can view pgcp session mirroring configuration
 pgcp-session-mirroring-control-- Can modify pgcp session mirroring configuration
 storage -- Can view fibre channel storage protocol configuration
 storage-control-- Can modify fibre channel storage protocol configuration
 all-control -- Can modify any configuration
```

**Required Privilege Level** view

## show cli directory

---

<b>Syntax</b>	show cli directory
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Display the current working directory.  user@host> <b>show cli directory</b> Current directory: /var/tmp
<b>Required Privilege Level</b>	view



## show cli history

---

<b>Syntax</b>	show cli history < <i>count</i> >
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	<p>Display a list of previous CLI commands.</p> <pre>user@host&gt; show cli history 11:14:14 -- show arp 11:22:10 -- show cli authorization 11:27:12 -- show cli history</pre>
<b>Options</b>	<p>none—Display all previous CLI commands.</p> <p><i>count</i>—(Optional) Maximum number of commands to display.</p>
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Displaying the Junos OS CLI Command and Word History on page 54</a></li></ul>



## CHAPTER 14

# Junos OS CLI Operational Mode Commands

- `configure`
- `file`
- `help`
- `|` (pipe)
- `request`
- `request system commit server pause`
- `request system commit server queue cleanup`
- `request system commit server start`
- `restart`
- `set`
- `show system commit server queue`
- `show system commit server status`

## configure

---

<b>Syntax</b>	<code>configure</code> <code>&lt;batch&gt;</code> <code>&lt;dynamic&gt;</code> <code>&lt;exclusive&gt;</code> <code>&lt;private&gt;</code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. Command introduced in Junos OS Release 9.0 for EX Series switches.
<b>Description</b>	Enter configuration mode. When this command is entered without any optional keywords, everyone can make configuration changes and commit all changes made to the configuration.
<b>Options</b>	<p><b>none</b>—Enter configuration mode.</p> <p><b>batch</b>—(Optional) Work in the batch commit mode where commit operations are executed in batches.</p> <p><b>dynamic</b>—(Optional) Configure routing policies and certain routing policy objects in a dynamic database that is not subject to the same verification required in the standard configuration database. As a result, the time it takes to commit changes to the dynamic database is much shorter than for the standard configuration database. You can then reference these policies and policy objects in routing policies you configure in the standard database.</p> <p><b>exclusive</b>—(Optional) Lock the candidate configuration for as long as you remain in configuration mode, allowing you to make changes without interference from other users. Other users can enter and exit configuration mode, but they cannot change the configuration.</p> <p><b>private</b>—(Optional) Allow multiple users to edit different parts of the configuration at the same time and to commit only their own changes, or to roll back without interfering with one another's changes. You cannot commit changes in configure private mode when another user is in configure exclusive mode.</p>
<b>Additional Information</b>	For more information about the different methods of entering configuration mode and the restrictions that apply, see the <i>Junos OS Administration Library</i> .
<b>Required Privilege Level</b>	configure
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">show configuration on page 295</a></li></ul>
<b>List of Sample Output</b>	<a href="#">configure on page 333</a>
<b>Output Fields</b>	When you enter this command, you are placed in configuration mode and the system prompt changes from <i>hostname&gt;</i> to <i>hostname#</i> .

## Sample Output

### configure

