JUNIPER NETWORKS | Engineering Simplicity

# Junos® OS

## CLI User Guide

Published
2020-09-21

junos

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

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, 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 © 2020 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 https://support.juniper.net/support/eula/. 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**

Using the Examples in This Manual | **xv**

- Merging a Full Example | **xvi**
- Merging a Snippet | **xvii**

Documentation Conventions | **xvii**

Documentation Feedback | **xx**

Requesting Technical Support | **xx**

- Self-Help Online Tools and Resources | **xxi**
- Creating a Service Request with JTAC | **xxi**

1 **Overview**

**About the CLI Guide and Junos OS Documentation | 23**

**CLI Overview | 23**

- Introducing the Junos OS Command-Line Interface | **24**
  - Key Features of the CLI | **24**
- Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies | **26**
  - CLI Command Hierarchy | **26**
  - Configuration Statement Hierarchy | **26**
  - Moving Among Hierarchy Levels | **27**
- Other Tools to Configure and Monitor Devices Running Junos OS | **28**
- Configuring Junos OS in a FIPS Environment | **28**

2 **Getting Started**

**Getting Started: A Quick Tour of the CLI | 31**

- Getting Started with the Junos OS Command-Line Interface | **31**
- Switching Between Junos OS CLI Operational and Configuration Modes | **34**
- Using Keyboard Sequences to Move Around and Edit the Junos OS CLI | **36**
- Configuring a User Account on a Device Running Junos OS | **37**
- Using the CLI Editor in Configuration Mode | **40**

Checking the Status of a Device Running Junos OS | 42

Rolling Back Junos OS Configuration Changes | 45

Configuring a Routing Protocol | 47

Shortcut | 47

Longer Configuration | 48

Making Changes to a Routing Protocol Configuration | 51

**Online Help in the CLI | 54**

Getting Online Help from the Junos OS Command-Line Interface | 55

Getting Help About Commands | 55

Getting Help About a String in a Statement or Command | 57

Getting Help About Configuration Statements | 57

Getting Help About System Log Messages | 58

Junos OS CLI Online Help Features | 58

Help for Omitted Statements | 58

Using CLI Command Completion | 59

Using Command Completion in Configuration Mode | 59

Displaying Tips About CLI Commands | 60

CLI Explorer Overview | 61

**CLI Environment Settings | 62**

Customizing the Junos OS CLI Environment | 62

Displaying the Current CLI Settings | 63

Setting the Terminal Type | 63

Setting the CLI Prompt | 63

Setting the CLI Directory | 64

Setting the CLI Timestamp | 64

Setting the Idle Timeout | 64

Setting the CLI to Prompt for Restart After a Software Upgrade | 64

Setting CLI Command Completion | 65

Setting the Junos OS CLI Screen Length and Width | 65

Setting the Screen Length | 66

Setting the Screen Width | 66

Enabling Configuration Breadcrumbs | 67

3

## Using Configuration Statements to Configure a Device


### CLI Configuration Mode Overview | 71

Understanding Junos OS CLI Configuration Mode | 71

Configuration Mode Commands | 72

Configuration Statements and Identifiers | 74

Configuration Statement Hierarchy | 75

Entering and Exiting the Junos OS CLI Configuration Mode | 77

Issuing Relative Junos OS Configuration Mode Commands | 80

Using Command Completion in Configuration Mode | 81

Notational Conventions Used in Junos OS Configuration Hierarchies | 83

### Configure Command Overview | 85

Forms of the configure Command | 85

Using the configure Command | 86

Using the configure exclusive Command | 87

Updating the configure private Configuration | 89

### Modifying the Configuration for a Device | 90

Displaying Users Currently Editing the Junos OS Configuration | 91

Modifying the Junos OS Configuration | 92

Adding Junos OS Configuration Statements and Identifiers | 93

Deleting a Statement from a Junos OS Configuration | 94

Example: Deleting a Statement from the Junos OS Configuration | 98

Copying a Junos OS Statement in the Configuration | 99

Example: Copying a Statement in the Junos Configuration | 100

Example: Replacing a Configuration | 103

Inserting a New Identifier in a Junos OS Configuration | 109

Example: Inserting a New Identifier in a Junos Configuration | 110

Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration | 114

Example: Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration | 115

Using Global Replace in the Junos OS Configuration | 117

Common Regular Expressions to Use with the replace Command | 118

Example: Using Global Replace in a Junos OS Configuration—Using the \n Back Reference | 120

Example: Using Global Replace in a Junos OS Configuration—Replacing an Interface Name | 122

Example: Using Global Replace in a Junos OS Configuration—Using the upto Option | 125

Adding Comments in a Junos OS Configuration | 127

Adding Comments in the CLI | 127

Adding Comments in a File | 128

Example: Including Comments in a Junos OS Configuration by Using the CLI | 129

**Using Configuration Groups to Quickly Configure Devices | 132**

Understanding Junos OS Configuration Groups | 133

Configuration Groups Overview | 133

Inheritance Model | 134

Configuring Configuration Groups | 134

Creating a Junos OS Configuration Group | 135

Applying a Junos OS Configuration Group | 137

Example: Creating and Applying Junos OS Configuration Groups | 138

Disabling Inheritance of a Junos OS Configuration Group | 139

Using the junos-defaults Configuration Group | 141

Using Wildcards with Configuration Groups | 143

Improving Commit Time When Using Configuration Groups | 146

Example: Configuring Sets of Statements with Configuration Groups | 146

Example: Configuring Interfaces Using Configuration Groups | 148

Example: Configuring a Consistent IP Address for the Management Interface Using Configuration Groups | 151

Example: Configuring Peer Entities Using Configuration Groups | 152

Example: Establishing Regional Configurations Using Configuration Groups | 154

Example: Configuring Wildcard Configuration Group Names | 156

Example: Referencing the Preset Statement from the Junos OS defaults Group | 157

Example: Viewing Default Statements That Have Been Applied to the Configuration | 158

Setting Up Routing Engine Configuration Groups | 159

Using Conditions to Apply Configuration Groups | 162

Example: Configuring Conditions for Applying Configuration Groups | 162

**Viewing the Configuration | 165**

Displaying the Current Junos OS Configuration | **165**

Example: Displaying the Current Junos OS Configuration | **167**

Displaying Additional Information About the Junos OS Configuration | **168**

Displaying set Commands from the Junos OS Configuration | **171**

> Example: Displaying set Commands from the Configuration | **172**
>
> Example: Displaying set Commands with the match Option | **173**

**Verifying the Junos OS Configuration | 174**

**Committing a Configuration | 174**

Junos OS Commit Model for Configurations | **175**

Committing a Junos OS Configuration | **176**

Commit Operation When Multiple Users Configure the Software | **178**

Commit Preparation and Activation Overview | **179**

Committing Junos OS Configurations in Two Steps: Preparation and Activation | **180**

Activating a Junos OS Configuration but Requiring Confirmation | **182**

Scheduling a Junos OS Commit Operation | **183**

Monitoring the Junos OS Commit Process | **185**

Adding a Comment to Describe the Committed Configuration | **186**

Junos OS Batch Commits Overview | **187**

> Aggregation and Error Handling | **187**

Example: Configuring Batch Commit Server Properties | **188**

Backing Up the Committed Configuration on the Alternate Boot Drive | **198**

4 **Managing Configurations**

**Configuration Files Overview | 200**

Understanding Configuration Files | **200**

> Configuration File Terms | **200**

Understanding How the Junos OS Configuration Is Stored | **201**

**Managing Configurations | 202**

Understanding the show | compare | display xml Command Output | **203**

> Adding a Statement (create Operation) | **204**
>
> Deleting a Statement (delete Operation) | **205**
>
> Changing a Statement (delete and create Operations) | **207**

Changing Metadata (inactive Attribute and Operation) | **208**

Adding an Annotation (comment Tag and create Operation) | **209**

Changing an Annotation (comment Tag, and delete and create Operations) | **210**

Adding a Statement Inside a Container (create Operation, and insert and key Attributes) | **211**

Changing the Order Inside a Container (merge Operation, and insert and key Attributes) | **211**

Returning to the Most Recently Committed Junos OS Configuration | **212**

Returning to a Previously Committed Junos OS Configuration | **213**

Returning to a Configuration Prior to the One Most Recently Committed | **213**

Displaying Previous Configurations | **214**

Comparing Configuration Changes with a Prior Version | **215**

Saving a Configuration to a File | **217**

Compressing the Current Configuration File | **218**

Freeing Up System Storage Space | **220**

Cleaning Up Files with the CLI | **221**

**Autoinstallation of Configuration Files | 223**

Understanding Autoinstallation of Configuration Files | **223**

Typical Uses for Autoinstallation | **224**

Autoinstallation Configuration Files and IP Addresses | **224**

Typical Autoinstallation Process on a New Device | **225**

Configuring Autoinstallation of Configuration Files (CLI Procedure) | **226**

**Loading Configuration Files | 228**

Loading a Configuration from a File or the Terminal | **229**

Understanding Character Encoding on Devices Running Junos OS | **232**

Additional Details About Specifying Junos OS Statements and Identifiers | **234**

Specifying Statements | **234**

Performing CLI Type Checking | **236**

Loading a Configuration from a File | **238**

Uploading a Configuration File | **240**

**Backing Up Configurations to an Archive Site | 242**

Configuring the Transfer of the Currently Active Configuration | **242**

**Factory Default Configuration | 244**

Reverting to the Default Factory Configuration | **245**

**Rescue Configuration | 245**

Creating and Returning to a Rescue Configuration | **246**

**Encrypting and Decrypting Configuration Files | 247**

Encrypting Configuration Files | **247**

Decrypting Configuration Files | **249**

Modifying the Encryption Key | **250**

**Synchronizing Configurations Across Routing Engines | 250**

Synchronizing Routing Engines | **251**

Configuring Multiple Routing Engines to Synchronize Committed Configurations Automatically | **254**

5 **Using Operational Commands to Monitor Devices**

**CLI Operational Mode Overview | 258**

Overview of Junos OS CLI Operational Mode Commands | **258**

CLI Command Categories | **258**

Commonly Used Operational Mode Commands | **259**

Understanding the Display Options of the show Command | **260**

Interface Naming Conventions Used in the Junos OS Operational Commands | **262**

Physical Part of an Interface Name | **262**

Logical Part of an Interface Name | **262**

Channel Identifier Part of an Interface Name | **263**

Using Wildcard Characters in Interface Names | **264**

**Using Operational Commands to Monitor a Device | 264**

Using the Junos OS CLI Command Completion | **265**

Controlling the Scope of an Operational Mode Command | **266**

Routing Matrix Command Options | **268**

Viewing Files and Directories on a Device Running Junos OS | **268**

Directories on the Device | **269**

Listing Files and Directories | **269**

Specifying Filenames and URLs | 272

Displaying Junos OS Information | 274

Managing Programs and Processes Using Junos OS Operational Mode Commands | 274

Showing Software Processes | 275

Restarting the Junos OS Process | 276

Stopping Junos OS | 278

Rebooting Junos OS | 279

Using the Junos OS CLI Comment Character # for Operational Mode Commands | 280

**Filtering Operational Command Output | 280**

Using the Pipe ( | ) Symbol to Filter Junos OS Command Output | 281

Using Regular Expressions with the Pipe ( | ) Symbol to Filter Junos OS Command Output | 282

Pipe ( | ) Filter Functions in the Junos OS Command-Line Interface | 284

Comparing Configurations and Displaying the Differences in Text | 285

Comparing Configurations and Displaying the Differences in XML | 286

Counting the Number of Lines of Output | 287

Displaying Output in XML Tag Format | 287

Displaying Static Configuration Data | 288

Displaying Ephemeral Configuration Data | 289

Displaying Output in JSON Format | 289

Displaying the Configuration with YANG Translation Scripts Applied | 290

Displaying the RPC Tags for a Command | 292

Ignoring Output That Does Not Match a Regular Expression | 293

Displaying Output from the First Match of a Regular Expression | 293

Retaining Output After the Last Screen | 294

Displaying Output Beginning with the Last Entries | 294

Displaying Output That Matches a Regular Expression | 295

Preventing Output from Being Paginated | 295

Sending Command Output to Other Users | 295

Resolving IP Addresses | 296

Saving Output to a File | 296

Appending Output to a File | 297

Displaying Output on Screen and Writing to a File | 297

Trimming Output by Specifying the Starting Column | 298

Refreshing the Output of a Command | 298

Filtering Operational Mode Command Output in a QFabric System | 298

6 Junos OS Configuration Statements

apply-groups | 301

apply-groups-except | 302

archival | 303

autoinstallation | 305

commit activate | 306

commit prepare | 308

export-format | 310

groups | 312

no-hidden-commands | 315

server (Batch Commits) | 316

synchronize | 318

traceoptions (Batch Commits) | 321

7 Junos OS CLI Commands

activate | 326

annotate | 327

clear log | 329

clear system commit | 331

clear system commit prepared | 333

commit | 335

configure | 341

copy | 343

deactivate | 344

delete | 345

edit | 347

exit | 348

file | 349

help | 350

insert | 351

load | 352

| (pipe) | 355

protect | 359

quit | 360

rename | 361

replace | 363

request | 364

request system commit server pause | 366

request system commit server queue cleanup | 367

request system commit server start | 369

request system configuration rescue delete | 370

request system configuration rescue save | 371

restart | 372

rollback | 385

run | 386

save | 387

set | 389

set cli complete-on-space | 390

set cli directory | 392

set cli idle-timeout | 393

set cli prompt | 395

set cli restart-on-upgrade | 396

set cli screen-length | 397

set cli screen-width | 398

set cli terminal | 399

set cli timestamp | 400

set date | 402

show | 404

show cli | 405

show cli authorization | 407

show cli directory | 411

show cli history | 412

show configuration | 413

show | display inheritance | 417

show | display omit | 420

show | display set | 421

show | display set relative | 423

show groups junos-defaults | 425

show system commit | 427

show system commit server queue | 431

show system commit server status | 436

show system configuration archival | 438

show system configuration rescue | 439

show system rollback | 441

status | 443

test configuration | 444

top | 446

unprotect | 447

up | 448

update | 449

wildcard delete | 450

# About the Documentation

IN THIS SECTION


- Documentation and Release Notes | xv
- Using the Examples in This Manual | xv
- Documentation Conventions | xvii
- Documentation Feedback | xx
- Requesting Technical Support | xx


The Junos OS command-line interface (CLI) is a Juniper Networks specific command shell that runs on top of a FreeBSD UNIX-based operating system kernel. Using 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. This page contains information about the Junos OS CLI.

## 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 https://www.juniper.net/documentation/.

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 https://www.juniper.net/books.

## 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.xsl; }
   ```

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 xviii 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:<br>user@host> **configure** |
| `Fixed-width text like this` | Represents output that appears on the terminal screen. | `user@host>` **show chassis alarms**<br>`No alarms currently active` |
| *Italic text like this* | • Introduces or emphasizes important new terms.<br>• Identifies guide names.<br>• Identifies RFC and Internet draft titles. | • A policy *term* is a named structure that defines match conditions and actions.<br>• *Junos OS CLI User Guide*<br>• RFC 1997, *BGP Communities Attribute* |

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

| Convention | Description | Examples |
|---|---|---|
| *Italic text like this* | Represents variables (options for which you substitute a value) in commands or configuration statements. | Configure the machine's domain name:<br>[edit]<br>root@# **set system domain-name** *domain-name* |
| **Text like this** | Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components. | • To configure a stub area, include the **stub** statement at the **[edit protocols ospf area area-id]** hierarchy level.<br>• The console port is labeled **CONSOLE**. |
| < > (angle brackets) | Encloses optional keywords or variables. | **stub <default-metric *metric*>;** |
| \| (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**<br>**(*string1* \| *string2* \| *string3*)** |
| # (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 [ *community-ids* ]** |
| Indention and braces ( { } ) | Identifies a level in the configuration hierarchy. | [edit]<br>routing-options {<br>  static {<br>  route default {<br>  nexthop *address*;<br>  retain;<br>  }<br>  }<br>} |
| ; (semicolon) | Identifies a leaf statement at a configuration hierarchy level. | |
| **GUI Conventions** | | |

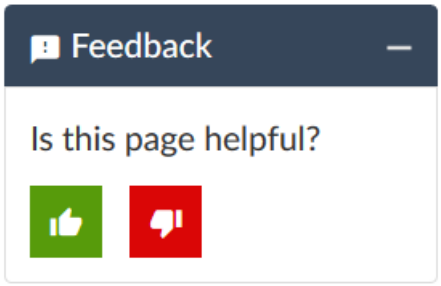
Table 2: Text and Syntax Conventions *(continued)*

| Convention | Description | Examples |
|---|---|---|
| **Bold text like this** | Represents graphical user interface (GUI) items you click or select. | • In the Logical Interfaces box, select **All Interfaces**.<br>• 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 so that we can improve our documentation. You can use either of the following methods:

- Online feedback system—Click TechLibrary Feedback, on the lower right of any page on the Juniper Networks TechLibrary site, and do one of the following:




  - Click the thumbs-up icon if the information on the page was helpful to you.
  - Click the thumbs-down icon if the information on the page was not helpful to you or if you have suggestions for improvement, and use the pop-up form to provide 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 Juniper Care or Partner Support Services 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 https://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf.
- Product warranties—For product warranty information, visit https://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: https://www.juniper.net/customers/support/
- Search for known bugs: https://prsearch.juniper.net/
- Find product documentation: https://www.juniper.net/documentation/
- Find solutions and answer questions using our Knowledge Base: https://kb.juniper.net/
- Download the latest versions of software and review release notes: https://www.juniper.net/customers/csc/software/
- Search technical bulletins for relevant hardware and software notifications: https://kb.juniper.net/InfoCenter/
- Join and participate in the Juniper Networks Community Forum: https://www.juniper.net/company/communities/
- Create a service request online: https://myjuniper.juniper.net

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

### Creating a Service Request with JTAC

You can create a service request with JTAC on the Web or by telephone.

- Visit https://myjuniper.juniper.net.
- 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 https://support.juniper.net/support/requesting-support/.

1
CHAPTER

# Overview

About the CLI Guide and Junos OS Documentation | 23

CLI Overview | 23

# About the CLI Guide and Junos OS Documentation

The Junos OS CLI Guide explains how to use the Junos OS command-line interface (CLI), as well as provides an understanding of advanced Junos OS and device configuration topics

In this guide, we cover:

- Using configuration statements to configure network devices
- Managing device configurations
- Using operational commands to monitor devices
- Syntax for configuration statements, operational commands, and environmental commands

For a basic introduction to Junos OS, see the *Getting Started Guide for Junos OS*. It provides a high-level description of Junos OS, describes how to access devices, and provides simple step-by-step instructions for initial device configuration.

For a technical and detailed exploration of Junos OS, see the *Overview for Junos OS*. It further explains how Junos OS works, as well as security, configuration, monitoring, and managing network devices.

Another useful learning resource is Day One: Exploring the Junos CLI.

RELATED DOCUMENTATION

> Day One: Exploring the Junos CLI
>
> *Getting Started Guide for Junos OS*
>
> *Overview for Junos OS*

# CLI Overview

**IN THIS SECTION**


- Introducing the Junos OS Command-Line Interface | **24**
- Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies | **26**
- Other Tools to Configure and Monitor Devices Running Junos OS | **28**
- Configuring Junos OS in a FIPS Environment | **28**

The Junos OS command-line interface (CLI) is the software interface used to access your device. You use the Junos OS CLI to configure the device, monitor its operations, and adjust the configuration as needed, using a console connection interface or through a network connection.

## Introducing the Junos OS Command-Line Interface

The Junos OS CLI is a Juniper Networks-specific command shell that runs on top of a FreeBSD UNIX-based operating system kernel. Through the use of 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.

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. 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. A configuration is stored as a hierarchy of configuration statements.

  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 revised configuration on the device. Candidate configurations enable you to alter your configuration without causing potential damage to your current network operations.

### 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 specific device type 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. 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.

SEE ALSO

Getting Started with the Junos OS Command-Line Interface | 31

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

IN THIS SECTION


- CLI Command Hierarchy | 26
- Configuration Statement Hierarchy | 26
- Moving Among Hierarchy Levels | 27


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

## 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.

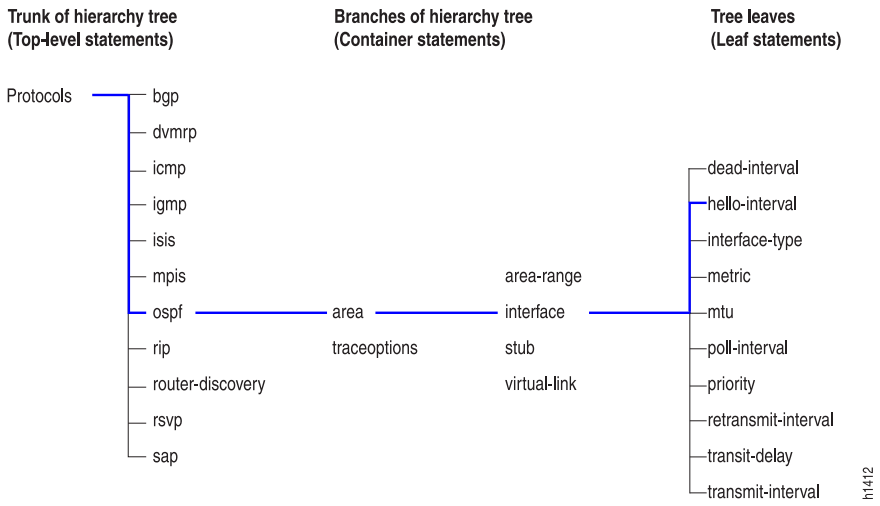
To execute a command, enter the full command name, starting at the top level of the hierarchy. For example, to display a brief view of the routes in the routing table, use the command **show route brief**.

## 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 the container and leaf statements together form the *configuration hierarchy*.

The following illustration shows 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 1: Configuration Statement Hierarchy Example**




## Moving Among Hierarchy Levels

The following table shows the CLI commands used to navigate the levels of the configuration statement hierarchy.

**Table 3: CLI Configuration Mode Navigation Commands**

| Command | Description |
|---|---|
| **edit *hierarchy-level*** | 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. |

SEE ALSO

Getting Started with the Junos OS Command-Line Interface | 31

## 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)—Available on select Juniper Networks devices, the J-Web 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—The Junos XML management protocol allows you to monitor and configure Juniper Networks devices. 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)—You 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 User 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 device. For more information about MIBs, see the *Network Management and Monitoring Guide*.

SEE ALSO

Getting Started with the Junos OS Command-Line Interface | 31

## Configuring Junos OS in a FIPS Environment

Junos-FIPS enables you to configure a network of Juniper Networks devices 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 the secure environment complies 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.

SEE ALSO

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

RELATED DOCUMENTATION

> Day One: Exploring the Junos CLI

2
CHAPTER

# Getting Started

Getting Started: A Quick Tour of the CLI | **31**

Online Help in the CLI | **54**

CLI Environment Settings | **62**

# Getting Started: A Quick Tour of the CLI

IN THIS SECTION


- Getting Started with the Junos OS Command-Line Interface | 31
- Switching Between Junos OS CLI Operational and Configuration Modes | 34
- Using Keyboard Sequences to Move Around and Edit the Junos OS CLI | 36
- Configuring a User Account on a Device Running Junos OS | 37
- Using the CLI Editor in Configuration Mode | 40
- Checking the Status of a Device Running Junos OS | 42
- Rolling Back Junos OS Configuration Changes | 45
- Configuring a Routing Protocol | 47


The following topics can help you get started with the Junos OS CLI to perform configuration changes, switch between operational mode and configuration mode, create a user account, and execute some of the basic commands.

> **NOTE:** If you need a basic introduction to Junos OS, see the *Getting Started Guide for Junos OS*. For more in-depth information, as well as to learn how to use Junos OS with Juniper Networks devices, see the *Overview for Junos OS*. This Junos OS CLI Guide generally assumes you are at least familiar with the content in the other two guides, as well as Junos OS concepts and operation principles in general.

## Getting Started with the Junos OS Command-Line Interface

This topic shows you how to start the Junos OS CLI, view the command hierarchy, and make small configuration changes. The related topics listed at the end of this topic provide more detailed information about using the CLI.

> **NOTE:** 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 device 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, 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 demonstrates some examples showing 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
```

SEE ALSO

Getting Online Help from the Junos OS Command-Line Interface | 55
Using the Junos OS CLI Command Completion | 265

## 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 or hash sign (**#**).

To switch between operational mode and configuration mode:

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

```
---  JUNOS 17.2B1.8 built 2018-05-09 23:41:29 UTC
% cli
user@host>
```

2. To enter configuration mode, type the **configure** command or the **edit** command from the CLI operational mode. The prompt in brackets (**[edit]**), also known as a *banner*, shows that you are in configuration edit mode at the top of the hierarchy. 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 device while you are modifying the VRRP configuration for a backup device:

```
[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 ...
```

SEE ALSO

Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies | 26

Getting Online Help from the Junos OS Command-Line Interface | 55

## 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. The following table lists some of the CLI keyboard sequences. They are the same as those used in Emacs.

**Table 4: CLI Keyboard Shortcuts**

| Keyboard sequence | Action |
|---|---|
| Ctrl+b | Move the cursor back one character. |
| Esc+b or Alt+b | Move the cursor back one word. |
| Ctrl+f | Move the cursor forward one character. |
| Esc+f or Alt+f | Move the cursor forward one word. |
| Ctrl+a | Move the cursor to the beginning of the command line. |
| Ctrl+e | Move the cursor to the end of the command line. |
| Ctrl+h, Delete, or Backspace | Delete the character before the cursor. |
| Ctrl+d | Delete the character at the cursor. |
| Ctrl+k | Delete the all characters from the cursor to the end of the command line. |
| Ctrl+u or Ctrl+x | Delete the all characters from the command line. |
| Ctrl+w, Esc+Backspace, or Alt+Backspace | Delete the word before the cursor. |
| Esc+d or Alt+d | Delete the word after the cursor. |
| Ctrl+y | Insert the most recently deleted text at the cursor. |
| Ctrl+l | Redraw the current line. |
| Ctrl+p | Scroll backward through the list of recently executed commands. |
| Ctrl+n | Scroll forward through the list of recently executed commands. |

**Table 4: CLI Keyboard Shortcuts *(continued)***

| Keyboard sequence | Action |
|---|---|
| Ctrl+r | Search the CLI history incrementally in reverse order for lines matching the search string. |
| Esc+/ or Alt+/ | Search the CLI history for words for which the current word is a prefix. |
| Esc+. or Alt+ | Scroll backward through the list of recently entered words in a command line. |
| Esc+*number sequence*<br>or Alt+*number sequence* | Specify the number of times to execute a keyboard sequence. |

SEE ALSO

Using Global Replace in the Junos OS Configuration | **117**

Using Wildcard Characters in Interface Names | **264**

## 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 (**[edit]**) prompt 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 17.2B1.8 built 2018-05-09 23:41:29 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.

SEE ALSO

## Using the CLI Editor in Configuration Mode

This topic describes some of the basic commands that you can 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.<br><br>When you start the CLI, the device is in operational mode. You must explicitly enter configuration mode. When you do, the CLI prompt changes from **user@host>** to **user@host#** and the hierarchy level appears in square brackets. | configure | user@host> **configure**<br><br>[edit]<br>user@host# |
| Create a statement hierarchy.<br><br>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. | edit *hierarchy-level value* | [edit]<br>user@host# **edit security zones security-zone myzone**<br><br>[edit security zones security-zone myzone]<br>user@host# |
| Create a statement hierarchy and set identifier values.<br><br>The **set** command is like **edit** except that your current level in the hierarchy does not change. | set *hierarchy-level value* | [edit]<br>user@host# **set security zones security-zone myzone**<br><br>[edit]<br>user@host# |
| **Navigate the Hierarchy** | | |
| Navigate down to an existing hierarchy level. | edit *hierarchy-level* | [edit]<br>user@host# **edit security zones**<br><br>[edit security zones]<br>user@host# |

| Task | Command/Statement | Example |
|---|---|---|
| Navigate up one level in the hierarchy. | up | [edit security zones]<br>user@host# **up**<br><br>[edit security]<br>user@host# |
| Navigate to the top of the hierarchy. | top | [edit security zones]<br>user@host# **top**<br><br>[edit]<br>user@host# |
| **Commit or Revert Changes** | | |
| Commit your configuration. | commit | [edit]<br>user@host# **commit**<br><br>commit complete |
| Roll back changes from the current session.<br><br>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. | rollback | [edit]<br>user@host# **rollback**<br><br>load complete |
| **Exit Configuration Mode** | | |
| Commit the configuration and exit configuration mode. | commit and-quit | [edit]<br>user@host# **commit and-quit**<br><br>user@host> |
| Exit configuration mode without committing your configuration.<br><br>You must navigate to the top of the hierarchy using the **up** or **top** commands before you can exit configuration mode. | exit | [edit]<br>user@host# **exit**<br><br>The configuration has been changed but not committed<br>Exit with uncommitted changes? [yes,no] (yes) |
| **Get Help** | | |

| Task | Command/Statement | Example |
| --- | --- | --- |
| Display a list of valid options for the current hierarchy level. | ? | [edit ]<br>user@host# **edit security zones ?**<br><br>Possible completions:<br><[Enter]> Execute this command<br>> functional-zone Functional zone<br>> security-zone Security zones<br>\| Pipe through a command<br>[edit] |

SEE ALSO


Understanding Junos OS CLI Configuration Mode | 71

Entering and Exiting the Junos OS CLI Configuration Mode | 77

Displaying the Current Junos OS Configuration | 165


## Checking the Status of a Device Running Junos OS

In operational mode, 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
```

SEE ALSO

Managing Programs and Processes Using Junos OS Operational Mode Commands | 274

Viewing Files and Directories on a Device Running Junos OS | 268

## Rolling Back Junos OS Configuration Changes

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

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.

SEE ALSO

Returning to the Most Recently Committed Junos OS Configuration | 212

## Configuring a Routing Protocol

IN THIS SECTION


- Shortcut | 47
- Longer Configuration | 48
- Making Changes to a Routing Protocol Configuration | 51


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;
            }
        }
    }
}
```

### 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** command:

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

   Notice that the prompt has changed to a pound or hash 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 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 return up a level and view what you have done 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. Move up to the top level and review 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 device:

```
[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#
```

SEE ALSO

*Service Now*

Interface Naming Conventions Used in the Junos OS Operational Commands | 262

# Online Help in the CLI

**IN THIS SECTION**


- Getting Online Help from the Junos OS Command-Line Interface | 55
- Junos OS CLI Online Help Features | 58
- CLI Explorer Overview | 61

# Getting Online Help from the Junos OS Command-Line Interface

IN THIS SECTION


- Getting Help About Commands | 55
- Getting Help About a String in a Statement or Command | 57
- Getting Help About Configuration Statements | 57
- Getting Help About System Log Messages | 58


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.

## 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 context-relevant 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, this is the result:

```
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 re-displays the command names and options 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 re-displays 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 is 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.

- The range of supported characters for attributes is 0 through 65499 characters.
- The 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.

SEE ALSO

Getting Started with the Junos OS Command-Line Interface | 31

## Junos OS CLI Online Help Features

**IN THIS SECTION**


- Help for Omitted Statements | 58
- Using CLI Command Completion | 59
- Using Command Completion in Configuration Mode | 59
- Displaying Tips About CLI Commands | 60


### Help for Omitted Statements

If you have omitted a required statement at a specific 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 Space. 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 Space.

### 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 later. 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>
```

SEE ALSO

Using the Junos OS CLI Command Completion | 265

# CLI Explorer Overview

CLI Explorer is a Web application that helps you to explore Junos OS configuration statements and commands. It lists all the configuration statements and commands supported in the Junos OS across different platforms on several products.

To view the available configuration statements and commands, you can use any of the following filtering options:

- Filter by product family—To find the CLI reference information by product family, you can either select “All products” or select any specific product.

  For example: ACX Series, EX Series.

- Filter by number or letter—To find the CLI reference information by number or letter, you can either select “All” or filter by numbers “3” or “8” or any of the letters (“A”, “B”, “C”...).

  For example, if you select the letter “A”, commands such as **aaa**, **aaa clients (TDF)**, **aaa-access-profile (L2TP LNS)** appear.

- Filter by the normal search option—To use this option to filter the commands and statements, you enter your search criteria.

  For example, if you enter the number “3”, all the commands and statements containing the number “3” appear in the search results.

When you click on the link in the search results, you are directed to a page describing the command or statement that is referenced in a user guide.

To explore the Junos OS configuration statements and commands, see the CLI Explorer.

SEE ALSO

Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies | 26

# CLI Environment Settings

IN THIS SECTION


- Customizing the Junos OS CLI Environment | 62
- Setting the Junos OS CLI Screen Length and Width | 65
- Enabling Configuration Breadcrumbs | 67


In operational mode, you can customize the Junos OS command-line interface (CLI) environment to suit your specific preferences and requirements.

## Customizing the Junos OS CLI Environment

IN THIS SECTION


- Displaying the Current CLI Settings | 63
- Setting the Terminal Type | 63
- Setting the CLI Prompt | 63
- Setting the CLI Directory | 64
- Setting the CLI Timestamp | 64
- Setting the Idle Timeout | 64
- Setting the CLI to Prompt for Restart After a Software Upgrade | 64
- Setting CLI Command Completion | 65


In operational mode, you can customize the Junos OS CLI environment by using the **set cli** command. For example, you can specify the number of lines that are displayed on the screen or your terminal type. The following output lists the available options:

```
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 device or log in from the console when its terminal type is already configured , your terminal type, screen length, and screen width are already set.

### Displaying the Current 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
```

### 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
```

**NOTE:** Changing the CLI prompt is not persistent across CLI sessions. When you exit the CLI and restart it, the prompt defaults back to **user@host**.

### Setting the CLI Directory

To set the current working directory, use the **set cli directory** command:

```
user@host> set cli directory directory
```

The *directory* should be the full pathname of desired working directory. After entering this command, the CLI will switch to the specified 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]
```

Enclose the format in single quotation marks ( ‘). If you do not specify a timestamp format, the default format is *'Mmm dd hh:mm:ss'* (for example, Feb 08 17:20:49).

### Setting the Idle Timeout

By default, a CLI session never times out after extended idle time, 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 device, use the **set cli idle-timeout** command:

```
user@host> set cli idle-timeout timeout
```

The *timeout* can be 0 through 100,000 minutes. Setting the *timeout* to 0 disables the idle timeout.

### Setting the CLI to Prompt for Restart After a Software Upgrade

By default, the CLI prompts you to restart after a software upgrade. To disable the prompt, 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 CLI Command Completion

By default, you can press Tab or the spacebar to have the CLI complete a command.

To have the CLI allow only 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 enable the use of the spacebar (as well as Tab) 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>
```

## Setting the Junos OS CLI Screen Length and Width

**IN THIS SECTION**


- Setting the Screen Length | 66
- Setting the Screen Width | 66


You can set the Junos OS CLI screen length and width according to your specific preferences and requirements.

### Setting the Screen Length

The default CLI screen length is 24 lines. If output is longer than this, the display will scroll to the configured screen length, then display a **more** prompt. You can press Enter to display the next line, or the Spacebar to show the next full screen. Alternatively, you can press **h** for all the available options, which include navigation, searching, and saving.

To change the screen length, use the **set cli screen-length** command:

```
user@host> set cli screen-length length
```

Setting the screen length to 0 lines disables the use of “one screen at a time” output, so that it will scroll all the way through to completion without displaying the **more** prompt. 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 can be 0 or in the range of 40 through 1024. The default CLI screen width is 80 characters. Using a CLI screen width value of 0 disables the display of the output screen, which may be desirable when using scripts. To change the width, use the **set cli screen-width** command:

```
user@host> set cli screen-width width
```

## Enabling Configuration Breadcrumbs

You can configure the output of **show configuration** operational mode command and **show** configuration mode commands to display configuration breadcrumbs. These breadcrumbs help you to know the exact location in the configuration hierarchy for 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: mxxj1-mr6 (64.12.137.160/27) (T=bblan, 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. If not there already, go to configuration mode in the CLI.

2. Define a class at the **[edit system login]** hierarchy level. Here, we are also setting an idle timeout value of 10 minutes.

   ```
   [edit system login]
   user@host# set class breadclass idle-timeout 10
   ```

3. Include 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. Add a user to the defined login class to enable the breadcrumbs output view when this user runs the **show configuration** operational mode command.