```
user@host> configure
Entering configuration mode
[edit]
user@host#
```

## file

---

<b>Syntax</b>	file <archive   checksum   compare   copy   delete   list   rename   show   source address>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Archive files from the device, copy files to and from the router or switch, calculate the file checksum, compare files, delete a file from the device, list files on the device, rename a file, show file contents, or show the local address to initiate a connection.
<b>Options</b>	<p><b>archive (Optional)</b>—Archive, and optionally compress, one or multiple local system files as a single file, locally or at a remote location.</p> <p><b>checksum (Optional)</b>—Calculate the Message Digest 5 (MD5) checksum of a file.</p> <p><b>compare (Optional)</b>—Compare two local files and describe the differences between them in default, context, or unified output styles.</p> <p><b>copy (Optional)</b>—Copy files from one place to another on the local switch or between the local switch and a remote system.</p> <p><b>delete (Optional)</b>—Delete a file on the local switch.</p> <p><b>list (Optional)</b>—Display a list of files on the local switch.</p> <p><b>rename (Optional)</b>—Rename a file on the local switch.</p> <p><b>show (Optional)</b>—Display the contents of a file.</p> <p><b>source address (Optional)</b>—Specify the source address of the local file.</p>
<b>Required Privilege Level</b>	maintenance
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Viewing Files and Directories on a Device Running Junos OS on page 173</a></li></ul>

## help

---

<b>Syntax</b>	<code>help &lt; (apropos <i>string</i>   reference &lt;<i>statement-name</i>&gt;   syslog &lt;<i>syslog-tag</i>&gt;   tip cli <i>number</i>   topic &lt;<i>word</i>&gt; )&gt;</code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. <b>apropos</b> option added in Junos OS Release 8.0.
<b>Description</b>	Display help about available operational commands, configuration statements, or general information about getting help. Entering the <b>help</b> command without an option provides introductory information about how to use the <b>help</b> and <b>?</b> commands.
<b>Options</b>	<p><b>apropos <i>string</i></b>—(Optional) Display command names and help text that matches the string specified. If the string contains spaces, enclose it in quotation marks (" "). You can also specify a regular expression for the string, using standard UNIX-style regular expression syntax.</p> <p><b>reference &lt;<i>statement-name</i>&gt;</b>—(Optional) Display summary information for a configuration statement. This information is based on summary descriptions that appear in the Junos configuration guides.</p> <p><b>syslog &lt;<i>syslog-tag</i>&gt;</b>—(Optional) Display information about system log messages.</p> <p><b>tip cli <i>number</i></b>—(Optional) Display a tip about using the CLI. Specify the number of the tip you want to view.</p> <p><b>topic &lt;<i>word</i>&gt;</b>—(Optional) Display usage guidelines for a topic or configuration statement. This information is based on subjects that appear in the Junos configuration guides.</p>
<b>Required Privilege Level</b>	None
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li>• <a href="#">Getting Online Help from the Junos OS Command-Line Interface on page 47</a></li> </ul>

## | (pipe)

---

<b>Syntax</b>	(compare   count   display (changed   commit-scripts   detail   inheritance   json   omit   set   translation-scripts <configured-delta   translated-config   translated-delta>   xml)   except <i>pattern</i>   find <i>pattern</i>   hold   last <i>lines</i>   match <i>pattern</i>   no-more   refresh <i>interval</i>   request message (all   <i>account@terminal</i> ) resolve <full-names>   save <i>filename</i>   append <i>filename</i>   tee   trim <i>columns</i> )
<b>Release Information</b>	Command introduced before Junos OS Release 7.4. <b>display commit-scripts</b> option added in Junos OS Release 7.4. <b>tee</b> option added in Junos OS Release 14.1. <b>display json</b> option added in Junos OS Release 14.2. <b>compare   display xml</b> option added in Junos OS Release 15.1. <b>display translation-scripts</b> option added in Junos OS Release 16.1.
<b>Description</b>	Filter the output of an operational mode or a configuration mode command.
<b>Options</b>	<p><b>append <i>filename</i></b>—Append the output to a file.</p> <p><b>compare (<i>filename</i>   rollback <i>n</i> )</b>—Compare configuration changes with another configuration file. In operational mode, use the <b>show configuration</b> command. In configuration mode, use the <b>show</b> command. See <a href="#">“Comparing Configurations and Displaying the Differences in Text” on page 190</a></p> <p><b>compare   display xml</b>—Compare configuration changes with the active configuration and display them in XML format. In operational mode, use the <b>show configuration</b> command. In configuration mode, use the <b>show</b> command. See <a href="#">“Understanding the show   compare   display xml Command Output” on page 128</a>.</p> <p><b>count</b>—Display the number of lines in the output.</p> <p><b>display</b>—Display additional information about the configuration contents.</p> <p><b>changed</b>—Tag changes with <b>junos:changed attribute</b> (XML only).</p> <p><b>commit-scripts</b>—(Configuration mode only) Display all statements that are in a configuration, including statements that were generated by transient changes. For more information, see the <i>Automation Scripting Feature Guide</i>.</p> <p><b>detail</b>—(Configuration mode only) Display configuration data detail.</p> <p><b>inheritance &lt;brief   default   no-comments   groups   terse&gt;</b>—(Configuration mode only) Display inherited configuration data and source group.</p> <p><b>json</b>—Display the output for operational commands and configuration data in JavaScript Object Notation (JSON) format.</p> <p><b>omit</b>—(Configuration mode only) Display configuration statements omitted by the <b>apply-flags omit</b> configuration statement.</p> <p><b>set</b>—Display the configuration as a series of configuration mode commands required to re-create the configuration.</p>



**translation-scripts**—Display the configuration with YANG translation scripts applied.

To view the complete post-inheritance configuration with the translated configuration data from all enabled YANG translation scripts included in the output, append the **| display translation-scripts** filter to the **show configuration** command in operational mode or the **show** command in configuration mode.

You can also append one of several keywords to display different views of the configuration data corresponding to the non-native YANG data models:

- **configured-delta**—In configuration mode, compare the candidate and active configurations, and display configuration changes in the statements or hierarchies corresponding to non-native YANG data models before any translation is applied. The XML output displays the deleted content, followed by the new content in the syntax defined by the YANG data model.
- **translated-config**—In operational or configuration mode, display all non-native configuration data present in the committed or candidate configuration, respectively, after processing by all enabled translation scripts into Junos OS syntax.
- **translated-delta**—In configuration mode, compare the candidate and active configurations, and display configuration changes in the statements or hierarchies corresponding to non-native YANG data models after translation is applied. The XML output displays the deleted content, followed by the new content in Junos OS syntax.

**xml**—(Operational mode only) Display the command output as Junos XML protocol (Extensible Markup Language [XML]) tags.

**except pattern**—Ignore text matching a regular expression when searching the output. If the regular expression contains spaces, operators, or wildcard characters, enclose it in quotation marks.

**find pattern**—Display the output starting at the first occurrence of text matching a regular expression. If the regular expression contains spaces, operators, or wildcard characters, enclose it in quotation marks (" ").

**hold**—Hold text without exiting the **--More--** prompt.

**last lines**—Display the last number of lines you want to view from the end of the configuration. However, when the number of lines requested is less than the number of lines that the screen length setting permits you to display, Junos returns as many lines as permitted by the screen length setting. For more information on using the **last lines** option, see [“Displaying Output Beginning with the Last Entries” on page 196](#).

**match pattern**—Search for text matching a regular expression. If the regular expression contains spaces, operators, or wildcard characters, enclose it in quotation marks.

**no-more**—Display output all at once rather than one screen at a time.

**resolve**—(Operational mode only) Convert IP addresses into Domain Name System (DNS) names. Truncates to fit original size unless **full-names** is specified. To prevent the names from being truncated, use the **full-names** option.

**refresh interval**—Refresh the display of the command according to the interval specified. The screen gets refreshed periodically to show you the current output of the command until you quit the command. The default refresh interval is one second. However, you can also explicitly specify a value from 1 through 604800 for the refresh interval.

**request message (all | account@terminal )**—Display command output on the terminal of a specific user logged in to your router, or on the terminals of all users logged in to your router.

**save filename**—Save the output to a file or URL. For information about specifying the filename, see [“Specifying Filenames and URLs” on page 176](#).

**tee**—Allows you to both display the command output on screen and write it to a file. Unlike the UNIX **tee** command, if the file cannot be opened, just an error message is displayed.

**trim columns**—Trim specified number of columns from the start line. Only positive values are accepted. An error message appears if a negative value is given.

**Required Privilege Level**

view

**Related Documentation**

- [Displaying the Current Junos OS Configuration on page 97](#).
- [Using the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 187](#)
- [Using Regular Expressions with the Pipe \( | \) Symbol to Filter Junos OS Command Output on page 188](#)
- [Pipe \( | \) Filter Functions in the Junos OS Command-Line Interface on page 190](#)

## request

**Syntax** request <chassis | ipsec switch | message | mpls | routing-engine | security | services | system | flow-collector | support information>

**Release Information** Command introduced before Junos OS Release 7.4.

**Description** Stop or reboot router components, switch between primary and backup components, display messages, and display system information.



**CAUTION:** Halt the backup Routing Engine before you remove it or shut off the power to the router; otherwise, you might need to reinstall the Junos OS.



**NOTE:** If your router contains two Routing Engines and you want to shut the power off to the router or remove a Routing Engine, you must first halt the backup Routing Engine (if it has been upgraded) and then the master Routing Engine. To halt a Routing Engine, enter the `request system halt` command. You can also halt both Routing Engines at the same time by issuing the `request system halt both-routing-engines` command.

If you want to reboot a router that has two Routing Engines, reboot the backup Routing Engine (if you have upgraded it) and then the master Routing Engine.



**NOTE:** If you reboot the TX Matrix router, all the T640 master Routing Engines connected to the TX Matrix router reboot. If you halt both Routing Engines on a TX Matrix router, all the T640 Routing Engines connected to the TX Matrix router are also halted. Likewise, if you reboot the TX Matrix Plus router, all the T1600 or T4000 master Routing Engines connected to the TX Matrix Plus router reboot. If you halt both Routing Engines on a TX Matrix Plus router, all the T1600 or T4000 Routing Engines connected to the TX Matrix Plus router are also halted.



**NOTE:** If you insert a Flexible PIC Concentrator (FPC) into your router, you may need to issue the `request chassis fpc` command (or press the online button) to bring the FPC online. This applies to FPCs in M20, M40, M40e, M160, M320, and T Series routers. For command usage, see the `request chassis fpc` command description in the [CLI Explorer](#).

**Additional Information** Most **request** commands are described in the *Junos System Basics and Services Command Reference*. The following **request** commands are described in the *Junos Interfaces Command Reference*: **request ipsec switch** and **request services**.

**Required Privilege Level** maintenance

**Related Documentation**

- [Overview of Junos OS CLI Operational Mode Commands on page 163](#)

---

## request system commit server pause

---

**Syntax**     `request system commit server pause`

**Release Information**     Command introduced in Junos OS Release 12.1.

**Description**     Pause the commit server.



**NOTE:** If you issue this command when a commit job is in process, the batch commit server pauses only after the current commit job is completed.

**Options**     This command has no options.

**Required Privilege Level**     view

**Related Documentation**     • [Example: Configuring Batch Commit Server Properties on page 117](#)

### Sample Output

When you enter the `request system commit server pause` command, you are provided feedback on the status of your request.

#### request system commit server pause