```
[edit system login user user1]
user@host# set class breadclass
```

5. Commit 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=bblan, 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. In this case, **user1** is currently viewing the interface configuration of a group.

> **NOTE:** If you are enabling configuration breadcrumbs for your own user account, log out and log in again to see the changes.

3

CHAPTER

# Using Configuration Statements to Configure a Device


CLI Configuration Mode Overview | **71**

Configure Command Overview | **85**

Modifying the Configuration for a Device | **90**

Using Configuration Groups to Quickly Configure Devices | **132**

Viewing the Configuration | **165**

Verifying the Junos OS Configuration | **174**

Committing a Configuration | **174**

# CLI Configuration Mode Overview

IN THIS SECTION


- Understanding Junos OS CLI Configuration Mode | 71
- Entering and Exiting the Junos OS CLI Configuration Mode | 77
- Issuing Relative Junos OS Configuration Mode Commands | 80
- Using Command Completion in Configuration Mode | 81
- Notational Conventions Used in Junos OS Configuration Hierarchies | 83


The configuration mode of the Junos OS CLI enables you to configure a device, using configuration statements to set, manage, and monitor device properties.

## Understanding Junos OS CLI Configuration Mode

IN THIS SECTION


- Configuration Mode Commands | 72
- Configuration Statements and Identifiers | 74
- Configuration Statement Hierarchy | 75


You can configure all Junos OS properties, 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 26, a device configuration is stored as a hierarchy of statements. In configuration mode, you create a set of configuration statements to use. When you have finished entering the configuration statements and you are certain they are complete and correct, you commit them, which activates the configuration on the device.

You can create the configuration interactively or you can create an ASCII text file containing the configuration, then load it on the device and commit it.

This topic covers:

### Configuration Mode Commands

The following table summarizes each CLI configuration mode command. The commands are organized alphabetically.

**Table 5: Summary of Configuration Mode Commands**

| Command | Description |
|---|---|
| **activate** | Remove the **inactive:** tag from a statement. 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 are ignored 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 equivalent. |
| **extension** | Manage configurations that are contributed by SDK application packages by either displaying or deleting 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 5: 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 CLI command without exiting from configuration mode. |
| **save** | Save the configuration to an ASCII file. The configuration statements up to and including the current level of the statement hierarchy 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** | 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, Junos OS searches for items that contain these patterns and deletes them. |

### Configuration Statements and Identifiers

You can configure device properties by including the corresponding statements in the configuration. Typically, a statement consists of a system-defined keyword, which is fixed text, and an optional 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 6: 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 *Junos OS Administration Library*. |
| **accounting-options** | Configure accounting statistics data collection for interfaces and firewall filters. For information about the statements in this hierarchy, see the *Network Management and Monitoring Guide*. |
| **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 *Junos OS Administration Library*. |
| **class-of-service** | Configure class-of-service parameters. For information about the statements in this hierarchy, see the Junos OS Class of Service User Guide for Routing Devices. |
| **firewall** | Configure filters that select packets based on their contents. For information about the statements in this hierarchy, see the *Routing Policies, Firewall Filters, and Traffic Policers User Guide*. |
| **forwarding-options** | Configure forwarding options, including traffic sampling options. For information about the statements in this hierarchy, see the *Junos OS Network Interfaces Library for Routing Devices*. |
| **groups** | Configure configuration groups. For information about statements in this hierarchy, see the *Junos OS Administration Library*. |
| **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 *Junos OS Network Interfaces Library for Routing Devices*. |
| **policy-options** | Configure 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 *Routing Policies, Firewall Filters, and Traffic Policers User Guide*. |

Table 6: 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 *Junos OS Routing Protocols Library* and the MPLS Applications User Guide for Routing Devices. |
| **routing-instances** | Configure one or more routing instances. For information about the statements in this hierarchy, see the *Junos OS Routing Protocols Library*. |
| **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 *Junos OS Routing Protocols Library*. |
| **security** | Configure IP Security (IPsec) services. For information about the statements in this hierarchy see the *Junos OS Administration Library*. |
| **snmp** | Configure SNMP community strings, interfaces, traps, and notifications. For information about the statements in this hierarchy, see the *Network Management and Monitoring Guide*. |
| **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 *Junos OS Administration Library*. |

## Configuration Statement Hierarchy

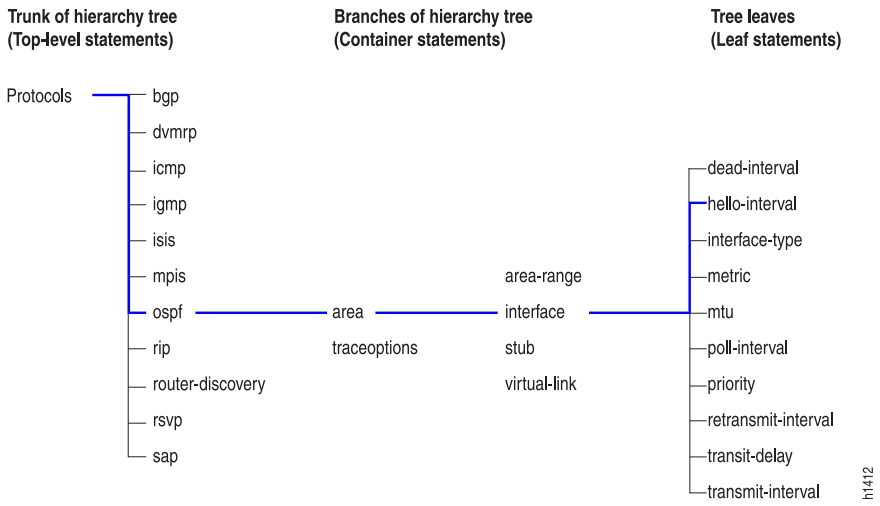
The Junos OS configuration consists of a hierarchy of statements. There are two types of statements:

- Container statements, which are branches that can contain other statements (including additional container statements or leaf statements). Container statements at the top of the hierarchy are considered to be the trunk of the hierarchy tree.
- Leaf statements (contained by container statements) which do not contain other statements

The container and leaf statements form the configuration hierarchy. Each statement at the top level of the configuration hierarchy resides at the trunk of a hierarchy tree. These 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.

The following illustration shows the hierarchy tree, illustrating a statement path for the part of the protocol configuration hierarchy responsible for configuring the **hello-interval** statement on an interface in an OSPF area.

**Figure 2: Configuration Mode Hierarchy of Statements**




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). The **hello-interval** statement is a leaf on the tree, which in this case contains a data value, namely the length of the **hello-interval**, in seconds.

The following configuration example illustrates the statement hierarchy as shown in Figure 2 on page 76:

**[edit protocols ospf area *area-number* interface *interface-name*]**

The command 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. Additionally, in general, it 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 lowest 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, **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;
}
```

## 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>
```

> **NOTE:** When making configuration changes, if you exit configuration mode without committing them, the intended changes will be lost.

You must have configure permission to view and use the **configure** command. When in configuration mode, you can view and modify only those statements for which you have access privileges.

- If you enter configuration mode and another user is also in configuration mode, a message shows the user's name and the part of the configuration the other 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
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>
```

SEE ALSO

*Junos OS Administration Library*

Switching Between Junos OS CLI Operational and Configuration Modes | 34

Using the configure exclusive Command | 87

Updating the configure private Configuration | 89
Modifying the Junos OS Configuration | 92
Displaying set Commands from the Junos OS Configuration | 171
Commit Operation When Multiple Users Configure the Software | 178
Managing Programs and Processes Using Junos OS Operational Mode Commands | 274

## Issuing Relative Junos OS Configuration Mode Commands

The **top** or **up** command followed by another configuration command—such as **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 and 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
```

SEE ALSO

Displaying the Current Junos OS Configuration | 165

## Using Command Completion in Configuration Mode

This topic shows you how to access command help and to use basic command completion in the Junos OS CLI configuration mode. In each case, it is the use of the question mark (**?**) character, either by itself or in combination with a partial command or configuration statement.

To list the configuration mode commands, use the **?** command by itself:

```
[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#
```

To list all the statements available at a particular hierarchy level, use **?** after the name of the hierarchy level you wish to view. In this example, we are viewing the **edit** and **edit protocols** hierarchies.

```
[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
```

To list all commands that start with a particular string or letter, enter the string and/or letter and end with the **?** character. Here is a list of all the **routing-options** commands starting with the letter “a.”

user@host# **edit routing-options a?**

```
Possible completions:
> aggregate           Coalesced routes
> autonomous-system   Autonomous system number
```

This is a list of all configured Asynchronous Transfer Mode (ATM) interfaces, using just a partial completion (**at?**):

```
user@host# edit interfaces at?
```

```
<interface_name>     Interface name
  at-0/2/0             Interface name
  at-0/2/1             Interface name
[edit]
```

You can also show a list of all configured policy statements:

```
user@host# show policy-options policy-statement ?
```

```
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
```

SEE ALSO

Adding Junos OS Configuration Statements and Identifiers | 93

*Service Now*

Adding Junos OS Configuration Statements and Identifiers | 93

## 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#
```

> **NOTE:** (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 configured 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 command hierarchy level and below.

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.

SEE ALSO

> *Configuration Features in the Junos OS*
>
> *Configuration Mode Commands in the Junos OS*

# Configure Command Overview

IN THIS SECTION


- Forms of the configure Command | 85
- Using the configure Command | 86
- Using the configure exclusive Command | 87
- Updating the configure private Configuration | 89


The **configure** command is used to enter the CLI configuration mode. It can also be used to gather other information, such as other users currently in configuration mode.

## 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 are managing the network and device configuration.

**Table 7: Forms of the configure Command**

| Command | Edit Access | Commit Access |
|---|---|---|
| **configure** | • No one can lock the configuration. All users can make configuration changes.<br>• When you enter configuration mode, the CLI displays the following information:<br>  ◦ A list of other users editing the configuration.<br>  ◦ Hierarchy levels the users are viewing or editing.<br>  ◦ Whether the configuration has been changed, but not committed.<br>• When more than one user makes changes to a configuration, the most recent changes take precedence when the configuration is committed. | • All users can commit any changes to the configuration.<br>• If you and another user make changes and the other user commits changes, your changes are committed as well. |

**Table 7: Forms of the configure Command *(continued)***

| Command | Edit Access | Commit Access |
|---|---|---|
| **configure exclusive** | • One user locks the configuration and makes changes without interference from other users.<br>• If you enter configuration mode while another user has locked the configuration (with the **configure exclusive** command), the CLI displays the user's PID and the hierarchy level the user is viewing or editing.<br>• If you enter configuration mode when another user has locked the configuration, you can attempt to forcibly log out that user using the **request system logout** operational mode command. For details, see the CLI Explorer. | • Only the user who has locked the configuration can commit it.<br>• Other users can enter and exit configuration mode, but they cannot commit any changes they attempt to make to the configuration until it is unlocked. |
| **configure private** | • Multiple users can edit the configuration at the same time.<br>• Each user has a private candidate configuration to edit independently of other users.<br>• When multiple users enter conflicting configurations, the first commit operation takes precedence over subsequent commit operations. | • When you commit the configuration, the device verifies that the operational (running) configuration has not been modified by another user before accepting your private candidate configuration as the new operational configuration.<br>• If the configuration has been modified by another user, you can merge the modifications into your private candidate configuration and attempt to commit again. |

SEE ALSO

Committing a Junos OS Configuration | 176
Displaying set Commands from the Junos OS Configuration | 171
Displaying Users Currently Editing the Junos OS Configuration | 91

## Using the configure Command

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 of who 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 each person is viewing or editing.

```
user@host> configure
Entering configuration mode
Current configuration users:
root terminal p3 (pid 1088) on since 2018-05-13 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#
```

## Using the configure exclusive Command

If you enter configuration mode using 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 make any permanent changes to the configuration. This additionally means any attempted changes by other users while the configuration is in the locked state will always be discarded as soon as they exit configuration mode.

If another user has locked the configuration, and you need to forcibly log them out, use the operational mode command **request system logout pid *pid_number***. You can locate the *pid_number* in the notification you receive upon entering configuration mode when someone else has locked it for exclusive access.

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. For example, in this is case, the *pid_number* of the user who has locked the configuration for exclusive access is 1088.

```
user@host> configure
Entering configuration mode
Users currently editing the configuration:
root terminal p3 (pid 1088) on since 2018-10-30 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 any uncommitted changes and rolls backs the configuration to its previously committed state. The **no** option allows you to continue editing or to commit your changes in configure exclusive mode.

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

If you enter configuration mode with the **configure exclusive** command, and issue **commit confirmed**, but do not actually confirm the commit within the specified interval, an automatic rollback is triggered. Once automatic rollback occurs, Junos OS removes the exclusive lock from your session and as a result, the error message “access has been revoked” is displayed. This is because the session is no longer an exclusive session. What this means for you is that the configuration is back to the default state where anyone with access can edit and/or commit it. To re-lock the configuration, you need to use the **configure exclusive** command again.

```
user@host>configure exclusive
warning: uncommitted changes will be discarded on exit
Entering configuration mode
[edit]
user@host# commit confirmed 1
commit confirmed will be automatically rolled back in 1 minutes unless confirmed
commit
# commit confirmed will be rolled back in 1 minute
Commit was not confirmed; automatic rollback complete.
[edit]
user@host# commit
error: access has been revoked.
user@host# commit check
```

```
error: access has been revoked.
user@host>configure exclusive
warning: uncommitted changes will be discarded on exit
Entering configuration mode
```

If you initiate a configure exclusive session, issue **commit confirmed** and confirm the commit, the exclusive lock is retained in your session. You can continue to make changes to the configuration while still in a locked exclusive session.

```
user@host> configure exclusive
warning: uncommitted changes will be discarded on exit
Entering configuration mode
[edit]
user@host# commit confirmed 1
commit confirmed will be automatically rolled back in 1 minutes unless confirmed
commit complete
# commit confirmed will be rolled back in 1 minute
[edit]
user@host# commit
commit complete
```

SEE ALSO

Adding Junos OS Configuration Statements and Identifiers | 93

## 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.

**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.

**NOTE:** Junos OS does not support using **configure private** mode to configure statements corresponding to third-party YANG data models, for example, OpenConfig or custom YANG data models.

# Modifying the Configuration for a Device

**IN THIS SECTION**


- Displaying Users Currently Editing the Junos OS Configuration | **91**
- Modifying the Junos OS Configuration | **92**
- Adding Junos OS Configuration Statements and Identifiers | **93**
- Deleting a Statement from a Junos OS Configuration | **94**
- Example: Deleting a Statement from the Junos OS Configuration | **98**
- Copying a Junos OS Statement in the Configuration | **99**
- Example: Copying a Statement in the Junos Configuration | **100**
- Example: Replacing a Configuration | **103**
- Inserting a New Identifier in a Junos OS Configuration | **109**
- Example: Inserting a New Identifier in a Junos Configuration | **110**
- Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration | **114**
- Example: Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration | **115**
- Using Global Replace in the Junos OS Configuration | **117**
- Common Regular Expressions to Use with the replace Command | **118**
- Example: Using Global Replace in a Junos OS Configuration—Using the \n Back Reference | **120**
- Example: Using Global Replace in a Junos OS Configuration—Replacing an Interface Name | **122**
- Example: Using Global Replace in a Junos OS Configuration—Using the upto Option | **125**
- Adding Comments in a Junos OS Configuration | **127**
- Example: Including Comments in a Junos OS Configuration by Using the CLI | **129**

Junos CLI enables you to modify an existing Junos OS configuration. This section also explains the specifics of adding a statement, deleting a statement, copying a statement, and inserting a new identifier, including examples.

## 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 2018-03-01 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 (**2018-03-01 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 (**2018-10-31 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 2018-10-31 14:55:15 PST, idle 00:03:09
commit at
```

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 (**2018-11-01 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 2018-11-01 13:05:11 PST
exclusive [edit]
```

SEE ALSO

Forms of the configure Command | 85

Scheduling a Junos OS Commit Operation | 183

Using the configure Command | 86

## 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 using the **edit** and **set** commands. 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 specified 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.

Statements can be either container statements or leaf statements. A container statement can include additional container statements within it, as well as leaf statements. A leaf statement, however, stands alone. The command **edit?** displays the container statements, while **set?** displays both the container and leaf statements, using **>** to differentiate between them.

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

SEE ALSO

Issuing Relative Junos OS Configuration Mode Commands | 80
Using the configure exclusive Command | 87
Updating the configure private Configuration | 89
Displaying the Current Junos OS Configuration | 165

## 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
                              ^
syntax error, expecting <identifier>
```

SEE ALSO

Using the configure exclusive Command | 87

Displaying the Current Junos OS Configuration | 165

Additional Details About Specifying Junos OS Statements and Identifiers | 234

## 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. For example, the following command would not work:

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

Instead, you must 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***

You can 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)
```

## 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] yes
[edit protocols ospf area 0.0.0.0]
user@host# show
[edit]
user@host#
```

Unconfigure a specific property, in this case, removing the interface speed setting:

```
[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;
}
```

## 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.

## Example: Copying a Statement in the Junos Configuration

**IN THIS SECTION**


- Requirements | **100**
- Overview | **101**
- Configuration | **101**


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

### 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

In this example illustrating how to copy statements, 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;
        }
    }
}
```

## Example: Replacing a Configuration

**IN THIS SECTION**


- Requirements | **104**
- Overview | **104**
- Configuration | **104**


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. However, there are 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

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

## Overview

During 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 existing configuration uses any of the loopback interface unit numbers used in this example, you must substitute different unused loopback interface unit numbers. Otherwise, following these steps could damage the existing 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 lopbck interface";
[edit]
user@host# show interfaces lo0 unit 101
description "this is a lopbck 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 lopbck 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 40 .

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 the previous example, where the **copy** command takes the keyword **unit** before the value to be copied:

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

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, you need to navigate down to that specific hierarchy level before using the **replace** command.

5. Show the 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@hostshow 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.

## 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.

# Example: Inserting a New Identifier in a Junos Configuration

IN THIS SECTION


- Requirements | 110
- Overview | 111
- Configuration | 111


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.

As this example shows, you must create the term (or it must already exist), before you can place it using the **insert** command. The reference point for placing the term must also exist, for example, to place the term T1 before the term T2, both T1 and T2 must already exist, and be populated (Junos automatically removes empty terms).

## 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 to verify 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

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, both 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 eliminate additional types of traffic. Both 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;
}
```

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

In a Junos OS configuration, you can deactivate statements and identifiers so 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 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.

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

IN THIS SECTION


- Requirements | 115
- Overview | 115
- Configuration | 115


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

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

Before you begin this example, make sure that you 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 *Installing the Software Package on a Device 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;
        }
    }
}
```

## 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>
```

**pattern *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.

## Common Regular Expressions to Use with the replace Command

**Table 8: 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. |

**Table 8: Common Regular Expressions to Use with the replace Command *(continued)***

| Operator | Function |
|---|---|
| **.** | 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 *n*th group. |
| **&** | Back reference. Matches the entire match. |

The following table lists some replacement examples.

**Table 9: Replacement Examples**

| Command | Result |
|---|---|
| **replace pattern myrouter with router1** | Match: **myrouter**<br>Result: **router1** |
| **replace pattern "192\.168\.(.*)/24" with "10.2.\1/28"** | Match: **192.168.3.4/24**<br>Result: **10.2.3.4/28** |
| **replace pattern "1.\1" with "abc&def"** | Match: **1.1**<br>Result: **abc1.1def** |
| **replace pattern 1.1 with " abc\&def"** | Match: **1#1**<br>Result: **abc&def** |

# Example: Using Global Replace in a Junos OS Configuration—Using the \n Back Reference

IN THIS SECTION


- Requirements | 120
- Overview | 121
- Configuration | 121


This example shows how you can use a backreference to replace a pattern.