```
user@host> request system commit server pause
```

```
Successfully paused the commit server.
```

## request system commit server queue cleanup

---

<b>Syntax</b>	<b>request system commit server queue cleanup</b> <b>&lt;id <i>commit-id</i>&gt;</b> <b>&lt;job-status (error  pending  success)&gt;</b>
<b>Release Information</b>	Command introduced in Junos OS Release 12.1.
<b>Description</b>	Clean up the batch commit queue.
<b>Options</b>	<b>id <i>commit-id</i></b> —(Optional) Clean up batch commit operation status messages for a specific commit ID.  <b>job-status</b> —(Optional) Clean up batch commit operation status messages for the following: <ul style="list-style-type: none"><li>• <b>error</b>—Clean up status messages for batch commit operations that have errors.</li><li>• <b>pending</b>—Clean up status messages for batch commit operations that are pending.</li><li>• <b>success</b>—Clean up status messages for batch commit operations that are successful.</li></ul>
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Example: Configuring Batch Commit Server Properties on page 117</a></li></ul>
<b>List of Sample Output</b>	<a href="#">request system commit server queue cleanup id on page 342</a> <a href="#">request system commit server queue cleanup job-status success on page 342</a>

### Sample Output

When you enter the **request system commit server queue cleanup** command, you are provided feedback on the status of your request.

#### **request system commit server queue cleanup id**

```
user@host> request system commit server queue cleanup id 1008

Successfully cleaned up jobs.
```

#### **request system commit server queue cleanup job-status success**

```
user@host> request system commit server queue cleanup job-status success

Successfully cleaned up jobs.
```

## request system commit server start

---

<b>Syntax</b>	<code>request system commit server start</code>
<b>Release Information</b>	Command introduced in Junos OS Release 12.1.
<b>Description</b>	Start the commit server.
<b>Options</b>	This command has no options.
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">Example: Configuring Batch Commit Server Properties on page 117</a></li></ul>

### Sample Output

When you enter the `request system commit server start` command, you are provided feedback on the status of your request.

#### request system commit server start

```
user@host> request system commit server start

Successfully started the commit server.
```

## restart

### List of Syntax Syntax on page 344

Syntax (ACX Series Routers) on page 344

Syntax (EX Series Switches) on page 344

Syntax (MX Series Routers) on page 345

Syntax (QFX Series) on page 345

Syntax (Routing Matrix) on page 345

Syntax (TX Matrix Routers) on page 345

Syntax (TX Matrix Plus Routers) on page 346

Syntax (MX Series Routers) on page 346

Syntax (QFX Series) on page 346

### Syntax restart

```
<adaptive-services | ancpd-service | application-identification | audit-process |
auto-configuration | captive-portal-content-delivery | ce-l2tp-service | chassis-control |
class-of-service | clksyncd-service | database-replication | datapath-trace-service
| dhcp-service | diameter-service | disk-monitoring | dynamic-flow-capture |
ecc-error-logging | ethernet-connectivity-fault-management
| ethernet-link-fault-management | event-processing | firewall
| general-authentication-service | gracefully | iccp-service | idp-policy | immediately
| interface-control | ipsec-key-management | kernel-replication | l2-learning | l2cpd-service
| l2tp-service | l2tp-universal-edge | lacp | license-service | link-management
| local-policy-decision-function | mac-validation | mib-process | mounstd-service
| mpls-traceroute | mspd | multicast-snooping | named-service | nfsd-service |
packet-triggered-subscribers | peer-selection-service | pgm | pic-services-logging | pki-service
| ppp | ppp-service | pppoe | protected-system-domain-service |
redundancy-interface-process | remote-operations | root-system-domain-service | routing
<logical-system logical-system-name> | sampling | sbc-configuration-process | sdk-service
| service-deployment | services | snmp | soft | static-subscribers | statistics-service |
subscriber-management | subscriber-management-helper | tunnel-oamd | usb-control |
vrrp | web-management>
<gracefully | immediately | soft>
```

### Syntax (ACX Series Routers)

#### restart

```
<adaptive-services | audit-process | auto-configuration | autoinstallation | chassis-control |
class-of-service | clksyncd-service | database-replication | dhcp-service | diameter-service
| disk-monitoring | dynamic-flow-capture | ethernet-connectivity-fault-management
| ethernet-link-fault-management | event-processing | firewall
| general-authentication-service | gracefully | immediately | interface-control |
ipsec-key-management | l2-learning | lacp | link-management | mib-process | mounstd-service
| mpls-traceroute | mspd | named-service | nfsd-service | pgm | pki-service | ppp | pppoe |
redundancy-interface-process | remote-operations | routing | sampling | sdk-service
| secure-neighbor-discovery | service-deployment | services | snmp | soft | statistics-service |
subscriber-management | subscriber-management-helper | tunnel-oamd | vrrp>
```

### Syntax (EX Series Switches)

#### restart

```
<autoinstallation | chassis-control | class-of-service | database-replication | dhcp |
dhcp-service | diameter-service | dot1x-protocol | ethernet-link-fault-management |
ethernet-switching | event-processing | firewall | general-authentication-service |
interface-control | kernel-replication | l2-learning | lacp | license-service | link-management
| lldpd-service | mib-process | mounstd-service | multicast-snooping | pgm |
```



	redundancy-interface-process   remote-operations   routing   secure-neighbor-discovery   service-deployment   sflow-service   snmp   vrrp   web-management>
<b>Syntax (MX Series Routers)</b>	restart <adaptive-services   ancpd-service   application-identification   audit-process   auto-configuration   captive-portal-content-delivery   ce-l2tp-service   chassis-control   class-of-service   clksyncd-service   database-replication   datapath-trace-service   dhcp-service   diameter-service   disk-monitoring   dynamic-flow-capture   ecc-error-logging   ethernet-connectivity-fault-management   ethernet-link-fault-management   event-processing   firewall   general-authentication-service   gracefully   iccp-service   idp-policy   immediately   interface-control   ipsec-key-management   kernel-replication   l2-learning   l2cpd-service   l2tp-service   l2tp-universal-edge   lacp   license-service   link-management   local-policy-decision-function   mac-validation   mib-process   mountd-service   mpls-traceroute   mspd   multicast-snooping   named-service   nfsd-service   packet-triggered-subscribers   peer-selection-service   pgm   pic-services-logging   pki-service   ppp   ppp-service   pppoe   protected-system-domain-service   redundancy-interface-process   remote-operations   root-system-domain-service   routing   routing <logical-system <i>logical-system-name</i> >   sampling   sbc-configuration-process   sdk-service   service-deployment   services   snmp   soft   static-subscribers   statistics-service   subscriber-management   subscriber-management-helper   tunnel-oamd   usb-control   vrrp   web-management> <all-members> <gracefully   immediately   soft> <local> <member <i>member-id</i> >
<b>Syntax (QFX Series)</b>	restart <adaptive-services   audit-process   chassis-control   class-of-service   dialer-services   diameter-service   dlsw   ethernet-connectivity   event-processing   fibre-channel   firewall   general-authentication-service   igmp-host-services   interface-control   ipsec-key-management   isdn-signaling   l2ald   l2-learning   l2tp-service   mib-process   named-service   network-access-service   nstrace-process   pgm   ppp   pppoe   redundancy-interface-process   remote-operations   <i>logical-system-name</i> >   routing   sampling   secure-neighbor-discovery   service-deployment   snmp   usb-control   web-management> <gracefully   immediately   soft>
<b>Syntax (Routing Matrix)</b>	restart <adaptive-services   audit-process   chassis-control   class-of-service   disk-monitoring   dynamic-flow-capture   ecc-error-logging   event-processing   firewall   interface-control   ipsec-key-management   kernel-replication   l2-learning   l2tp-service   lacp   link-management   mib-process   pgm   pic-services-logging   ppp   pppoe   redundancy-interface-process   remote-operations   routing <logical-system <i>logical-system-name</i> >   sampling   service-deployment   snmp> <all   all-lcc   lcc <i>number</i> > <gracefully   immediately   soft>
<b>Syntax (TX Matrix Routers)</b>	restart <adaptive-services   audit-process   chassis-control   class-of-service   dhcp-service   diameter-service   disk-monitoring   dynamic-flow-capture   ecc-error-logging   event-processing   firewall   interface-control   ipsec-key-management   kernel-replication   l2-learning   l2tp-service   lacp   link-management   mib-process   pgm   pic-services-logging   ppp   pppoe   redundancy-interface-process   remote-operations   routing <logical-system <i>logical-system-name</i> >   sampling   service-deployment   snmp   statistics-service>

	<p>&lt;all-chassis   all-lcc   lcc <i>number</i>   scc&gt;</p> <p>&lt;gracefully   immediately   soft&gt;</p>
<b>Syntax (TX Matrix Plus Routers)</b>	<p>restart</p> <p>&lt;adaptive-services   audit-process   chassis-control   class-of-service   dhcp-service   diameter-service   disk-monitoring   dynamic-flow-capture   ecc-error-logging   event-processing   firewall   interface-control   ipsec-key-management   kernel-replication   l2-learning   l2tp-service   lacp   link-management   mib-process   pgm   pic-services-logging   ppp   pppoe   redundancy-interface-process   remote-operations   routing &lt;logical-system <i>logical-system-name</i>&gt;   sampling   service-deployment   snmp   statistics-service&gt;</p> <p>&lt;all-chassis   all-lcc   all-sfc   lcc <i>number</i>   sfc <i>number</i>&gt;</p> <p>&lt;gracefully   immediately   soft&gt;</p>
<b>Syntax (MX Series Routers)</b>	<p>restart</p> <p>&lt;adaptive-services   ancpd-service   application-identification   audit-process   auto-configuration   captive-portal-content-delivery   ce-l2tp-service   chassis-control   class-of-service   clksyncd-service   database-replication   datapath-trace-service   dhcp-service   diameter-service   disk-monitoring   dynamic-flow-capture   ecc-error-logging   ethernet-connectivity-fault-management   ethernet-link-fault-management   event-processing   firewall   general-authentication-service   gracefully   iccp-service   idp-policy   immediately   interface-control   ipsec-key-management   kernel-replication   l2-learning   l2cpd-service   l2tp-service   l2tp-universal-edge   lacp   license-service   link-management   local-policy-decision-function   mac-validation   mib-process   mobile-ip   mounstd-service   mpls-traceroute   mspd   multicast-snooping   named-service   nfsd-service   packet-triggered-subscribers   peer-selection-service   pgcp-service   pgm   pic-services-logging   pki-service   ppp   ppp-service   pppoe   protected-system-domain-service   redundancy-interface-process   remote-operations   root-system-domain-service   routing   routing &lt;logical-system <i>logical-system-name</i>&gt;   sampling   sbc-configuration-process   sdk-service   service-deployment   services   services pgcp gateway <i>gateway-name</i>   snmp   soft   static-subscribers   statistics-service   subscriber-management   subscriber-management-helper   tunnel-oamd   usb-control   vrrp   web-management&gt;</p> <p>&lt;all-members&gt;</p> <p>&lt;gracefully   immediately   soft&gt;</p> <p>&lt;local&gt;</p> <p>&lt;member <i>member-id</i>&gt;</p>
<b>Syntax (QFX Series)</b>	<p>restart</p> <p>&lt;adaptive-services   audit-process   chassis-control   class-of-service   dialer-services   diameter-service   dlsu   ethernet-connectivity   event-processing   fibre-channel   firewall   general-authentication-service   igmp-host-services   interface-control   ipsec-key-management   isdn-signaling   l2ald   l2-learning   l2tp-service   mib-process   named-service   network-access-service   nstrace-process   pgm   ppp   pppoe   redundancy-interface-process   remote-operations   <i>logical-system-name</i>&gt;   routing   sampling   secure-neighbor-discovery   service-deployment   snmp   usb-control   web-management&gt;</p> <p>&lt;gracefully   immediately   soft&gt;</p>
<b>Release Information</b>	<p>Command introduced before Junos OS Release 7.4.</p> <p>Command introduced in Junos OS Release 9.0 for EX Series switches.</p> <p>Command introduced in Junos OS Release 11.1 for the QFX Series.</p> <p>Command introduced in Junos OS Release 12.2 for ACX Series routers.</p>

Command introduced in Junos OS Release 14.1X53-D20 for the OCX Series.

Options added:

- **dynamic-flow-capture** in Junos OS Release 7.4.
- **dls** in Junos OS Release 7.5.
- **event-processing** in Junos OS Release 7.5.
- **ppp** in Junos OS Release 7.5.