## 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.

## Example: Using Global Replace in a Junos OS Configuration—Replacing an Interface Name

**IN THIS SECTION**


- Requirements | **123**
- Overview | **123**
- Configuration | **123**


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

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 123 .

```
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.

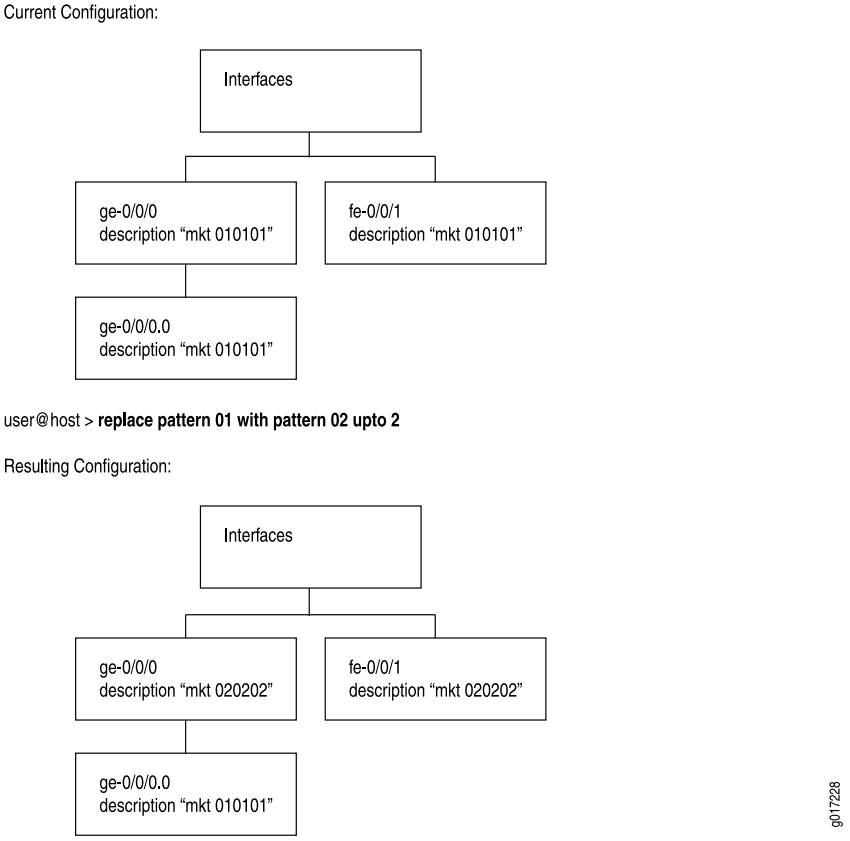
```
[edit]
user@host# show interfaces
so-1/1/0 {
    dce;
}
user@host# show protocols
ospf {
    area 0.0.0.0 {
        interface so-1/1/0.0 {
            hello-interval 5;
        }
    }
}
```

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

## Example: Using Global Replace in a Junos OS Configuration—Using the upto Option

Consider the hierarchy shown in Figure 3 on page 125. 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 3: Replacement by Object**




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;
        }
    }
}
```

# Adding Comments in a Junos OS Configuration

IN THIS SECTION


- Adding Comments in the CLI | 127
- Adding Comments in a File | 128


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

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).

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

**IN THIS SECTION**


- Requirements | **130**
- Overview | **130**
- Configuration | **130**


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

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 verify 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, 2018"
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, 2018 */
   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.

# Using Configuration Groups to Quickly Configure Devices

**IN THIS SECTION**


- Understanding Junos OS Configuration Groups | **133**
- Creating a Junos OS Configuration Group | **135**
- Applying a Junos OS Configuration Group | **137**
- Example: Creating and Applying Junos OS Configuration Groups | **138**
- Disabling Inheritance of a Junos OS Configuration Group | **139**
- Using the junos-defaults Configuration Group | **141**
- Using Wildcards with Configuration Groups | **143**
- Improving Commit Time When Using Configuration Groups | **146**
- Example: Configuring Sets of Statements with Configuration Groups | **146**
- Example: Configuring Interfaces Using Configuration Groups | **148**
- Example: Configuring a Consistent IP Address for the Management Interface Using Configuration Groups | **151**
- Example: Configuring Peer Entities Using Configuration Groups | **152**
- Example: Establishing Regional Configurations Using Configuration Groups | **154**
- Example: Configuring Wildcard Configuration Group Names | **156**
- Example: Referencing the Preset Statement from the Junos OS defaults Group | **157**
- Example: Viewing Default Statements That Have Been Applied to the Configuration | **158**
- Setting Up Routing Engine Configuration Groups | **159**

- Using Conditions to Apply Configuration Groups | 162
- Example: Configuring Conditions for Applying Configuration Groups | 162


Configuration groups are used to set up and apply common elements that are reused within the same configuration.

## Understanding Junos OS Configuration Groups

IN THIS SECTION


- Configuration Groups Overview | 133
- Inheritance Model | 134
- Configuring Configuration Groups | 134


This topic provides an overview of configuration groups and the inheritance model in the Junos OS CLI.

### 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.

For areas of your configuration to inherit configuration statements:

1. Configure statements into a configuration group. To configure configuration groups and inheritance, you can include the groups statement at the [edit] hierarchy level:

   ```
   [edit]
   groups {
       group-name {
           configuration-data;
       }
   }
   ```

2. Apply that group to the levels in the configuration hierarchy that require the statements.

   Include the **apply-groups [ *group-names* ]** statement anywhere in the configuration where the configuration statements contained in a configuration group are needed.

## Creating a Junos OS Configuration Group

The Junos OS CLI allows you to create re-usable groups containing configuration statements. You can apply these groups to to different sections of the configuration where the same configuration statements are repreated multiple times.

Configuration groups enable you to create smaller, more logically constructed configuration files, making it easier to configure and maintain Junos OS configuration. 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.

When you apply the group in the different sections of the configuration, that part of the configuration inherits the statements configured in that group. Configuration groups follow the rule of inheritance where the dynamic, ongoing relationship is set between the source of the configuration data and the target of that data. If you change the data values in the configuration group, the changes are reflected automatically in the inherited target.

You can overwrite the values in the target configuration if required, which does not affect the source in the group.

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. For example, you may want to configure all of your **ge-0/0/1** interfaces for the MTU value of 1500. To do this, you create a group with MTU value 1500:

```
[edit groups group-1]
lab@vSRX3-05# show
interfaces {
    ge-0/0/1 {
        unit 0 {
            family inet {
                mtu 1500;
            }
        }
    }
}
```

Next, you apply the group in the interface configuration.

```
[edit interfaces ge-0/0/1]
lab@vSRX3-05# set apply-groups group-1
```

View the inherited configuration.

```
[edit]
lab@vSRX3-05# show interfaces ge-0/0/1 | display inheritance
unit 0 {
    family inet {
        ##
        ## '1500' was inherited from group 'group-1'
        ##
        mtu 1500;
        address 5.0.0.254/24;
    }
}
```

In this way, if you want to configure MTU value for interface **ge-0/0/1** in different parts of configuration, you can apply the group statement using apply-groups option. If you do this manually and later want to increase the MTU, you may have to manually change every interface. If you had used a configuration group, you could just change the group config and all associated interfaces would be automatically updated.

You can also use wildcards in a configuration group to allow configuration data to be inherited by any object that matches a wildcard expression. For example:

```
[edit groups group-1]
lab@vSRX3-05# show
interfaces {
    ge-* {
        unit 0 {
            family inet {
                mtu 1500;
            }
        }
    }
}
```

SEE ALSO

Using the junos-defaults Configuration Group | **141**

Applying a Junos OS Configuration Group | **137**

*User Access and Authentication Administration Guide*

## Applying a 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.

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.

## Example: Creating and Applying Junos OS Configuration Groups

In this example illustrating the creation and application of configuration groups, the SNMP configuration is divided between the group **basic** and the normal configuration hierarchy.

There are several advantages to placing the system-specific configuration (SNMP contact) into a configuration group and thus separating it from the normal configuration hierarchy—you 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;
    }
}
```

SEE ALSO

Disabling Inheritance of a Junos OS Configuration Group | 139

## 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"
        }
    }
```

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;
    }
}
```

SEE ALSO

Applying a Junos OS Configuration Group | 137

## Using the junos-defaults Configuration Group

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.

## 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;
    }
  }
}
```

## 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 the configuration data of the group into memory 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, rather than in the process memory. This can improve commit time performance. However, it can also increase the database size.

## 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;
    }
}
```

## Example: Configuring Interfaces Using 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;
        }
    }
}
```

SEE ALSO

Interface Naming Conventions Used in the Junos OS Operational Commands | 262

## Example: Configuring a Consistent IP Address for the Management Interface Using Configuration Groups

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**.

## Example: Configuring Peer Entities Using Configuration Groups

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 {
        ##
        ## "gigether-options" was inherited from group "some-isp"
        ##
        gigether-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;
        }
      }
    }
```

## Example: Establishing Regional Configurations Using Configuration Groups

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;
      }
    }
}
```

## Example: 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 many 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;
        }
    }
}
```

## 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
            }
         }
      }
   }
}
```

## 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 omitting any comments marked with **##**, use the **no-comments** option with the **display inheritance** command.

## 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 the initial configuration for redundant Routing Engine systems and the re0 group, see *High Availability User Guide*.

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 device hostname.

   ```
   [edit groups re0]
   root# set system host-name host-name
   ```

**NOTE:** The hostname specified in the device 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 you are logged in to the CLI:

```
user-name@host-name>
```

4. Configure the IP address and prefix length for the device 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 device hostname.

```
[edit groups re1]
root# set system host-name host-name
```

9. Configure the IP address and prefix length for the device 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 ]
```

## Using Conditions to Apply Configuration Groups

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.

## Example: Configuring Conditions for Applying Configuration Groups

**IN THIS SECTION**


- Requirements | **162**
- Overview | **162**
- Configuration | **163**


This example shows how to configure conditions under which a specified configuration group is to be applied.

### Requirements

No special configuration beyond device initialization is required before you configure this example, although it should be noted that this example was configured and tested using an MX240 device.

### 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***

***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
```

# Viewing the Configuration

IN THIS SECTION


- Displaying the Current Junos OS Configuration | 165
- Example: Displaying the Current Junos OS Configuration | 167
- Displaying Additional Information About the Junos OS Configuration | 168
- Displaying set Commands from the Junos OS Configuration | 171


The **show** configuration mode command displays the current configuration for a device running Junos OS.

## Displaying the Current Junos OS Configuration

To display the current configuration for a device running Junos OS, use the **show** command in configuration mode. 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 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: 2018-07-18 11:21:58 PDT by echen
version 8.3
```

If you have omitted a required statement at a specific hierarchy level, when you issue the **show** command in configuration mode, a message indicates which statement is missing. If 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'
        }
    }
}
```

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.

## Example: Displaying the Current Junos OS Configuration

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

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 current configuration:

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

Display the configuration at a particular hierarchy:

```
[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;
}
```

Display the last committed configuration:

```
user@host> show configuration
## Last commit: 2018-08-10 11:21:58 PDT by user
version 8.3
protocols {
ospf {
    area 0.0.0.0 {
        interface so-0/0/0 {
            hello-interval 5;
        }
    }
}
}
```

## 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 "18.2R1 [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;
    }
}
...
```

## Displaying set Commands from the Junos OS Configuration

**IN THIS SECTION**


- Example: Displaying set Commands from the Configuration | **172**
- Example: Displaying set Commands with the match Option | **173**


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
```

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**.

This topic contains the following examples:

### 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;
    }
}
[edit interfaces fe-0/0/0]
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:

```
[edit interfaces fe-0/0/0]
```

```
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
```

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

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
```

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
```

# Verifying the 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 | 93
- Committing a Junos OS Configuration | 176


# Committing a Configuration

IN THIS SECTION


- Junos OS Commit Model for Configurations | 175
- Committing a Junos OS Configuration | 176
- Commit Operation When Multiple Users Configure the Software | 178
- Commit Preparation and Activation Overview | 179
- Committing Junos OS Configurations in Two Steps: Preparation and Activation | 180
- Activating a Junos OS Configuration but Requiring Confirmation | 182
- Scheduling a Junos OS Commit Operation | 183
- Monitoring the Junos OS Commit Process | 185

- Adding a Comment to Describe the Committed Configuration | **186**
- Junos OS Batch Commits Overview | **187**
- Example: Configuring Batch Commit Server Properties | **188**
- Backing Up the Committed Configuration on the Alternate Boot Drive | **198**


The **commit** configuration mode command enables you to save the Junos OS configuration changes to the configuration database and to activate the configuration on the device.

## Junos OS Commit Model for Configurations

The device 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 device 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 device can have a maximum of 49 rollback configurations (numbered 1 through 49) saved on the system.

On the device, the current 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. (The remaining rollback files, 4 through 49, are located in **/var/db/config**.)

If the recovery configuration 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 device:

- Synchronization: Propagates a configuration from one Routing Engine to a second Routing Engine within the same device chassis.

  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 device. 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 device remote from the device where the command was issued, the synchronization fails on the remote device. You need to clear the lock and reissue the **synchronization** command.

SEE ALSO

> *Configuring Junos OS for the First Time on a Device with a Single Routing Engine*

## Committing a Junos OS Configuration

To save Junos OS configuration changes to the configuration database and to activate the configuration on the device, 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 device configuration.

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

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/rundb/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/rundb 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.

**NOTE:**

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

## 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 are 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 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.

## Commit Preparation and Activation Overview

Beginning with Junos OS Release 17.3R1, you can complete the commit process in two steps. This feature enables you to configure several devices and simultaneously activate the configurations. Prior to Junos OS Release 17.3R1, the commit process was completed in a single step. The purpose of decoupling these stages of commit is to provide a definitive time window for the commit to be effective on the system. You can enter commit mode after the commit is prepared, but you will receive a message informing that the commit is pending activation.

In the first step, known as the preparation stage, the commit is validated and a new database with the necessary files is generated. If the configuration contains any syntax errors, an appropriate error message is displayed, and the configuration is not prepared. In the event of failure during the preparation stage, the error message **commit check-out failed** is displayed.

In the second step, referred to as the activation stage, the previously prepared configuration is activated. Next, if you need to clear the prepared configuration, you can do so by using **clear system commit prepared** command. A log message is generated upon successful clearing of the pending commit.

> **NOTE:** Commit operations cannot be performed in between preparation and activation stages.

The two-step commit process is superior to the single-step process for time-critical commits. In the single-step process, the preparation time can vary depending on the existing configuration on the device. In the two-step process, the complex preparation work is more efficiently handled.

Configuration commands are provided that allow you to prepare the configuration cache and activate the configuration. You can prepare the devices with new configurations and activate them at the exact times you want.

The **commit prepare** command validates the configurations, and the **commit activate** command activates the configurations. The commands have the following configuration options:

- **and-quit**
- **no-synchronize**
- **peers-synchronize**
- **synchronize**

The **commit prepare** and **commit activate** commands are available for private, exclusive and shared commits only. The commands are not applicable for dynamic and ephemeral modes. This feature is applicable for multichassis devices, but it is not applicable for batch commits.

To support this functionality using Network Configuration Protocol (NETCONF), the following new remote procedure calls (RPCs) are provided:

- **<commit-configuration>< prepare/></commit-configuration>**
- **<commit-configuration><activate/></commit-configuration>**
- **<clear-system-commit><prepared/></clear-system-commit>**

**NOTE:**

- In an MX Series Virtual Chassis setup the following applies: When **commit prepare** is issued on one Routing Engine followed by switchover, the Routing Engine where the switchover command is issued reboots. Therefore, the prepared cache is cleared in that Routing Engine.
- In an MX Series Virtual Chassis setup, it is advisable to execute **clear system commit prepared** command only on VC master.

## Committing Junos OS Configurations in Two Steps: Preparation and Activation

Beginning with Junos OS Release 17.3, you can complete the commit process in two steps. This enables you to configure several devices, and the configurations can be activated simultaneously. In the first step, known as the preparation stage, the commit is validated and a new database along with necessary files is generated. If the configuration contains any syntax errors, an appropriate error message is displayed, and the configuration is not prepared. In the second step, referred to as the activation stage, the previously prepared configuration is activated and becomes the current, operational device configuration.

To prepare the configuration:

1. At the **[edit]** hierarchy level in configuration mode, make the necessary changes to the configuration.

   For example, to configure the scripts of the system, issue the following command:

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

   For example:

   ```
   [edit]
   user@host#set system scripts language python
   ```

2. Issue the **commit prepare** command.

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

The message **commit prepare successful** is displayed.

If the preparation stage fails, the error message **commit check-out failed** is displayed.

```
[edit]
user@host# set interfaces ge-0/0/0 unit 0 family inet address 1.1.1.2/2
[edit]
user@host# set interfaces ge-0/0/1 unit 0 family inet address 1.1.1.2/24
[edit]
user@host# commit prepare
[edit interfaces ge-2/0/0 unit 0 family inet]
'address 1.1.1.2/24'
Cannot have the same local address on the same unit of an interface
error: configuration check-out failed
```

3. To verify the output of the **show system commit** command after **commit prepare** is issued, use the following command:

```
user@host> show system commit
commit prepared by user via cli is pending activation
```

To activate the prepared configuration:

1. Use the **commit activate** command

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

The message **commit complete** is displayed.

2. To verify the activated system configuration, use the following command:

```
user@host> show configuration system scripts
language python;
```

To verify the output of the **show system commit** and **show system commit revision detail** commands after **commit activate** is issued, issue the following commands.

```
user@host> show system commit
0 2018-07-12 22:54:46 PDT by user via cli commit activate
```

```
user@host> show system commit revision detail
Revision: re0-1499925285-2214
User : user
Client : cli
Time : 2018-07-12 22:54:46 PDT
Comment : commit activate
```

## 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 device. 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
configuration check succeeds
```

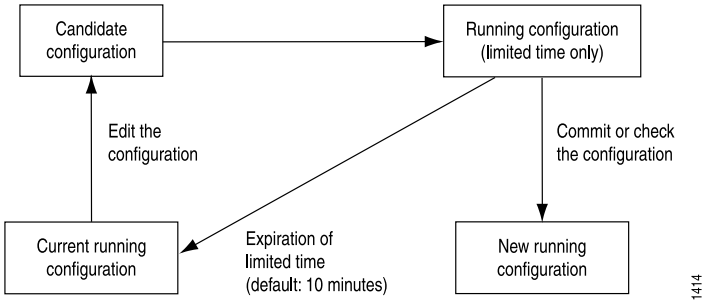
If the commit is not confirmed within a certain time (10 minutes by default), Junos OS automatically rolls back to the previous configuration and a broadcast message is sent to all logged-in users.

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 2018-01-05 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 temporarily (10 minutes by default) and begins running on the device.

**Figure 4: Confirm a Configuration**




To change the amount of time before you must 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#
```

You can also use the **commit confirmed** command in the **[edit private]** configuration mode.

## 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, **2018-08-21 12:30:00** is 12:30 PM on August 21, 2018. 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 "2018-03-10 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 device.