- **l2ald** in Junos OS Release 8.0.
- **link-management** in Release 8.0.
- **pgcp-service** in Junos OS Release 8.4.
- **sbc-configuration-process** in Junos OS Release 9.5.
- **services pgcp gateway** in Junos OS Release 9.6.
- **sfc** and **all-sfc** for the TX Matrix Router in Junos OS Release 9.6.

**Description** Restart a Junos OS process.



**CAUTION:** Never restart a software process unless instructed to do so by a customer support engineer. A restart might cause the router or switch to drop calls and interrupt transmission, resulting in possible loss of data.

**Options** **none**—Same as **gracefully**.

**adaptive-services**—(Optional) Restart the configuration management process that manages the configuration for stateful firewall, Network Address Translation (NAT), intrusion detection services (IDS), and IP Security (IPsec) services on the Adaptive Services PIC.

**all-chassis**—(TX Matrix and TX Matrix Plus routers only) (Optional) Restart the software process on all chassis.

**all-lcc**—(TX Matrix and TX Matrix Plus routers only) (Optional) For a TX Matrix router, restart the software process on all T640 routers connected to the TX Matrix router. For a TX Matrix Plus router, restart the software process on all T1600 routers connected to the TX Matrix Plus router.

**all-members**—(MX Series routers only) (Optional) Restart the software process for all members of the Virtual Chassis configuration.

**all-sfc**—(TX Matrix Plus routers only) (Optional) For a TX Matrix Plus router, restart the software processes for the TX Matrix Plus router (or switch-fabric chassis).

**ancpd-service**—(Optional) Restart the Access Node Control Protocol (ANCP) process, which works with a special Internet Group Management Protocol (IGMP) session to collect outgoing interface mapping events in a scalable manner.

**application-identification**—(Optional) Restart the process that identifies an application using intrusion detection and prevention (IDP) to allow or deny traffic based on applications running on standard or nonstandard ports.

**audit-process**—(Optional) Restart the RADIUS accounting process that gathers statistical data that can be used for general network monitoring, analyzing, and tracking usage patterns, for billing a user based on the amount of time or type of services accessed.

**auto-configuration**—(Optional) Restart the Interface Auto-Configuration process.

**autoinstallation**—(EX Series switches only) (Optional) Restart the autoinstallation process.

**captive-portal-content-delivery**—(Optional) Restart the HTTP redirect service by specifying the location to which a subscriber's initial Web browser session is redirected, enabling initial provisioning and service selection for the subscriber.

**ce-l2tp-service**—(M10, M10i, M7i, and MX Series routers only) (Optional) Restart the Universal Edge Layer 2 Tunneling Protocol (L2TP) process, which establishes L2TP tunnels and Point-to-Point Protocol (PPP) sessions through L2TP tunnels.

**chassis-control**—(Optional) Restart the chassis management process.

**class-of-service**—(Optional) Restart the class-of-service (CoS) process, which controls the router's or switch's CoS configuration.

**clksyncd-service**—(Optional) Restart the external clock synchronization process, which uses synchronous Ethernet (SyncE).

**database-replication**—(EX Series switches and MX Series routers only) (Optional) Restart the database replication process.

**datapath-trace-service**—(Optional) Restart the packet path tracing process.

**dhcp**—(EX Series switches only) (Optional) Restart the software process for a Dynamic Host Configuration Protocol (DHCP) server. A DHCP server allocates network IP addresses and delivers configuration settings to client hosts without user intervention.

**dhcp-service**—(Optional) Restart the Dynamic Host Configuration Protocol process.

**dialer-services**—(EX Series switches only) (Optional) Restart the ISDN dial-out process.

**diameter-service**—(Optional) Restart the diameter process.

**disk-monitoring**—(Optional) Restart disk monitoring, which checks the health of the hard disk drive on the Routing Engine.

**dls**—(QFX Series only) (Optional) Restart the data link switching (DLSw) service.

**dot1x-protocol**—(EX Series switches only) (Optional) Restart the port-based network access control process.

**dynamic-flow-capture**—(Optional) Restart the dynamic flow capture (DFC) process, which controls DFC configurations on Monitoring Services III PICs.

**ecc-error-logging**—(Optional) Restart the error checking and correction (ECC) process, which logs ECC parity errors in memory on the Routing Engine.

**ethernet-connectivity-fault-management**—(Optional) Restart the process that provides IEEE 802.1ag Operation, Administration, and Management (OAM) connectivity fault management (CFM) database information for CFM maintenance association end points (MEPs) in a CFM session.

**ethernet-link-fault-management**—(EX Series switches and MX Series routers only)  
(Optional) Restart the process that provides the OAM link fault management (LFM) information for Ethernet interfaces.

**ethernet-switching**—(EX Series switches only) (Optional) Restart the Ethernet switching process.

**event-processing**—(Optional) Restart the event process (eventd).

**fibre-channel**—(QFX Series only) (Optional) Restart the Fibre Channel process.

**firewall**—(Optional) Restart the firewall management process, which manages the firewall configuration and enables accepting or rejecting packets that are transiting an interface on a router or switch.

**general-authentication-service**—(EX Series switches and MX Series routers only)  
(Optional) Restart the general authentication process.

**gracefully**—(Optional) Restart the software process.

**iccp-service**—(Optional) Restart the Inter-Chassis Communication Protocol (ICCP) process.

**idp-policy**—(Optional) Restart the intrusion detection and prevention (IDP) protocol process.

**immediately**—(Optional) Immediately restart the software process.

**interface-control**—(Optional) Restart the interface process, which controls the router's or switch's physical interface devices and logical interfaces.

**ipsec-key-management**—(Optional) Restart the IPsec key management process.

**isdn-signaling**—(QFX Series only) (Optional) Restart the ISDN signaling process, which initiates ISDN connections.

**kernel-replication**—(Optional) Restart the kernel replication process, which replicates the state of the backup Routing Engine when graceful Routing Engine switchover (GRES) is configured.

**l2-learning**—(Optional) Restart the Layer 2 address flooding and learning process.

**l2cpd-service**—(Optional) Restart the Layer 2 Control Protocol process, which enables features such as Layer 2 protocol tunneling and nonstop bridging.

**l2tp-service**— (M10, M10i, M7i, and MX Series routers only) (Optional) Restart the Layer 2 Tunneling Protocol (L2TP) process, which sets up client services for establishing Point-to-Point Protocol (PPP) tunnels across a network and negotiating Multilink PPP if it is implemented.

**l2tp-universal-edge**— (MX Series routers only) (Optional) Restart the L2TP process, which establishes L2TP tunnels and PPP sessions through L2TP tunnels.

**lACP**— (Optional) Restart the Link Aggregation Control Protocol (LACP) process. LACP provides a standardized means for exchanging information between partner systems on a link to allow their link aggregation control instances to reach agreement on the identity of the LAG to which the link belongs, and then to move the link to that LAG, and to enable the transmission and reception processes for the link to function in an orderly manner.

**lcc number**— (TX Matrix and TX Matrix Plus routers only) (Optional) For a TX Matrix router, restart the software process for a specific T640 router that is connected to the TX Matrix router. For a TX Matrix Plus router, restart the software process for a specific router that is connected to the TX Matrix Plus router.

Replace *number* with the following values depending on the LCC configuration:

- 0 through 3, when T640 routers are connected to a TX Matrix router in a routing matrix.
- 0 through 3, when T1600 routers are connected to a TX Matrix Plus router in a routing matrix.
- 0 through 7, when T1600 routers are connected to a TX Matrix Plus router with 3D SIBs in a routing matrix.
- 0, 2, 4, or 6, when T4000 routers are connected to a TX Matrix Plus router with 3D SIBs in a routing matrix.

**license-service**— (EX Series switches only) (Optional) Restart the feature license management process.

**link-management**— (TX Matrix and TX Matrix Plus routers and EX Series switches only) (Optional) Restart the Link Management Protocol (LMP) process, which establishes and maintains LMP control channels.

**lldpd-service**— (EX Series switches only) (Optional) Restart the Link Layer Discovery Protocol (LLDP) process.

**local**— (MX Series routers only) (Optional) Restart the software process for the local Virtual Chassis member.

**local-policy-decision-function**— (Optional) Restart the process for the Local Policy Decision Function, which regulates collection of statistics related to applications and application groups and tracking of information about dynamic subscribers and static interfaces.

**mac-validation**— (Optional) Restart the Media Access Control (MAC) validation process, which configures MAC address validation for subscriber interfaces created on demux interfaces in dynamic profiles on MX Series routers.

**member *member-id***— (MX Series routers only) (Optional) Restart the software process for a specific member of the Virtual Chassis configuration. Replace ***member-id*** with a value of 0 or 1.

**mib-process**— (Optional) Restart the Management Information Base (MIB) version II process, which provides the router's MIB II agent.

**mobile-ip**— (Optional) Restart the Mobile IP process, which configures Junos OS Mobile IP features.

**moundd-service**— (EX Series switches and MX Series routers only) (Optional) Restart the service for NFS mount requests.

**mpls-traceroute**— (Optional) Restart the MPLS Periodic Traceroute process.

**mspd**— (Optional) Restart the Multiservice process.

**multicast-snooping**— (EX Series switches and MX Series routers only) (Optional) Restart the multicast snooping process, which makes Layer 2 devices, such as VLAN switches, aware of Layer 3 information, such as the media access control (MAC) addresses of members of a multicast group.

**named-service**— (Optional) Restart the DNS Server process, which is used by a router or a switch to resolve hostnames into addresses.

**network-access-service**— (QFX Series only) (Optional) Restart the network access process, which provides the router's Challenge Handshake Authentication Protocol (CHAP) authentication service.

**nfsd-service**— (Optional) Restart the Remote NFS Server process, which provides remote file access for applications that need NFS-based transport.

**packet-triggered-subscribers**— (Optional) Restart the packet-triggered subscribers and policy control (PTSP) process, which allows the application of policies to dynamic subscribers that are controlled by a subscriber termination device.

**peer-selection-service**— (Optional) Restart the Peer Selection Service process.

**pgcp-service**— (Optional) Restart the pgcpd service process running on the Routing Engine. This option does not restart pgcpd processes running on mobile station PICs. To restart pgcpd processes running on mobile station PICs, use the **services pgcp gateway** option.

**pgm**— (Optional) Restart the process that implements the Pragmatic General Multicast (PGM) protocol for assisting in the reliable delivery of multicast packets.

**pic-services-logging**— (Optional) Restart the logging process for some PICs. With this process, also known as fsad (the file system access daemon), PICs send special logging information to the Routing Engine for archiving on the hard disk.

**pki-service**—(Optional) Restart the PKI Service process.

**ppp**—(Optional) Restart the Point-to-Point Protocol (PPP) process, which is the encapsulation protocol process for transporting IP traffic across point-to-point links.

**ppp-service**—(Optional) Restart the Universal edge PPP process, which is the encapsulation protocol process for transporting IP traffic across universal edge routers.

**pppoe**—(Optional) Restart the Point-to-Point Protocol over Ethernet (PPPoE) process, which combines PPP that typically runs over broadband connections with the Ethernet link-layer protocol that allows users to connect to a network of hosts over a bridge or access concentrator.

**protected-system-domain-service**—(Optional) Restart the Protected System Domain (PSD) process.

**redundancy-interface-process**—(Optional) Restart the ASP redundancy process.

**remote-operations**—(Optional) Restart the remote operations process, which provides the ping and traceroute MIBs.

**root-system-domain-service**—(Optional) Restart the Root System Domain (RSD) service.

**routing**—(ACX Series routers, QFX Series, EX Series switches, and MX Series routers only) (Optional) Restart the routing protocol process.

**routing <logical-system *logical-system-name*>**—(Optional) Restart the routing protocol process, which controls the routing protocols that run on the router or switch and maintains the routing tables. Optionally, restart the routing protocol process for the specified logical system only.

**sampling**—(Optional) Restart the sampling process, which performs packet sampling based on particular input interfaces and various fields in the packet header.

**sbc-configuration-process**—(Optional) Restart the session border controller (SBC) process of the border signaling gateway (BSG).

**scc**—(TX Matrix routers only) (Optional) Restart the software process on the TX Matrix router (or switch-card chassis).

**sdk-service**—(Optional) Restart the SDK Service process, which runs on the Routing Engine and is responsible for communications between the SDK application and Junos OS. Although the SDK Service process is present on the router, it is turned off by default.

**secure-neighbor-discovery**—(QFX Series, EX Series switches, and MX Series routers only) (Optional) Restart the secure Neighbor Discovery Protocol (NDP) process, which provides support for protecting NDP messages.

**sfc *number***—(TX Matrix Plus routers only) (Optional) Restart the software process on the TX Matrix Plus router (or switch-fabric chassis). Replace ***number*** with **0**.



**service-deployment**—(Optional) Restart the service deployment process, which enables Junos OS to work with the Session and Resource Control (SRC) software.

**services**—(Optional) Restart a service.

**services pgcp gateway gateway-name**—(Optional) Restart the pgcpd process for a specific border gateway function (BGF) running on an MS-PIC. This option does not restart the pgcpd process running on the Routing Engine. To restart the pgcpd process on the Routing Engine, use the **pgcp-service** option.

**sflow-service**—(EX Series switches only) (Optional) Restart the flow sampling (sFlow technology) process.

**snmp**—(Optional) Restart the SNMP process, which enables the monitoring of network devices from a central location and provides the router's or switch's SNMP master agent.

**soft**—(Optional) Reread and reactivate the configuration without completely restarting the software processes. For example, BGP peers stay up and the routing table stays constant. Omitting this option results in a graceful restart of the software process.

**static-subscribers**—(Optional) Restart the static subscribers process, which associates subscribers with statically configured interfaces and provides dynamic service activation and activation for these subscribers.

**statistics-service**—(Optional) Restart the process that manages the Packet Forwarding Engine statistics.

**subscriber-management**—(Optional) Restart the Subscriber Management process.

**subscriber-management-helper**—(Optional) Restart the Subscriber Management Helper process.

**tunnel-oamd**—(Optional) Restart the Tunnel OAM process, which enables the Operations, Administration, and Maintenance of Layer 2 tunneled networks. Layer 2 protocol tunneling (L2PT) allows service providers to send Layer 2 protocol data units (PDUs) across the provider's cloud and deliver them to Juniper Networks EX Series Ethernet Switches that are not part of the local broadcast domain.

**usb-control**—(MX Series routers) (Optional) Restart the USB control process.

**vrrp**—(ACX Series routers, EX Series switches, and MX Series routers only) (Optional) Restart the Virtual Router Redundancy Protocol (VRRP) process, which enables hosts on a LAN to make use of redundant routing platforms on that LAN without requiring more than the static configuration of a single default route on the hosts.

**web-management**—(QFX Series, EX Series switches, and MX Series routers only) (Optional) Restart the Web management process.

**Required Privilege Level**    reset

**Related Documentation**

- [Overview of Junos OS CLI Operational Mode Commands on page 163](#)

**List of Sample Output**   [restart interfaces on page 354](#)

**Output Fields**   When you enter this command, you are provided feedback on the status of your request.

## Sample Output

### restart interfaces