## 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
2018-09-22 15:39:39 PDT: exporting juniper.conf
2018-09-22 15:39:39 PDT: setup foreign files
2018-09-22 15:39:39 PDT: propagating foreign files
2018-09-22 15:39:39 PDT: complete foreign files
2018-09-22 15:39:40 PDT: copying configuration to juniper.data+
2018-09-22 15:39:40 PDT: dropping unchanged foreign files
2018-09-22 15:39:40 PDT: daemons checking new configuration
2018-09-22 15:39:41 PDT: commit wrapup...
2018-09-22 15:39:42 PDT: activating '/var/etc/ntp.conf'
2018-09-22 15:39:42 PDT: activating '/var/etc/kmd.conf'
2018-09-22 15:39:42 PDT: activating '/var/db/juniper.data'
2018-09-22 15:39:42 PDT: notifying daemons of new configuration
2018-09-22 15:39:42 PDT: signaling 'Firewall daemon', pid 24567, signal 1,
status 0
2018-09-22 15:39:42 PDT: signaling 'Interface daemon', pid 24568, signal 1,
status 0
2018-09-22 15:39:43 PDT: signaling 'Routing protocol daemon', pid 25679,
signal 1, status 0
2018-09-22 15:39:43 PDT: signaling 'MIB2 daemon', pid 24549, signal 1,
status 0
2018-09-22 15:39:43 PDT: signaling 'NTP daemon', pid 37863, signal 1, status 0
2018-09-22 15:39:43 PDT: signaling 'Sonet APS daemon', pid 24551, signal 1,
status 0
2018-09-22 15:39:43 PDT: signaling 'VRRP daemon', pid 24552, signal 1,
```

```
status 0
2018-09-22 15:39:43 PDT: signaling 'PFE daemon', pid 2316, signal 1, status 0
2018-09-22 15:39:43 PDT: signaling 'Traffic sampling control daemon', pid 24553
signal 1, status 0
2018-09-22 15:39:43 PDT: signaling 'IPsec Key Management daemon', pid
24556, signal 1, status 0
2018-09-22 15:39:43 PDT: signaling 'Forwarding UDP daemon', pid 2320,
signal 1, status 0
commit complete
```

## 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.

> **NOTE:** Beginning with Junos OS Release 11.4, you can also use the **commit confirmed** command in the **[edit private]** configuration mode.

## 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**.

## Example: Configuring Batch Commit Server Properties

IN THIS SECTION


- Requirements | 188
- Overview | 188
- Configuration | 188
- Verification | 191


This example shows how to configure batch commit server properties to manage batch commit operations.

### Requirements

This example uses the following hardware and software components:

- MX Series 5G Universal Routing Platform
- 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

**IN THIS SECTION**


- Checking the Batch Commit Server Status | **192**
- Checking the Batch Commit Status | **192**
- Viewing the Patch Files in a Batch Commit Job | **193**
- Viewing the Trace Files for Batch Commit Operations | **196**

Confirm that the configuration is working properly.

***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 2018

Completed commits:
```

```
  Id: 1000
  Last Modified: Tue Nov  1 22:46:43 2018
  Status: Successfully committed 1000

  Id: 1002
  Last Modified: Tue Nov  1 22:50:35 2018
  Status: Successfully committed 1002

  Id: 1004
  Last Modified: Tue Nov  1 22:51:48 2018
  Status: Successfully committed 1004

  Id: 1007
  Last Modified: Wed Nov  2 01:08:04 2018
  Status: Successfully committed 1007

  Id: 1009
  Last Modified: Wed Nov  2 01:16:45 2018
  Status: Successfully committed 1009

  Id: 1010
  Last Modified: Wed Nov  2 01:19:25 2018
  Status: Successfully committed 1010

  Id: 1011
  Last Modified: Wed Nov  2 01:28:16 2018
  Status: Successfully committed 1011


Error commits:
  Id: 1008
  Last Modified: Wed Nov  2 01:08:18 2018
  Status: Error while commiting 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. Use 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 2018
  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 2018
  Status: Successfully committed 1002

Patch:
[edit]
+ snmp {
+     community abc;
```

```
+ }
  Id: 1010
  Last Modified: Wed Nov  2 01:19:25 2018
  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 2018
  Status: Error while commiting 1008

Patch:
[edit system]
+ radius-server {
+     10.1.1.1 port 222;
+ }
```

The output shows the changes in configuration for each commit job ID.

2. 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 2018
  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**

- Use 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 commiting 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.

  The output shows batch commit job IDs for successful commit operations.

  `user@R0>` **file show/var/log/commitd_nov | match committed**

```

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.

  The output shows commit job IDs for failed commit operations.

  `user@R0>` **file show/var/log/commitd_nov | match "Error while"**

```
Nov  1 22:47:18 Error while commiting 1001
Nov  1 22:51:10 Error while commiting 1003
Nov  1 22:52:15 Error while commiting 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.

  The output shows commit server event logs.

  `user@R0>` **file show/var/log/commitd_nov | match "commit server"**

```
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
...
```

## 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).

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

Configure Command Overview | 85

4

CHAPTER

# Managing Configurations

Configuration Files Overview | 200

Managing Configurations | 202

Autoinstallation of Configuration Files | 223

Loading Configuration Files | 228

Backing Up Configurations to an Archive Site | 242

Factory Default Configuration | 244

Rescue Configuration | 245

Encrypting and Decrypting Configuration Files | 247

Synchronizing Configurations Across Routing Engines | 250

# Configuration Files Overview

IN THIS SECTION


- Understanding Configuration Files | 200
- Understanding How the Junos OS Configuration Is Stored | 201


A configuration file stores the complete configuration of a device. The active (running) configuration is the operational file of the device. The candidate configuration is the working copy storing configuration updates.

## Understanding Configuration Files

A configuration file stores the complete configuration of a network device. The current configuration of a device is called the active configuration. You can alter this current configuration and you can also return to a previous configuration or to a rescue configuration.

Juniper Networks Junos OS saves the 50 most recently committed configuration files on a device so that you can return to a previous configuration. The configuration files are named:

- **juniper.conf.gz**—The current active configuration.
- **juniper.conf.1.gz** to **juniper.conf.49.gz**—Rollback configurations.

To make changes to the configuration file, you must use configuration mode in the CLI. When making changes to a configuration file, you are viewing and changing the candidate configuration file. The candidate configuration allows you to make configuration changes without causing operational changes to the active configuration or causing potential damage to your current network operations. Once you commit the changes made to the candidate configuration, the system updates the active configuration.

### Configuration File Terms

**Table 10: Configuration File Terms**

| Term | Definition |
|---|---|
| active configuration | Current committed configuration of a device. |

**Table 10: Configuration File Terms *(continued)***

| Term | Definition |
|---|---|
| candidate configuration | Working copy of the configuration that allows users to make configurational changes without causing any operational changes until this copy is committed. |
| configuration group | Group of configuration statements that can be inherited by the rest of the configuration. |
| commit a configuration | Check configuration for proper syntax, activate and mark as the current configuration file running on the device. |
| configuration hierarchy | Junos OS configuration consists of a hierarchy of statements. There are two types of statements: Container statements, which contain other statements, and leaf statements, which do not contain other statements. All the container and leaf statements together form the configuration hierarchy. |
| default configuration | Default configuration contains the initial values set for each configuration parameter when a device is shipped. |
| rescue configuration | Well-known configuration that recovers a device from a configuration that denies management access. You set a current committed configuration to be the rescue configuration through the CLI. |
| roll back a configuration | Return to a previously committed configuration. |

SEE ALSO

Uploading a Configuration File | **240**

*Reverting to the Rescue Configuration*

*Uploading a Configuration File (CLI Procedure)*

*Reverting to the Rescue Configuration for the EX Series Switch*

## 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 stored on the device'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.

SEE ALSO

*Using Junos OS to Specify the Number of Configurations Stored on the CompactFlash Card*

Returning to the Most Recently Committed Junos OS Configuration | 212

Returning to a Previously Committed Junos OS Configuration | 213

Loading a Configuration from a File or the Terminal | 229

# Managing Configurations

**IN THIS SECTION**

- Understanding the show | compare | display xml Command Output | 203
- Returning to the Most Recently Committed Junos OS Configuration | 212
- Returning to a Previously Committed Junos OS Configuration | 213
- Saving a Configuration to a File | 217
- Compressing the Current Configuration File | 218
- Freeing Up System Storage Space | 220
- Cleaning Up Files with the CLI | 221

# Understanding the show | compare | display xml Command Output

IN THIS SECTION


- Adding a Statement (create Operation) | **204**
- Deleting a Statement (delete Operation) | **205**
- Changing a Statement (delete and create Operations) | **207**
- Changing Metadata (inactive Attribute and Operation) | **208**
- Adding an Annotation (comment Tag and create Operation) | **209**
- Changing an Annotation (comment Tag, and delete and create Operations) | **210**
- Adding a Statement Inside a Container (create Operation, and insert and key Attributes) | **211**
- Changing the Order Inside a Container (merge Operation, and insert and key Attributes) | **211**

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 is 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.

> **NOTE:** Beginning with Junos OS Release 16.2R2, the **show | compare | display xml** command omits the **<configuration>** tag in the XML output if the comparison returns no differences or if the comparison returns only differences for non-native configuration data, for example, configuration data associated with an OpenConfig data model.

The following sections explain the XML that is generated for specific types of configuration changes. The corresponding text changes are shown for comparison.

### 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>
```

SEE ALSO

- Pipe ( | ) Filter Functions in the Junos OS Command-Line Interface | 284
- Using Regular Expressions with the Pipe ( | ) Symbol to Filter Junos OS Command Output | 282
- Using the Pipe ( | ) Symbol to Filter Junos OS Command Output | 281

## 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
```

SEE ALSO

Rolling Back Junos OS Configuration Changes | 45
Understanding How the Junos OS Configuration Is Stored | 201

## Returning to a Previously Committed Junos OS Configuration

IN THIS SECTION

- Returning to a Configuration Prior to the One Most Recently Committed | 213
- Displaying Previous Configurations | 214
- Comparing Configuration Changes with a Prior Version | 215

This topic explains how you can return to a configuration prior to the most recently committed one.

### 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              2018-02-27 12:52:10 PST by abc via cli
1              2018-02-26 14:47:42 PST by def via cli
2              2018-02-14 21:55:45 PST by ghi via cli
3              2018-02-10 16:11:30 PST by jkl via cli
4              2018-02-10 16:02:35 PST by mno via cli
5              2018-03-16 15:10:41 PST by pqr via cli
6              2018-03-16 14:54:21 PST by stu via cli
7              2018-03-16 14:51:38 PST by vwx via cli
8              2018-03-16 14:43:29 PST by yzz via cli
9              2018-03-16 14:15:37 PST by abc via cli
10             2018-03-16 14:13:57 PST by def via cli
11             2018-03-16 12:57:19 PST by root via other
12             2018-03-16 10:45:23 PST by root via other
13             2018-03-16 10:08:13 PST by root via other
14             2018-03-16 01:20:56 PST by root via other
15             2018-03-16 00:40:37 PST by ghi via cli
16             2018-03-16 00:39:29 PST by jkl via cli
17             2018-03-16 00:32:36 PST by mno via cli
18             2018-03-16 00:31:17 PST by pqr via cli
19             2018-03-15 19:59:00 PST by stu via cli
20             2018-03-15 19:53:39 PST by vwx via cli
21             2018-03-15 18:07:19 PST by yzz via cli
22             2018-03-15 17:59:03 PST by abc via cli
23             2018-03-15 15:05:14 PST by def via cli
24             2018-03-15 15:04:51 PST by ghi via cli
25             2018-03-15 15:03:42 PST by jkl via cli
26             2018-03-15 15:01:52 PST by mno via cli
27             2018-03-15 14:58:34 PST by pqr via cli
28             2018-03-15 13:09:37 PST by root via other
```

```
29              2018-03-12 11:01:20 PST by stu via cli
30              2018-03-12 10:57:35 PST by vwx via cli
31              2018-03-11 10:25:07 PST by yzz via cli
32              2018-03-10 23:40:58 PST by abc via cli
33              2018-03-10 23:40:38 PST by def via cli
34              2018-03-10 23:14:27 PST by ghi via cli
35              2018-03-10 23:10:16 PST by jkl via cli
36              2018-03-10 23:01:51 PST by mno via cli
37              2018-03-10 22:49:57 PST by pqr via cli
38              2018-03-10 22:24:07 PST by stu via cli
39              2018-03-10 22:20:14 PST by vwx via cli
40              2018-03-10 22:16:56 PST by yzz via cli
41              2018-03-10 22:16:41 PST by abc via cli
42              2018-03-10 20:44:00 PST by def via cli
43              2018-03-10 20:43:29 PST by ghi via cli
44              2018-03-10 20:39:14 PST by jkl via cli
45              2018-03-10 20:31:30 PST by root via other
46              2018-03-10 18:57:01 PST by mno via cli
47              2018-03-10 18:56:18 PST by pqr via cli
48              2018-03-10 18:47:49 PST by stu via cli
49              2018-03-10 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;
}
```

SEE ALSO

Loading a Configuration from a File or the Terminal | 229

Viewing Files and Directories on a Device Running Junos OS | 268

## Saving a Configuration to a File

Saving Junos OS configuration to a file allows you to edit it with any plain 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;
        ...
    }
}
```

## 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
```

- 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# set no-compress-configuration-files
user@host# commit
commit complete
```

SEE ALSO

Junos OS Commit Model for Configurations | 175

*compress-configuration-files*

## Freeing Up System Storage Space

**Problem**

**Description:** The system file storage space on the device is full. Rebooting the switch does not solve the problem.

The following error message is displayed during a typical operation on the device after the file storage space is full.

```
user@host% cli
user@host> configure
/var: write failed, filesystem is full
```

**Solution**

Clean up the file storage on the device by deleting system files.

1. Request to delete system files.

   ```
   user@host> request system storage cleanup
   ```

   The list of files to be deleted is displayed.

   ```
   List of files to delete:

            Size Date         Name
       11B Jul 26 20:55 /var/jail/tmp/alarmd.ts
      124B Aug  4 18:05 /var/log/default-log-messages.0.gz
     1301B Jul 26 20:42 /var/log/install.0.gz
      387B Jun  3 14:37 /var/log/install.1.gz
     4920B Aug  4 18:05 /var/log/messages.0.gz
     20.0K Jul 26 21:00 /var/log/messages.1.gz
     16.3K Jun 25 13:45 /var/log/messages.2.gz
      804B Aug  4 18:05 /var/log/security.0.gz
     16.8K Aug  3 11:15 /var/log/security.1.gz
      487B Aug  4 18:04 /var/log/wtmp.0.gz
      855B Jul 29 22:54 /var/log/wtmp.1.gz
      920B Jun 30 16:32 /var/log/wtmp.2.gz
       94B Jun  3 14:36 /var/log/wtmp.3.gz
    353.2K Jun  3 14:37 /var/sw/pkg/jloader-qfx-11.2I20110303_1117_dc-builder.tgz
    124.0K Jun  3 14:30 /var/tmp/gres-tp/env.dat
        0B Apr 14 16:20 /var/tmp/gres-tp/lock
        0B Apr 14 17:37 /var/tmp/if-rtsdb/env.lck
     12.0K Jul 26 20:55 /var/tmp/if-rtsdb/env.mem
   ```

```
2688.0K Jul 26 20:55 /var/tmp/if-rtsdb/shm_usr1.mem
 132.0K Jul 26 20:55 /var/tmp/if-rtsdb/shm_usr2.mem
2048.0K Jul 26 20:55 /var/tmp/if-rtsdb/trace.mem
   155B Jul 26 20:55 /var/tmp/krt_gencfg_filter.txt
     0B Jul 26 20:55 /var/tmp/rtsdb/if-rtsdb
1400.6K Aug  3 10:13 /var/tmp/sfid.core.0.gz
1398.9K Aug  3 17:01 /var/tmp/sfid.core.1.gz
Delete these files ? [yes,no] (no)
```

2. Enter **yes** to delete the files.

3. Reboot the device.

**BEST PRACTICE:** We recommend that you regularly request a system file storage cleanup to optimize the performance of the device.

SEE ALSO

*request system storage cleanup*

## Cleaning Up Files with the CLI

You can use the CLI **request system storage cleanup** command to rotate log files and delete unnecessary files on the device. If you are running low on storage space, the file cleanup procedure quickly identifies files that can be deleted.

The file cleanup procedure performs the following tasks:

- Rotates log files—Archives all information in the current log files, deletes old archives, and creates fresh log files.
- Deletes log files in **/var/log**—Deletes any files that are not currently being written to.
- Deletes temporary files in **/var/tmp**—Deletes any files that have not been accessed within two days.
- Deletes all crash files in **/var/crash**—Deletes any core files that the device has written during an error.
- Deletes all software images (***.tgz** files) in **/var/sw/pkg**—Deletes any software images copied to this directory during software upgrades.

To rotate log files and delete unnecessary files with the CLI:

1. Enter operational mode in the CLI.

2. Rotate log files and identify the files that can be safely deleted.

   ```
   user@host> request system storage cleanup
   ```

   The device rotates log files and displays the files that you can delete.

3. Enter **yes** at the prompt to delete the files.

> **NOTE:** You can issue the **request system storage cleanup dry-run** command to review the list of files that can be deleted with the **request system storage cleanup** command, before actually deleting the files.

> **NOTE:**
> On SRX Series devices, the **/var** hierarchy is hosted in a separate partition (instead of the root partition). If Junos OS installation fails as a result of insufficient space:
>
> - Use the **request system storage cleanup** command to delete temporary files.
> - Delete any user-created files in both the root partition and under the **/var** hierarchy.

SEE ALSO

*Cleaning Up Files in J-Web*

Decrypting Configuration Files | 249

Encrypting Configuration Files | 247

*Managing Accounting Files*

# Autoinstallation of Configuration Files

IN THIS SECTION


- Understanding Autoinstallation of Configuration Files | 223
- Configuring Autoinstallation of Configuration Files (CLI Procedure) | 226


Autoinstallation is the automatic configuration of devices over the network without manual intervention, or without any need for any configuration.

## Understanding Autoinstallation of Configuration Files

IN THIS SECTION


- Typical Uses for Autoinstallation | 224
- Autoinstallation Configuration Files and IP Addresses | 224
- Typical Autoinstallation Process on a New Device | 225

Autoinstallation is the automatic configuration of a device over the network from a preexisting configuration file that you create and store on a configuration server—typically a Trivial File Transfer Protocol (TFTP) server. You can use autoinstallation to configure new devices automatically and to deploy multiple devices from a central location in the network.

You enable autoinstallation so that the switches in your network implement autoinstallation when they are powered on. To configure autoinstallation, you specify a configuration server, an autoinstallation interface, and a protocol for IP address acquisition.

> **NOTE:** The QFX5200 switches only work with HTTP for autoinstallation. TFTP and FTP protocols are not supported. Autoinstallation as a feature is not supported on all devices. Refer to your hardware information for specific details.

### Typical Uses for Autoinstallation

Typical uses for autoinstallation of the software include:

- To deploy and update multiple devices from a central location in the network.
- To update a device—Autoinstallation occurs when a device that has been manually configured for autoinstallation is powered on.

### Autoinstallation Configuration Files and IP Addresses

For the autoinstallation process to work, you must store one or more host-specific or default configuration files on a configuration server in the network and have a service available—typically Dynamic Host Configuration Protocol (DHCP)—to assign an IP address to the switch.

You can set up the following configuration files for autoinstallation on the device:

- **network.conf**—Default configuration file for autoinstallation, in which you specify IP addresses and associated hostnames for devices on the network.
- **switch.conf**—Default configuration file for autoinstallation with a minimum configuration sufficient for you to telnet to the device and configure it manually.
- ***hostname*.conf**—Host-specific configuration file for autoinstallation on a device that contains all the configuration information necessary for the device. In the filename, *hostname* is replaced with the hostname assigned to the device.

If the server with the autoinstallation configuration file is not on the same LAN segment as the new device, or if a specific device is required by the network, you must configure an intermediate device directly attached to the new device, through which the new device can send TFTP, Boot Protocol (BOOTP), and

Domain Name System (DNS) requests. In this case, you specify the IP address of the intermediate device as the location to receive TFTP requests for autoinstallation.

### Typical Autoinstallation Process on a New Device

When the device configured for autoinstallation is powered on, it performs the following autoinstallation tasks:

1. The device sends out DHCP or BOOTP requests on each connected interface simultaneously to obtain an IP address.

   If a DHCP server responds to these requests, it provides the device with some or all of the following information:

   - An IP address and subnet mask for the autoinstallation interface.
   - The location of the (typically) TFTP server, Hypertext Transfer Protocol (HTTP) server, or FTP server on which the configuration file is stored.
   - The name of the configuration file to be requested from the TFTP server.
   - The IP address or hostname of the TFTP server.

     If the DHCP server provides the server's hostname, a DNS server must be available on the network to resolve the name to an IP address.
   - The IP address of an intermediate device if the configuration server is on a different LAN segment from the device.

2. After the device acquires an IP address, the autoinstallation process on the device attempts to download a configuration file in the following ways:

   a. If the DHCP server specifies the host-specific configuration file ***hostname*.conf**, the device uses that filename in the TFTP server request. The autoinstallation process on the new device makes three unicast TFTP requests for ***hostname*.conf**. If these attempts fail, the device broadcasts three requests to any available TFTP server for the file.

   b. If the device does not locate a ***hostname*.conf** file, the autoinstallation process sends three unicast TFTP requests for a **network.conf** file that contains the device's hostname-to-IP-address mapping information. If these attempts fail, the device broadcasts three requests to any available TFTP server for the file.

   c. If the device fails to find a **network.conf** file that contains a hostname entry for the device, the autoinstallation process sends out a DNS request and attempts to resolve the device's IP address to a hostname.

d. If the device determines its hostname, it sends a TFTP request for the ***hostname*.conf** file.

e. If the device is unable to map its IP address to a hostname, it sends TFTP requests for the default configuration file **device.conf**. The TFTP request procedure is the same as for the **network.conf** file.

3. After the device locates a configuration file on a TFTP server, the autoinstallation process downloads the file, installs the file on the device, and commits the configuration.

SEE ALSO

> *Connecting and Configuring an EX Series Switch (CLI Procedure)*
>
> *Connecting and Configuring an EX Series Switch (J-Web Procedure)*
>
> *Configuration Files Terms*

## Configuring Autoinstallation of Configuration Files (CLI Procedure)

Autoinstallation is the automatic configuration of a device over the network from a pre-existing configuration file that you create and store on a configuration server—typically a Trivial File Transfer Protocol (TFTP) server. You can use autoinstallation to automatically deploy multiple devices from a central location in the network.

To specify autoinstallation to run when you power on a device already installed in your network, you can enable it by specifying one or more interfaces, protocols, and configuration servers to be used for autoinstallation.

Before you explicitly enable and configure autoinstallation on the device, perform these tasks as needed for your network's configuration:

- Have a service available—typically Dynamic Host Configuration Protocol (DHCP)—to assign an IP address to the device.
- Configure a DHCP server on your network to meet your network requirements. You can configure a switch to operate as a DHCP server.
- Create one of the following configuration files, and store it on a TFTP server (or HTTP server or FTP server) in the network:
  - A host-specific file with the name ***hostname*.conf** for each device undergoing autoinstallation. Replace ***hostname*** with the name of a device. The ***hostname*.conf** file typically contains all the configuration information necessary for the device with this hostname.

  - A default configuration file named **device.conf** with the minimum configuration necessary to enable you to telnet into the new device for further configuration.

- Physically attach the device to the network using a Gigabit Ethernet port.
- If you configure the DHCP server to provide only the TFTP server hostname, add an IP address-to-hostname mapping entry for the TFTP server to the DNS database file on the Domain Name System (DNS) server in the network.
- If the device is not on the same network segment as the DHCP server (or other device providing IP address resolution), configure an existing device as an intermediate device to receive TFTP and DNS requests and forward them to the TFTP server and the DNS server. You must configure the LAN or serial interface on the intermediate device with the IP addresses of the hosts providing TFTP and DNS services. Connect this interface to the device.
- If you are using ***hostname*.conf** files for autoinstallation, you must also complete the following tasks:
  - Configure the DHCP server to provide a ***hostname*.conf** filename to each device. Each device uses its ***hostname*.conf** filename to request a configuration file from the TFTP server. Copy the necessary ***hostname*.conf** configuration files to the TFTP server.
  - Create a default configuration file named **network.conf**, and copy it to the TFTP server. This file contains IP-address-to-hostname mapping entries. If the DHCP server does not send a ***hostname*.conf** filename to a new device, the device uses **network.conf** to resolve its hostname based on its IP address.

    Alternatively, you can add the IP-address-to-hostname mapping entry for the device to a DNS database file.

    The device uses the hostname to request a ***hostname*.conf** file from the TFTP server.

To configure autoinstallation:

1. Specify the URL address of one or more servers from which to obtain configuration files.

   ```
   [edit system]
   user@host# set autoinstallation configuration-servers tftp://tftpconfig.example.com
   ```

   > **NOTE:** You can also use an FTP address, for example, **ftp://*user*:*password*@sftpconfig.example.com**.

2. Configure one or more Ethernet interfaces to perform autoinstallation and one or two procurement protocols for each interface. The switch uses the protocols to send a request for an IP address for the interface:

   ```
   [edit system]
   user@host# set autoinstallation interfaces ge-0/0/0 bootp
   ```

To verify autoinstallation, from the CLI, enter the **show system autoinstallation status** command.

```
user@host> show system autoinstallation status
```

```
Autoinstallation status:
Master state: Active
Last committed file: None
Configuration server of last committed file: 10.25.100.1
Interface:
  Name: ge-0/0/0
  State: Configuration Acquisition
  Acquired:
    Address: 192.168.124.75
    Hostname: host-ge-000
    Hostname source: DNS
    Configuration filename: device-ge-000.conf
    Configuration filename server: 10.25.100.3
  Address acquisition:
    Protocol: DHCP Client
    Acquired address: None
    Protocol: RARP Client
    Acquired address: None
Interface:
  Name: ge-0/0/1
  State: None
  Address acquisition:
    Protocol: DHCP Client
    Acquired address: None
    Protocol: RARP Client
    Acquired address: None
```

# Loading Configuration Files

**IN THIS SECTION**


- Loading a Configuration from a File or the Terminal | **229**
- Understanding Character Encoding on Devices Running Junos OS | **232**
- Additional Details About Specifying Junos OS Statements and Identifiers | **234**

Loading a Configuration from a File | 238
Uploading a Configuration File | 240

Loading configuration files on the device are helpful for loading parts of configuration files that might be common across many devices within a network.

## 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 device, and then load the file into the CLI. After you have loaded the file, you can commit it to activate the configuration on the device, 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 device, use the **load** configuration mode command:

```
[edit]
user@host# load (factory-default | merge | override | patch | replace | set | update) filename <relative> <json>
```

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 device. 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
```

SEE ALSO

Viewing Files and Directories on a Device Running Junos OS | 268

## Understanding Character Encoding on Devices Running Junos OS

Junos OS configuration data and operational command output might contain non-ASCII characters, which are outside of the 7-bit ASCII character set. When displaying operational or configuration data in certain formats or within a certain type of session, Junos OS escapes and encodes these characters using the equivalent UTF-8 decimal character reference.

The Junos OS command-line interface (CLI) attempts to display any non-ASCII characters in configuration data that is emitted in text, set, or JSON format, and similarly attempts to display these characters in command output that is emitted in text format. In the exception cases, which include configuration data in XML format and command output in XML or JSON format, the Junos OS CLI displays the UTF-8 decimal character reference instead. In NETCONF and Junos XML protocol sessions, if you request configuration data or command output that contains non-ASCII characters, the server returns the equivalent UTF-8 decimal character reference for those characters for all formats.

For example, suppose the following user account, which contains the Latin small letter n with a tilde (ñ), is configured on the device running Junos OS.

```
[edit]
user@host# set system login user mariap class super-user uid 2007 full-name "Maria Peña"
```

When you display the resulting configuration in text format, the CLI prints the corresponding character.

```
[edit]
```

```
user@host# show system login user mariap
```

```
full-name "Maria Peña";
uid 2007;
class super-user;
```

When you display the resulting configuration in XML format in the CLI or display the configuration in any format in a NETCONF or Junos XML protocol session, the ñ character maps to its equivalent UTF-8 decimal character reference **Ã±**.

```
[edit]
user@host# show system login user mariap | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/17.2R1/junos">
    <configuration junos:changed-seconds="1494033077"
junos:changed-localtime="2017-05-05 18:11:17 PDT">
            <system>
                <login>
                    <user>
                        <name>mariap</name>
                        <full-name>Maria Pe&#195;&#177;a</full-name>
                        <uid>2007</uid>
                        <class>super-user</class>
                    </user>
                </login>
            </system>
    </configuration>
    <cli>
        <banner>[edit]</banner>
    </cli>
</rpc-reply>
```

When you load configuration data onto a device running Junos OS, you can load non-ASCII characters using their equivalent UTF-8 decimal character reference.

# Additional Details About Specifying Junos OS Statements and Identifiers

IN THIS SECTION


- Specifying Statements | 234
- Performing CLI Type Checking | 236


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

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 11 on page 236 lists the data types the CLI checks.

**Table 11: CLI Configuration Input Types**

| Data Type | Format | Examples |
|---|---|---|
| Physical interface name (used in the **[edit interfaces]** hierarchy) | ***type-fpc/pic/port*** | **Correct: so-0/0/1**<br>**Incorrect: so-0** |
| Full interface name | ***type-fpc/pic/port<:channel>.logical*** | **Correct: so-0/0/1.0**<br>**Incorrect: so-0/0/1** |
| Full or abbreviated interface name (used in places other than the **[edit interfaces]** hierarchy) | ***type-<fpc</pic/port>><<:channel>.logical>*** | **Correct: so, so-1, so-1/2/3:4.5** |

**Table 11: CLI Configuration Input Types *(continued)***

| Data Type | Format | Examples |
|---|---|---|
| IP address | **0x*hex-bytesoctet*<. *octet*<.*octet*. <*octet*>>>** | **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 | **0x*hex-bytes*</*length*>*octet*<*octet* <*octet*.<*octet*>>></*length*>** | **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 | ***hex-nibble*<*hex-nibble* ...>** | **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** |
| OSPF area identifier (ID) | **0x*hex-bytesoctet*<.*octet*<.*octet*.< *octet* >>> *decimal-number*** | **Correct: 54, 0.0.0.54, 0x01020304, 1.2.3.4**<br><br>**Sample translations:**<br><br>**54** becomes **0.0.0.54**<br><br>**257** becomes **0.0.1.1**<br>**128.8** becomes **128.8.0.0**<br>**0x010203** becomes **0.1.2.3** |

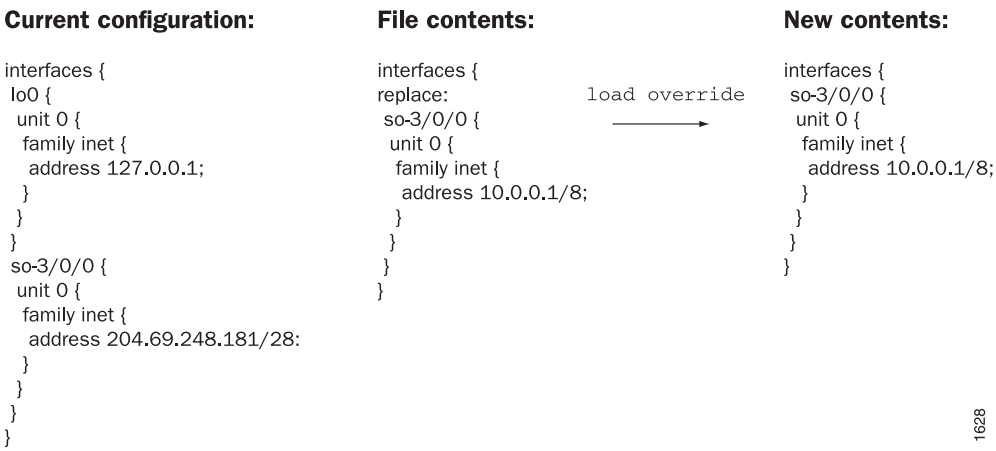
SEE ALSO

Entering and Exiting the Junos OS CLI Configuration Mode | 77

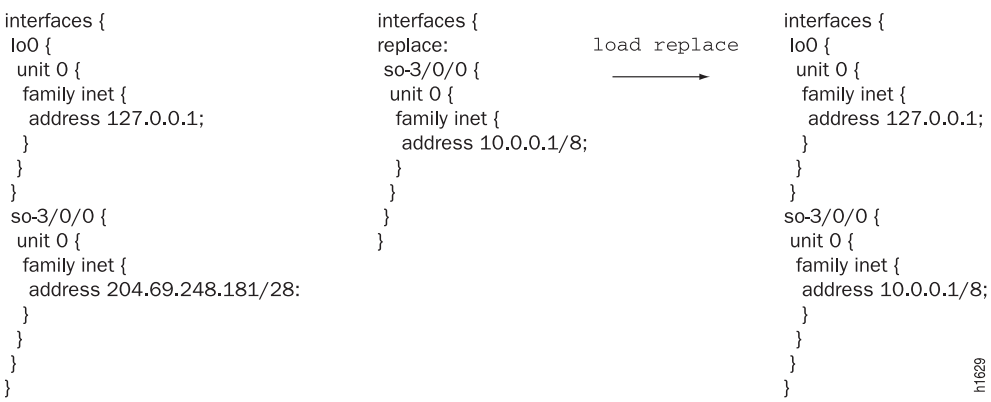
## Loading a Configuration from a File

The following examples demonstrate the process of loading a configuration from a file.

### Figure 5: Overriding the Current Configuration




### Figure 6: Using the replace Option



## Figure 7: Using the merge Option

**Current configuration:**

```
interfaces {
 lo0 {
  unit 0 {
   family inet {
    address 127.0.0.1;
   }
  }
 }
 so-3/0/0 {
  unit 0 {
   family inet {
    address 204.69.248.181/28;
   }
  }
 }
}
```

**File contents:**

```
interfaces {
replace:
 so-3/0/0 {
  unit 0 {
   family inet {
    address 10.0.0.1/8;
   }
  }
 }
}
```

load merge →

**New contents:**

```
interfaces {
 lo0 {
  unit 0 {
   family inet {
    address 127.0.0.1;
   }
  }
 }
 so-3/0/0 {
  unit 0 {
   family inet {
    address 10.0.0.1/8;
    address 204.69.248.181/28;
   }
  }
 }
}
```

1705

## Figure 8: 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 9: 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 "$9$Ydg4ZDjqf5FVw"; ## SECRET-DATA
      }
    }
  }
}
```

g017215

## Uploading a Configuration File

You can create a configuration file on your local system, copy the file to the device, and then load the file into the CLI. After you have loaded the configuration file, you can commit it to activate the configuration on the device. You can also edit the configuration interactively using the CLI and commit it at a later time.

To upload a configuration file from your local system:

1. Create the configuration file using a text editor such as Notepad, making sure that the syntax of the configuration file is correct.

2. In the configuration text file, use an option to perform the required action when the file is loaded.

**Table 12: Options for the load Command**

| Options | Description |
|---|---|
| **merge** | Combines the current active configuration and the configuration in the filename you specify or the one that you type in the terminal window. A **merge** operation is useful when you are adding a new section to an existing configuration. If the active configuration and the incoming configuration contain conflicting statements, the statements in the incoming configuration override those in the active configuration. |
| **override** | Discards the current candidate configuration and loads the configuration in the filename you specify or the one that you type at the terminal. When you use the **override** option and commit the configuration, all system processes reparse the configuration. You can use the **override** option at any level of the hierarchy. |
| **replace** | 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 the statements marked with the **replace** tag to the active configuration.<br><br>**NOTE:** For this operation to work, you must include **replace** tags in the text file or in the configuration you enter using the terminal. |

3. Press Ctrl+a to select all the text in the configuration file.

4. Press Ctrl+c to copy the contents of the configuration text file to the Clipboard.

5. Log in to the device using your username and password.

6. Enter configuration mode:
   user@host> **configure**

   [edit]
   user@host#

7. Load the configuration file:
   [edit]
   user@host# **load merge terminal**

8. At the cursor, paste the contents of the Clipboard using the mouse and the Paste icon:
   [edit]
   user@host# **load merge terminal**
   [Type ^D at a new line to end input]
   >Cursor is here. Paste the contents of the clipboard here<

9. Press Enter.

10. Press Ctrl+d to set the end-of-file marker.

To view results of the configuration steps before committing the configuration, type the **show** command at the user prompt.

To commit these changes to the active configuration, type the **commit** command at the user prompt. You can also edit the configuration interactively using the CLI and commit it at a later time.

SEE ALSO

Understanding Configuration Files | 200

RELATED DOCUMENTATION

Day One: Exploring the Junos CLI

# Backing Up Configurations to an Archive Site

You can configure a device to transfer its configuration to an archive file periodically.

1. Configuring the Transfer of the Currently Active Configuration | 242

## Configuring the Transfer of the Currently Active Configuration

If you want to back up your device's current configuration to an archive site, you can configure the device to transfer its currently active configuration by FTP, HTTP, or secure copy (SCP) periodically or after each commit.

To configure the device to transfer its currently active configuration to an archive site, include statements at the **[edit system archival configuration]** hierarchy level:

```
[edit system archival configuration]
archive-sites {
```

```
    ftp://username<:password>@host-address<:port>/url-path;
    scp://username<:password>@host-address<:port>/url-path;
    http://username @host-address :url-path <password>;
}
transfer-interval interval;
transfer-on-commit;
```

To configure the device to periodically transfer its currently active configuration to an archive site, include the **transfer-interval** statement at the **[edit system archival configuration]** hierarchy level:

```
[edit system archival configuration]
transfer-interval interval;
```

The ***interval*** is a period of time ranging from 15 through 2880 minutes.

To configure the device to transfer its currently active configuration to an archive site each time you commit a candidate configuration, include the **transfer-on-commit** statement at the **[edit system archival configuration]** hierarchy level:

```
[edit system archival configuration]
transfer-on-commit;
```

**NOTE:** When specifying a URL in a Junos OS statement using an IPv6 host address, you must enclose the entire URL in quotation marks ("") and enclose the IPv6 host address in brackets ([ ]). For example, **"ftp://*username*<:*password*>@[*ipv6-host-address*]<:*port*>/*url-path*"**

When you configure the device to transfer its configuration files, you specify an archive site to which the files are transferred. If you specify more than one archive site, the device attempts to transfer files to the first archive site in the list, moving to the next site only if the transfer fails.

When you use the **archive-sites** statement, you can specify a destination as an FTP URL, HTTP URL, or SCP-style remote file specification. The URL type **file://** is also supported.

To configure the archive site, include the **archive-sites** statement at the **[edit system archival configuration]** hierarchy level:

```
[edit system archival configuration]
archive-sites {
    ftp://username@host:<port>url-path password password;
    scp://username@host:<port>url-path password password;
```

```
    file://<path>/<filename>;
    http://username@host: url-path password password;
}
```

When you specify the archive site, do not add a forward slash (/) to the end of the URL.

The destination filename is saved in the following format, where *n* corresponds to the number of the compressed configuration rollback file that has been archived:

```
<router-name>_YYYYMMDD_HHMMSS_juniper.conf.n.gz
```

**NOTE:** Whenever configurations are made, the time included in the destination filename is in Coordinated Universal Time (UTC).

**NOTE:** When you configure file archival by using the **archive-sites** statement, the transfer file utility does not work if you have enabled the management instance.