```
user@host> restart interfaces
interfaces process terminated
interfaces process restarted
```


---

## set

---

<b>Syntax</b>	<code>set &lt;<i>statement-path</i>&gt; <i>identifier</i></code>
<b>Release Information</b>	Command introduced before Junos OS Release 7.4.
<b>Description</b>	Create a statement hierarchy and set identifier values. This is similar to <b>edit</b> except that your current level in the hierarchy does not change.
<b>Options</b>	<p><i>identifier</i>—Name of the statement or identifier to set.</p> <p><i>statement-path</i>—(Optional) Path to an existing statement hierarchy level. If that hierarchy level does not exist, it is created.</p>
<b>Required Privilege Level</b>	configure—To enter configuration mode, but other required privilege levels depend on where the statement is located in the configuration hierarchy.
<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">edit on page 274</a></li><li>• <a href="#">Displaying the Current Junos OS Configuration on page 97</a></li></ul>

## show system commit server queue

<b>Syntax</b>	<b>show system commit server queue</b> <code>&lt;id commit-id&gt;</code> <code>&lt;job-status (all  error  pending  success)&gt;</code> <code>&lt;patch (none   id commit-id)   (job-status (all   error   pending   success))&gt;</code>
<b>Release Information</b>	Command introduced in Junos OS Release 12.1.
<b>Description</b>	Display the status of commit server queue transactions.
	<div>  <p><b>NOTE:</b> Only 50 successful commit jobs are stored in the database and displayed in the output. When the fifty-first job is committed, the first job is deleted from the database and is no longer displayed in the output.</p> </div>
<b>Options</b>	<p><b>id commit-id</b>—(Optional) Display the batch commit operation status messages for a specific commit ID.</p> <p><b>job-status</b>—(Optional) Display batch commit operation status messages for the following batch commit statuses:</p> <ul style="list-style-type: none"> <li><b>all</b>—Status messages for all batch commit operations.</li> <li><b>error</b>—Status messages for batch commit operations that have errors.</li> <li><b>pending</b>—Status messages for batch commit operations that are pending.</li> <li><b>success</b>—Status messages for batch commit operations that are successful.</li> </ul> <p><b>patch (none   id commit-id)   job-status (all   error   pending   success)</b>—(Optional) Display the patch file containing the configuration changes for all batch commit operations, a specific batch commit ID, or a specific job status.</p>
<b>Required Privilege Level</b>	view
<b>Related Documentation</b>	<ul style="list-style-type: none"> <li><a href="#">Example: Configuring Batch Commit Server Properties on page 117</a></li> </ul>
<b>List of Sample Output</b>	<a href="#">show system commit server queue on page 356</a> <a href="#">show system commit server queue job-status success on page 357</a> <a href="#">show system commit server queue patch on page 357</a>

## Sample Output

### show system commit server queue