# Factory Default Configuration

**IN THIS SECTION**


- Reverting to the Default Factory Configuration | 245


The default factory configuration which contains the basic configuration settings is the first configuration of the device and is loaded when the device is first installed and powered on.

## Reverting to the Default Factory Configuration

If for any reason the current active configuration fails, you can revert to the default factory configuration. The default factory configuration contains the basic configuration settings. This is the first configuration of the switch, and it is loaded when the switch is first installed and powered on.

The **load factory default** command is a standard Junos OS configuration command. This configuration command replaces the current active configuration with the default factory configuration.

To revert the switch to the rescue configuration:

```
[edit]
user@switch# load factory-default
[edit]
user@switch# delete system commit factory-settings
[edit]
user@switch# commit
```

**NOTE:** This process clears prior committed configuration parameters, except for those which preserve a Virtual Chassis configuration. This is how you can restore the factory default configuration on a Virtual Chassis (multiple devices configured to work together that look like a single device) without removing anything needed to keep the Virtual Chassis working.

SEE ALSO

Understanding Configuration Files | 200

*Reverting to the Rescue Configuration*

# Rescue Configuration

**IN THIS SECTION**

- Creating and Returning to a Rescue Configuration | 246

A rescue configuration is the known working configuration. If the active configuration is corrupted, the device automatically loads the rescue configuration file as the active configuration.

## 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 for recovery, if necessary. This alleviates the necessity of having to remember the rollback number with the **rollback** command. The rescue configuration rolls back the device to a known configuration, or can serve as a last resort if your device 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. To commit the rescue configuration, thereby activating it, use the **commit** 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 delete an existing rescue configuration, issue the **request system configuration rescue delete** command:

```
user@host> request system configuration rescue delete
user@host>
```

# Encrypting and Decrypting Configuration Files

IN THIS SECTION


- Encrypting Configuration Files | 247
- Decrypting Configuration Files | 249
- Modifying the Encryption Key | 250


Encrypting configuration file enables you to store configuration data or sensitive information in a configuration file. Decrypting is disabling the encryption of configuration files on a device and make them readable to all.

> **NOTE:** Encryption features are not available on all Juniper Networks devices. If so, the Junos OS CLI encryption-related commands described here may be hidden or not function. See your hardware documentation for details.

## Encrypting Configuration Files

To configure an encryption key in EEPROM and determine the encryption process, enter one of the **request system set-encryption-key** commands in operational mode described in Table 13 on page 247.

**Table 13: request system set-encryption-key Commands**

| CLI Command | Description |
|---|---|
| **request system set-encryption-key** | Sets the encryption key and enables default configuration file encryption:<br>• AES encryption for the Canada and U.S. version of Junos OS<br>• DES encryption for the international version of Junos OS |
| **request system set-encryption-key algorithm des** | Sets the encryption key and specifies configuration file encryption by DES. |

**Table 13: request system set-encryption-key Commands *(continued)***

| CLI Command | Description |
|---|---|
| **request system set-encryption-key unique** | Sets the encryption key and enables default configuration file encryption with a unique encryption key that includes the chassis serial number of the device.<br><br>Configuration files encrypted with the unique key can be decrypted only on the current device. You cannot copy such configuration files to another device and decrypt them. |
| **request system set-encryption-key des unique** | Sets the encryption key and specifies configuration file encryption by DES with a unique encryption key. |

To encrypt configuration files on a device:

1. Enter operational mode in the CLI.

2. Configure an encryption key in EEPROM and determine the encryption process; for example, enter the **request system set-encryption-key** command.

```
user@host> request system set-encryption-key
Enter EEPROM stored encryption key:
```

3. At the prompt, enter the encryption key. The encryption key must have at least six characters.

```
Enter EEPROM stored encryption key:juniper1
Verifying EEPROM stored encryption key:
```

4. At the second prompt, reenter the encryption key.

5. Enter configuration mode in the CLI.

6. Enable configuration file encryption to take place.

```
[edit]
user@host# edit system
user@host# set encrypt-configuration-files
```

7. Begin the encryption process by committing the configuration.

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

## Decrypting Configuration Files

To disable the encryption of configuration files on a device and make them readable to all:

1. Enter operational mode in the CLI.

2. Verify your permission to decrypt configuration files on this device by entering the encryption key for the device.

```
user@host> request system set-encryption-key
Enter EEPROM stored encryption key:
Verifying EEPROM stored encryption key:
```

3. At the second prompt, reenter the encryption key.

4. Enter configuration mode in the CLI.

5. Enable configuration file decryption.

```
[edit]
user@host# edit system
user@host# set no-encrypt-configuration-files
```

6. Begin the decryption process by committing the configuration.

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

## Modifying the Encryption Key

When you modify the encryption key, the configuration files are decrypted and then reencrypted with the new encryption key.

To modify the encryption key:

1. Enter operational mode in the CLI.

2. Configure a new encryption key in EEPROM and determine the encryption process; for example, enter the **request system set-encryption-key** command.

```
user@host> request system set-encryption-key
Enter EEPROM stored encryption key:
```

3. At the prompt, enter the new encryption key. The encryption key must have at least six characters.

```
Enter EEPROM stored encryption key:juniperone
Verifying EEPROM stored encryption key:
```

4. At the second prompt, reenter the new encryption key.

RELATED DOCUMENTATION

*Managing Accounting Files*

# Synchronizing Configurations Across Routing Engines

IN THIS SECTION


- Synchronizing Routing Engines | 251
- Configuring Multiple Routing Engines to Synchronize Committed Configurations Automatically | 254

On devices with redundant Routing Engines, you can perform a **commit synchronize**, which activates and synchronizes the configuration on both Routing Engines.

## Synchronizing Routing Engines

If your device 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) first commits the configuration, then copies and loads its candidate configuration to the other (responding Routing Engine). Each Routing Engine will perform a syntax check on the candidate configuration file before committing it, and the commit synchronization process takes place one Routing Engine at a time.

If no errors are found, the configuration is activated and becomes the current operational configuration on both Routing Engines.

> **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.

> **NOTE:** If the configuration has a large text size or many apply-groups, commit times can be longer than desired.

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 a Junos OS Configuration Group" on page 137.

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:** If the backup Routing Engine is partially committed due to invalid configuration during system reboot, the **commit synchronize** command with the **force** option from the master Routing Engine does not work.

The **commit synchronize** command does not work if the responding Routing Engine has uncommitted configuration changes. However, you can force 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.

To force a **commit synchronize** on the Routing Engines, log in to the Routing Engine from which you want to synchronize and issue the 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 device, 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.

**NOTE:** Starting in Junos OS Evolved Release 19.4R1, **commit synchronize** is enabled by default on PTX10008. If you issue **commit** at the **[edit system]** hierarchy level from the master routing engine, you see that the backup routing engine is automatically synchronized.

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.
- If **fast-synchronize** is enabled and if the master and backup Routing Engines run different software versions, even if the master validates the configuration, you cannot be sure if the configuration is valid for the backup Routing Engine. Therefore, 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
```

**NOTE:** 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;
```

SEE ALSO

commit | 335

Applying a Junos OS Configuration Group | 137

*Junos OS Administration Library*

## Configuring Multiple Routing Engines to Synchronize Committed Configurations Automatically

If your device 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.

> **NOTE:** Starting in Junos OS Evolved Release 19.4R1-S1, **commit synchronize** is enabled by default on PTX10008. If you issue **commit** at the **[edit system]** hierarchy level from the master routing engine, you see that the backup routing engine is automatically synchronized.

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.
- If **fast-synchronize** is enabled and if the master and backup Routing Engines run different software versions, even if the master validates the configuration, you cannot be sure if the configuration is valid for the backup Routing Engine. Therefore, ensure that the Junos OS software version running on both the Routing Engines is same.

## RELATED DOCUMENTATION

*Configuring Junos OS for the First Time on a Device with Dual Routing Engines*

*Configuring the Junos OS to Support Redundancy on Routers Having Multiple Routing Engines or Switching Boards*

Junos OS Commit Model for Configurations | **175**

*Junos OS Routing Engine Components and Processes*

CHAPTER 5

# Using Operational Commands to Monitor Devices

CLI Operational Mode Overview | 258

Using Operational Commands to Monitor a Device | 264

Filtering Operational Command Output | 280

# CLI Operational Mode Overview

IN THIS SECTION


- Overview of Junos OS CLI Operational Mode Commands | 258
- Understanding the Display Options of the show Command | 260
- Interface Naming Conventions Used in the Junos OS Operational Commands | 262
- Using Wildcard Characters in Interface Names | 264


In the operational mode, you can use Junos OS CLI commands to monitor and troubleshoot the device. The **monitor**, **ping**, **show**, **test**, and **traceroute** commands let you display information and test network connectivity for the device.

## Overview of Junos OS CLI Operational Mode Commands

IN THIS SECTION


- CLI Command Categories | 258
- Commonly Used Operational Mode Commands | 259


This topic provides an overview of Junos OS CLI operational mode commands.

### CLI Command Categories

There are several broad categories of CLI commands, including:

- Operational mode commands for monitoring and troubleshooting—The following commands perform functions related to information and statistics about the software and to test network connectivity.
  - **clear**—Clear statistics and protocol database information.
  - **file**—Perform file operations.
  - **mtrace**—Trace a multicast path 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 restarting software processes—The commands in the **restart** hierarchy restart the various Junos OS processes, including the routing protocol, interface, and SNMP.
- A command—**request**—for performing system-level operations, including stopping and rebooting the router or switch and loading Junos OS images.
- 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 more information about the CLI operational mode commands, see the CLI Explorer. Alternatively, you can also enter **?** at the operational mode command prompt to view a list of available commands.

SEE ALSO

> CLI Explorer
>
> Understanding Junos OS CLI Configuration Mode | **71**
>
> Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies | **26**

### Commonly Used Operational Mode Commands

The following table 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.

**Table 14: Commonly Used Operational Mode Commands**

| Items to Check | Description | Command |
|---|---|---|
| Software version | Versions of software running on the router or switch | **show version** |

**Table 14: Commonly Used Operational Mode Commands *(continued)***

| Items to Check | Description | Command |
|---|---|---|
| Log files | Contents of the log files | **monitor** |
| | Log files and their contents and recent user logins | **show log** |
| Remote systems | Host reachability and network connectivity | **ping** |
| | Route to a network system | **traceroute** |
| Configuration | Current system configuration | **show configuration** |
| Manipulate files | List of files and directories on the router or switch | **file list** |
| | Contents of a file | **file show** |
| Interface information | Detailed information about interfaces | **show interfaces** |
| Chassis | Chassis alarm status | **show chassis alarms** |
| | Information currently on craft display | **show chassis craft-interface** |
| | Router or switch environment information | **show chassis environment** |
| | Hardware inventory | **show chassis hardware** |
| Routing table information | Information about entries in the routing tables | **show route** |
| Forwarding table information | Information about data in the kernel's forwarding table | **show route forwarding-table** |

## Understanding the Display Options of the show Command

The Junos OS **show** command can include **brief**, **detail**, **extensive**, or **terse** options. You can use these—as well as other options—to control the amount and type of information you want to view.

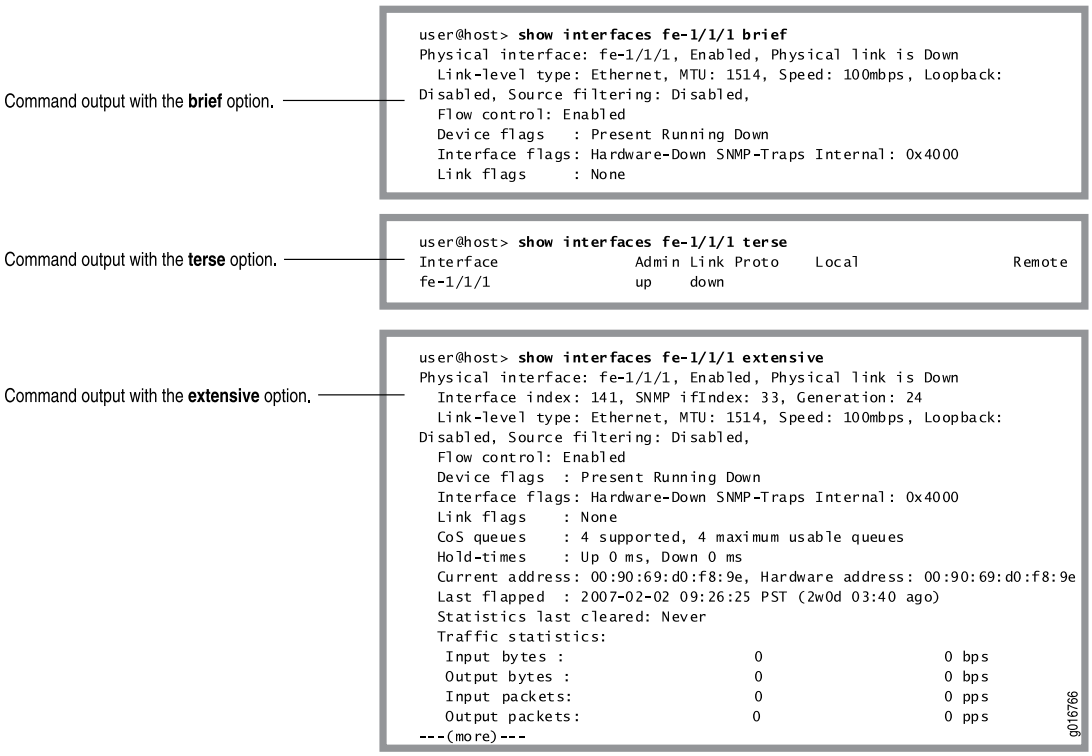
1. At any point in the CLI, you can enter the **?** character to view all the currently available options. For example:

   ```
   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.

**Figure 10: Command Output Options**




SEE ALSO

Controlling the Scope of an Operational Mode Command | 266

# Interface Naming Conventions Used in the Junos OS Operational Commands

IN THIS SECTION


- Physical Part of an Interface Name | 262
- Logical Part of an Interface Name | 262
- Channel Identifier Part of an Interface Name | 263


This topic explains the interface naming conventions used in the Junos OS operational commands.

## 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 **ae*number*** and **as*number***, 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. Logical unit numbers are used to uniquely identify physical or virtual storage

systems within a network. 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.

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
```

SEE ALSO

Example: Configuring Interfaces Using Configuration Groups | 148

*Junos OS Network Interfaces Library for Routing Devices*

## 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. The following table lists the available wildcard characters. You must enclose all wildcard characters except the asterisk (*) in quotation marks (“ ”).

**Table 15: Wildcard Characters for Specifying Interface Names**

| Wildcard Character | Description |
|---|---|
| * **(asterisk)** | Match any string of characters in that position in the interface name. For example, **so*** matches all SONET/SDH interfaces. |
| **"[*character<character...>*]"** | Match one or more individual characters in that position in the interface name. For example, **so-“[03]”*** matches all SONET/SDH interfaces in slots 0 and 3. |
| **"[!*character<character...>*]"** | Match all characters except the ones included in the brackets. For example, **so-“[!03]”*** matches all SONET/SDH interfaces except those in slots 0 and 3. |
| **"[*character1-character2*]"** | Match a range of characters. For example, **so-“[0-3]”** * 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, **so-”[!0-3]”*** matches all SONET/SDH interfaces in slots 4, 5, 6, and 7. |

SEE ALSO

Using Keyboard Sequences to Move Around and Edit the Junos OS CLI | **36**

Using Global Replace in the Junos OS Configuration | **117**

# Using Operational Commands to Monitor a Device

**IN THIS SECTION**

Using the Junos OS CLI Command Completion | **265**

Controlling the Scope of an Operational Mode Command | **266**

- Viewing Files and Directories on a Device Running Junos OS | 268
- Displaying Junos OS Information | 274
- Managing Programs and Processes Using Junos OS Operational Mode Commands | 274
- Using the Junos OS CLI Comment Character # for Operational Mode Commands | 280


Operational mode CLI commands enable you to monitor and control the operation of a device running the Junos OS. The operational mode commands exist in a hierarchical structure.

## Using the Junos OS CLI Command Completion

The command completion feature in Junos OS can help make it easier both to enter commands or to learn what possible completion options are available at any given time.

Issue the **show interfaces** command. In this case, the spacebar is used to autocomplete the 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. Here, the Tab key is used to perform the autocompletion.

```
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
...
```

## 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, mpls:
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)---
```

**NOTE:** This example output shows only one interface, for the sake of brevity, but in reality, the interfaces information for all four would be shown after the **—(more)—** prompts.

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>
```

### Routing Matrix Command Options

The **show version** command has several 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
```

SEE ALSO

Interface Naming Conventions Used in the Junos OS Operational Commands | 262

## Viewing Files and Directories on a Device Running Junos OS

IN THIS SECTION


- Directories on the Device | 269
- Listing Files and Directories | 269
- Specifying Filenames and URLs | 272


Junos OS stores information in files on the device, including configuration files, log files, and device 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 Device

The following table lists some standard directories on a device running Junos OS.

**Table 16: Directories on the Device**

| DIrectory | Description |
|---|---|
| **/config** | This directory is located on the device's router's internal flash drive. It contains the active configuration (**juniper.conf)** and rollback files 1, 2, and 3. |
| **/var/db/config** | This directory is located on the router'sdevice's hard drive and contains rollback files 4 through 49. |
| **/var/tmp** | This directory is located on thedevice'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. |
| **/var/log** | 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 **traceoptions** command. |
| **/var/home** | 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. |
| **/altroot** | 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. |
| **/altconfig** | This directory is located on the device's hard drive and contains a copy of the **/config** 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.

1. To get help about the **file** command, enter **file ?**:

```
user@host> file ?
Possible completions:

  <[Enter]>            Execute this command
```

```
  archive              Archives files from the system
  change-owner         Change owner of file
  change-permission    Change permission of file
  checksum             Calculate file checksum
  compress             Compress file
  copy                 Copy files (local or remote)
  delete               Delete files from the system
  delete-directory     Delete a directory
  link                 Create link between files
  list                 List file information
  make-directory       Create a new
  make-directory       Create a new
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 2017
  /altconfig/          Last changed: Aug 07 2017
```

```
  /altroot/            Last changed: Aug 07 2017
  /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 2017
  /dev/                Last changed: Apr 09 22:36:21
  /etc/                Last changed: Apr 11 03:14:22
  /kernel              Size: 27823246, Last changed: Aug 07 2017
  /mfs/                Last changed: Apr 09 22:36:49
  /mnt/                Last changed: Jan 11 2017
  /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 2017
  /var/at/             Last changed: Jul 09 2017
  /var/backups/        Last changed: Jul 09 2017
  /var/bin/            Last changed: Jul 09 2017
  /var/crash/          Last changed: Apr 09 22:31:08
  /var/cron/           Last changed: Jul 09 2017
  /var/db/             Last changed: May 07 20:28:40
  /var/empty/          Last changed: Jul 09 2017
  /var/etc/            Last changed: Apr 16 22:35:36
  /var/heimdal/        Last changed: Jul 10 2017
  /var/home/           Last changed: Apr 09 22:59:18
  /var/jail/           Last changed: Oct 31 2017
  /var/log/            Last changed: Apr 17 02:00:10
  /var/mail/           Last changed: Jul 09 2017
  /var/msgs/           Last changed: Jul 09 2017
  /var/named/          Last changed: Jul 10 2017
  /var/packages/       Last changed: Jan 18 02:38:59
  /var/pdb/            Last changed: Oct 31 2017
  /var/preserve/       Last changed: Jul 09 2017
  /var/run/            Last changed: Apr 17 02:00:01
```