```
user@host> show system commit server queue
```

```
Pending commits:
none
```

## Completed commits:

Id: 1000  
 Last Modified: Tue Nov 1 22:46:43 2011  
 Status: Successfully committed 1000

Id: 1002  
 Last Modified: Tue Nov 1 22:50:35 2011  
 Status: Successfully committed 1002

Id: 1004  
 Last Modified: Tue Nov 1 22:51:48 2011  
 Status: Successfully committed 1004

Id: 1007  
 Last Modified: Wed Nov 2 01:08:04 2011  
 Status: Successfully committed 1007

Id: 1009  
 Last Modified: Wed Nov 2 01:16:45 2011  
 Status: Successfully committed 1009

Id: 1010  
 Last Modified: Wed Nov 2 01:19:25 2011  
 Status: Successfully committed 1010

Id: 1011  
 Last Modified: Wed Nov 2 01:28:16 2011  
 Status: Successfully committed 1011

## Error commits:

Id: 1008  
 Last Modified: Wed Nov 2 01:08:18 2011  
 Status: Error while committing 1008

**show system commit server queue job-status success**

```
user@host> show system commit server queue job-status success
```

## Completed commits:

Id: 1000  
 Last Modified: Tue Nov 1 22:46:43 2011  
 Status: Successfully committed 1000

Id: 1001  
 Last Modified: Tue Nov 1 22:47:02 2011  
 Status: Successfully committed 1001

**show system commit server queue patch**

```
user@host> show system commit server queue patch
```

## Pending commits:

none

## Completed commits:

Id: 1000  
 Last Modified: Tue Nov 1 22:46:43 2011  
 Status: Successfully committed 1000

## Patch:

[edit system commit]

```
+ server {
+ days-to-keep-error-logs 4294967295;
+ traceoptions {
+ file commitd_nov;
+ flag all;
+ }
+ }
Id: 1002
Last Modified: Tue Nov 1 22:50:35 2011
Status: Successfully committed 1002
```

Patch:

```
[edit system commit server]
- days-to-keep-error-logs 4294967295;
 Id: 1004
 Last Modified: Tue Nov 1 22:51:48 2011
 Status: Successfully committed 1004
```

Patch:

```
[edit system commit server]
+ days-to-keep-error-logs 4294967295;
 Id: 1007
 Last Modified: Wed Nov 2 01:08:04 2011
 Status: Successfully committed 1007
```

Patch:

```
[edit system commit server]
- days-to-keep-error-logs 4294967295;
+ days-to-keep-error-logs 2;
 Id: 1009
 Last Modified: Wed Nov 2 01:16:45 2011
 Status: Successfully committed 1009
```

Patch:

```
[edit]
+ snmp {
+ community abc;
+ }
 Id: 1010
 Last Modified: Wed Nov 2 01:19:25 2011
 Status: Successfully committed 1010
```

Patch:

```
[edit system syslog]
 file test { ... }
+ file j {
+ any any;
+ }
 Id: 1011
 Last Modified: Wed Nov 2 01:28:16 2011
 Status: Successfully committed 1011
```

Error commits:

```
Id: 1008
Last Modified: Wed Nov 2 01:08:18 2011
Status: Error while committing 1008
```


Patch:

```
[edit system]
+ radius-server {
```

```
+ 10.1.1.1 port 222;
+ }
```

## show system commit server status

---

Syntax	show system commit server status
Release Information	Command introduced in Junos OS Release 12.1.
Description	Display commit server status.
<div> <b>NOTE:</b> By default, the status of the commit server is “Not running”. The commit server starts running only when a commit job is added to the batch.</div>	
Options	This command has no options.
Required Privilege Level	view
Related Documentation	<ul style="list-style-type: none"><li>• <a href="#">Example: Configuring Batch Commit Server Properties on page 117</a></li></ul>
List of Sample Output	<a href="#">show system commit server status (When Server Is Inactive) on page 360</a> <a href="#">show system commit server status (When Server Is Active) on page 360</a>

### Sample Output

#### show system commit server status (When Server Is Inactive)

```
user@host> show system commit server status
Commit server status : Not running
```

#### show system commit server status (When Server Is Active)

```
user@R0> show system commit server status

Commit server status : Running
Jobs in process:
 1369 1370 1371
```