```
  /var/rundb/          Last changed: Apr 17 00:46:00
  /var/rwho/           Last changed: Jul 09 2017
  /var/sdb/            Last changed: Apr 09 22:37:31
  /var/spool/          Last changed: Jul 09 2017
  /var/sw/             Last changed: Jul 09 2017
  /var/tmp/            Last changed: Apr 09 23:28:41
  /var/transfer/       Last changed: Jul 09 2017
  /var/yp/             Last changed: Jul 09 2017
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 2017-06-12 07:58:27
 UTC
Jul  9 23:18:05 CHASSISD release 8.4I0 built by builder on 2017-06-12 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*/%2F*path/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
  ```

## 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 specific 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.

   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.

## Managing Programs and Processes Using Junos OS Operational Mode Commands

**IN THIS SECTION**


- Showing Software Processes | **275**
- Restarting the Junos OS Process | **276**
- Stopping Junos OS | **278**
- Rebooting Junos OS | **279**

This topic shows some examples of Junos operational commands that you can use to manage programs and processes on a device running Junos OS.

### 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. Enter the **show system processes extensive** command. This command shows the CPU utilization on the device and lists the processes in order of CPU utilization.

The following table lists and describes the output fields included in this example. The fields are listed in alphabetical order.

**Table 17: show system process extensive Command Output Fields**

| Field | Description |
|---|---|
| **COMMAND** | Command that is running. |
| **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 (**sleeping, running, starting, zombies,** and **stopped**). |
| **RES** | Current amount of resident memory, in KB. |
| **SIZE** | Total size of the process (**text, data,** and **stack**), in KB. |
| **STATE** | Current state of the process (**sleep**, **wait**, **run**, **idle**, **zombi**, or **stop**). |
| **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.
   - The option **gracefully** restarts the software process after performing clean-up tasks.
   - The option **immediately** restarts the software process without performing any clean-up tasks.
   - The option **soft** rereads and reactivates the configuration without completely restarting the software processes. For example, BGP peers stay up and the routing table stays constant.

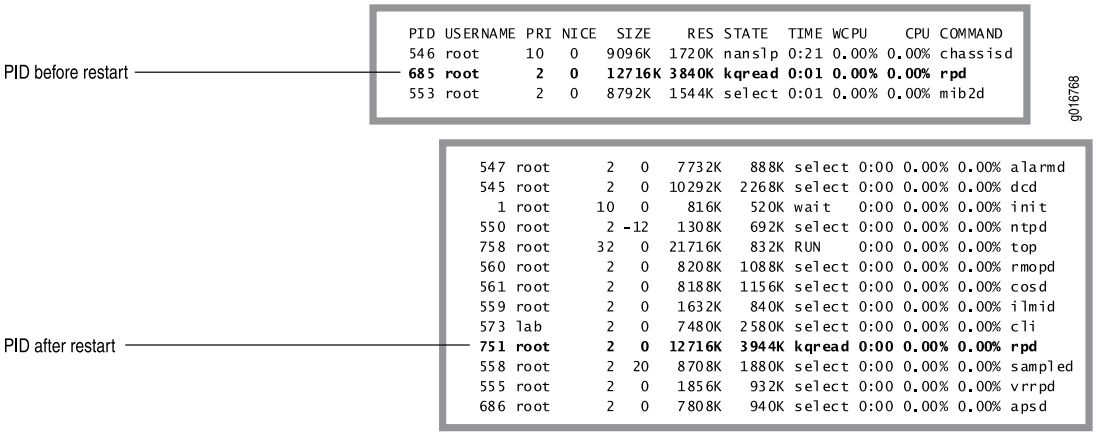
**NOTE:** The **gracefully**, **immediately**, and **soft** options for the **restart** command are optional and not required for executing the command.

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 identifer (PID) is updated.

**Figure 11: Restarting a Process**




## Stopping Junos OS

**CAUTION:** To avoid possible damage to the file system and to prevent loss of data, you must always gracefully shut down Junos OS before powering off the device.

Stopping Junos OS on a device should always be done through a direct console connection, not through the network. As Junos OS shuts down, the network will go down, and if you were connected that way, you will not see the results output. 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. As with the **shutdown** command, rebooting Junos OS should be done through a direct console connection, otherwise you will not see the command output when the network goes down during the reboot process. 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...
```

SEE ALSO

Checking the Status of a Device Running Junos OS | 42

### 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.

> **NOTE:** Comments entered in the CLI, whether individually or by pasting the contents of a configuration file, are not actually saved in the device configuration, but instead are ignored.

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 #.

# Filtering Operational Command Output

**IN THIS SECTION**


- Using the Pipe ( | ) Symbol to Filter Junos OS Command Output | **281**
- Using Regular Expressions with the Pipe ( | ) Symbol to Filter Junos OS Command Output | **282**
- Pipe ( | ) Filter Functions in the Junos OS Command-Line Interface | **284**
- Filtering Operational Mode Command Output in a QFabric System | **298**


The pipe **|** symbol lets you filter the command output in both operational and configuration modes.

## 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*.

## 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 18: 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:

```
user@host> show chassis hardware
Hardware inventory:
Item Version Part number Serial number Description
Chassis F0632 MX80
Midplane REV 09 711-031594 ZW0568 MX80
PEM 0 Rev 04 740-028288 VK09886 AC Power Entry Module
Routing Engine BUILTIN BUILTIN Routing Engine
TFEB 0 BUILTIN BUILTIN Forwarding Engine Processor
QXM 0 REV 06 711-028408 ZW4288 MPC QXM
FPC 0 BUILTIN BUILTIN MPC BUILTIN
MIC 0 BUILTIN BUILTIN 4x 10GE XFP
PIC 0 BUILTIN BUILTIN 4x 10GE XFP
Xcvr 0 REV 02 740-014289 C825XU010 XFP-10G-SR
Xcvr 1 REV 03 740-014289 CB25BQ0WD XFP-10G-SR
Xcvr 2 REV 01 740-011571 C739XJ039 XFP-10G-SR
FPC 1 BUILTIN BUILTIN MPC BUILTIN
MIC 1 *** Hardware Not Supported ***
Fan Tray Fan Tray
```

A pipe filter of **| match "FPC 1"** displays the following output:

```
FPC 1 BUILTIN BUILTIN MPC BUILTIN
```

A pipe filter of **| except "FPC 1"** displays the following output:

```
Hardware inventory:
Item Version Part number Serial number Description
Chassis F0632 MX80
PEM 0 Rev 04 740-028288 VK09886 AC Power Entry Module
Routing Engine BUILTIN BUILTIN Routing Engine
TFEB 0 BUILTIN BUILTIN Forwarding Engine Processor
FPC 0 BUILTIN BUILTIN MPC BUILTIN
Fan Tray Fan Tray
```

# Pipe ( | ) Filter Functions in the Junos OS Command-Line Interface

IN THIS SECTION


- Comparing Configurations and Displaying the Differences in Text | **285**
- Comparing Configurations and Displaying the Differences in XML | **286**
- Counting the Number of Lines of Output | **287**
- Displaying Output in XML Tag Format | **287**
- Displaying Static Configuration Data | **288**
- Displaying Ephemeral Configuration Data | **289**
- Displaying Output in JSON Format | **289**
- Displaying the Configuration with YANG Translation Scripts Applied | **290**
- Displaying the RPC Tags for a Command | **292**
- Ignoring Output That Does Not Match a Regular Expression | **293**
- Displaying Output from the First Match of a Regular Expression | **293**
- Retaining Output After the Last Screen | **294**
- Displaying Output Beginning with the Last Entries | **294**
- Displaying Output That Matches a Regular Expression | **295**
- Preventing Output from Being Paginated | **295**
- Sending Command Output to Other Users | **295**
- Resolving IP Addresses | **296**
- Saving Output to a File | **296**
- Appending Output to a File | **297**
- Displaying Output on Screen and Writing to a File | **297**
- Trimming Output by Specifying the Starting Column | **298**
- Refreshing the Output of a Command | **298**


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

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, as follows:

```
filename| compare 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.

> **NOTE:** Starting in Junos OS Release 16.2R2, the **show | compare | display xml** command omits the **<configuration>** tag in the XML output if the comparison returns no differences or if the comparison returns only differences for non-native configuration data, for example, configuration data associated with an OpenConfig data model.

SEE ALSO

Understanding the show | compare | display xml Command Output | 203

### 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>
```

If the configuration data or command output contains characters that are outside of the 7-bit ASCII character set, the Junos OS CLI displays the equivalent UTF-8 decimal character reference for those characters in the XML output.

SEE ALSO


- Comparing Configurations and Displaying the Differences in XML | 286
- Understanding Character Encoding on Devices Running Junos OS | 232


### Displaying Static Configuration Data

To view the inherited configuration data and information about the source group from which the configuration has been inherited with respect to the static configuration database, issue the **show configuration | display inheritance** command.

```
user@host> show configuration | display inheritance
```

```
## Last commit: 2018-03-29 15:54:17 PDT
version 16.2R2;
system {
...
}
```

### Displaying Ephemeral Configuration Data

Juniper Extension Toolkit (JET) applications and NETCONF and Junos XML protocol client applications can configure the ephemeral configuration database. The ephemeral database is an alternate configuration database that provides a fast programmatic interface for performing configuration updates.

To view the complete post-inheritance configuration merged with the configuration data in all instances of the ephemeral database, use the **show ephemeral-configuration merge** command.

```
user@host> show ephemeral-configuration merge
```

```
## Last changed: 2019-02-01 09:47:20 PST
version 18.2R1;
system {
...
}
```

> **NOTE:** In Junos OS Release 18.1 and earlier, to view the complete post-inheritance configuration merged with the configuration data in all instances of the ephemeral configuration database, use the **show ephemeral-configuration | display merge** command. Starting in Junos OS Release 18.2R1, the **display merge** option is deprecated.

### 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"
        }
        ]
```

```
    }
    ]
}
```

If the operational command output contains characters that are outside of the 7-bit ASCII character set, the Junos OS CLI displays the equivalent UTF-8 decimal character reference for those characters in the JSON output.

**NOTE:**

- Starting in Junos OS Release 16.1, devices running Junos OS emit JSON-formatted configuration data using a new default implementation for serialization.
- Starting in Junos OS Releases 16.1R4, 16.2R2, and 17.1R1, integers in Junos OS configuration data emitted in JSON format are not enclosed in quotation marks. In earlier releases, integers in JSON configuration data were treated as strings and enclosed in quotation marks.
- Starting in Junos OS Release 17.3R1, OpenConfig supports the operational state emitted by daemons directly in JSON format in addition to XML format. To configure JSON compact format, use the command **set system export-format state-data json compact**. This command converts XML format to compact JSON format. Else, it emits the JSON in non-compact format.

SEE ALSO

Understanding Character Encoding on Devices Running Junos OS | 232

### 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-inhertitance 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;
        gigether-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.

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
```

SEE ALSO

Using Regular Expressions with the Pipe ( | ) Symbol to Filter Junos OS Command Output | 282

### 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.

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 lo0.0
```

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               rjct     3     1
0                  user     0               recv     5     2
1                  user     0               recv     5     2
32769              user     0               ucst    45     1 fe-0/0/0.534
fe-0/0/0. (CCC)    user     0               indr    44     2
                              10.0.16.2     Push 32768, Push
```

SEE ALSO

Using Regular Expressions with the Pipe ( | ) Symbol to Filter Junos OS Command Output | 282

### 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.

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 {
```

SEE ALSO

Using Regular Expressions with the Pipe ( | ) Symbol to Filter Junos OS Command Output | 282

### 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 Space.

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 email.

### 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@ttyp1**:

```
user@host> show interfaces | request message user root@ttyp1
```

The user **root@ttyp1** sees the following output appear on the terminal screen:

```
Message from user@host on /dev/ttyp0 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’
```

### 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'
```

### 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 604,800 seconds. For example, to refresh the output of the **show interfaces** command every 5 seconds, you would run the following command:

```
user@host> show interfaces | refresh 5
```

## 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
```

CHAPTER 6

# Junos OS Configuration Statements


apply-groups | 301

apply-groups-except | 302

archival | 303

autoinstallation | 305

commit activate | 306

commit prepare | 308

export-format | 310

groups | 312

no-hidden-commands | 315

server (Batch Commits) | 316

synchronize | 318

traceoptions (Batch Commits) | 321

# apply-groups

**Syntax**

```
apply-groups [ group-names ];
```

**Hierarchy Level**

All hierarchy levels

**Release Information**

Statement introduced before Junos OS Release 7.4.

**Description**

Apply a configuration group to a specific hierarchy level in a configuration, to have a configuration inherit the statements in the configuration group.

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.

**Options**

***group-names***—One or more names specified in the **groups** statement.

**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

Applying a Junos OS Configuration Group | 137

groups | 312

# apply-groups-except

**Syntax**

```
apply-groups-except [ group-names ];
```

**Hierarchy Level**

All hierarchy levels except the top level

**Release Information**

Statement introduced before Junos OS Release 7.4.

**Description**

Disable inheritance of a configuration group.

**Options**

***group-names***—One or more names specified in the **groups** statement.

**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

groups | 312

Disabling Inheritance of a Junos OS Configuration Group | 139

# archival

### Syntax

```
archival {
    configuration {
        archive-sites {
            file://<path>/<filename>;
            ftp://username@host:<port>url-path password password;
            http://username@host:<port>url-path password password;
            pasvftp://username@host:<port>url-path password password;
            scp://username@host:<port>url-path password password;
        }
        transfer-interval interval;
        transfer-on-commit;
    }
    routing-instance routing-instance;
}
```

### Hierarchy Level

```
[edit system]
```

### Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Statement introduced in Junos OS Release 11.1 for the QFX Series.
Statement introduced in Junos OS Release 14.1X53-D20 for OCX Series switches.

### Description

Configure copying of the currently active configuration to an archive site. An archive site can be a file, or an FTP, HTTP, passive FTP, or SCP location.

### Options

**configuration**—Configure the router or switch to periodically transfer its currently active configuration (or after each commit). Parameters include **archive-sites**, **transfer-interval**, and **transfer-on-commit**.

**NOTE:** The **[edit system archival]** hierarchy is not available on QFabric systems.

**archive-sites**—Specify where to transfer the current configuration files. When specifying a URL in a Junos OS statement using an IPv6 host address, you must enclose the entire URL in quotation marks (" ") and enclose the IPv6 host address in brackets ([ ]). For example: **"scp://*username*<:*password*>@[*ipv6-host-address*]<:*port*>/*url-path*"**.

If you specify more than one archive site, the router or switch attempts to transfer the configuration files to the first archive site in the list, moving to the next only if the transfer fails. The destination filename is saved in the following format, where *n* corresponds to the number of the compressed configuration rollback file that has been archived:

```
router-name_YYYYMMDD_HHMMSS_juniper.conf.n.gz
```

**NOTE:** The time included in the destination filename is always in Coordinated Universal Time (UTC) regardless of whether the time on the router or switch is configured as UTC or the local time zone. The default time zone on the router or switch is UTC.

**transfer-interval**—The frequency, in minutes, for transferring the current configuration to an archive site. Valid intervals are 15 to 2880 minutes.

**transfer-on-commit**—Configure the router or switch to transfer its currently active configuration to an archive site each time you commit a candidate configuration.

**routing-instance**—Defines the routing instance through which a server is reachable.

**Required Privilege Level**
admin—To view this statement in the configuration.
admin-control—To add this statement to the configuration.

## RELATED DOCUMENTATION

Backing Up Configurations to an Archive Site | 242

# autoinstallation

**Syntax**

```
autoinstallation;
```

**Hierarchy Level**

```
[edit system]
```

**Release Information**
Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.1 for EX Series switches.
Statement introduced in Junos OS Release 12.2 for ACX Series Universal Access Routers.

**Description**

Download a configuration file automatically from an FTP, Hypertext Transfer Protocol (HTTP), or Trivial FTP (TFTP) server. When you power on a router or switch configured for autoinstallation, it requests an IP address from a Dynamic Host Configuration Protocol (DHCP) server. Once the router or switch has an address, it sends a request to a configuration server and downloads and installs a configuration.

**Options**

There are no options.

**Required Privilege Level**
system—To view this statement in the configuration.
system-control—To add this statement to the configuration.

## RELATED DOCUMENTATION

- *ACX Series Autoinstallation Overview*
- *Before You Begin Autoinstallation on an ACX Series Universal Metro Router*
- *Autoinstallation Configuration of ACX Series Universal Metro Routers*
- *USB Autoinstallation on ACX Series Routers*
- *Verifying Autoinstallation on ACX Series Universal Metro Routers*
- *show system autoinstallation status*
- *Upgrading Software by Using Automatic Software Download for Switches*

# commit activate

**Syntax**

```
commit activate{
    comment;
    and-quit;
    peers-synchronize;
    synchronize;
}
```

**Hierarchy Level**

```
[edit system]
```

**Release Information**
Statement introduced in Junos OS Release 17.3R1.

**Description**
Activate a previously prepared commit. Upon successful validation, during the activation stage, previously prepared commits are activated. Also, pending activation files are checked during this stage. If there are pending activation files, the existence of required files and daemon map present in the database data structures are checked. If there is any failure, a log message is generated that informs you that the commit has failed.

**Options**

**and-quit**—(Optional) Commit the configuration and, if the configuration contains no errors and the commit succeeds, exit from configuration mode.

**no-synchronize**—(Optional) Do not synchronize the commit. Configure the **commit prepare** statement to run without synchronization.

**peers-synchronize**—(Optional) Synchronize the commit on remote peers.

**synchronize**—(Optional) Synchronize the commit on both Routing Engines.

**Required Privilege Level**
configure—To enter configuration mode.
system—To view this statement in the configuration.
system-control—To add this statement to the configuration.

RELATED DOCUMENTATION

Commit Preparation and Activation Overview | 179

Committing Junos OS Configurations in Two Steps: Preparation and Activation | 180

# commit prepare

**Syntax**

```
commit prepare{
    and-quit;
    no-synchronize;
    peers-synchronzie;
synchronize;
    }
```

**Hierarchy Level**

```
[edit system]
```

**Release Information**

Statement introduced in Junos OS Release 17.3.

**Description**

Prepare for an upcoming commit activation. Prepare the configurations that can be activated at a later stage. During the preparation stage, all the required files and databases are generated and the configuration is validated. A file is created that indicates if the commit is pending for activation. In the event of failure during the preparation stage, the log message **commit preparation failed** is generated.

**Options**

**and-quit**—(Optional) Commit the configuration and, if the configuration contains no errors and the commit succeeds, exit from configuration mode.

**no-synchronize**—(Optional) Do not synchronize the commit. Configure the **commit prepare** statement to run without synchronization.

**peers-synchronize**—(Optional) Synchronize the commit on remote peers.

**synchronize**—(Optional) Synchronize the commit on both Routing Engines.

**Required Privilege Level**

configure—To enter configuration mode.
system—To view this statement in the configuration.
system-control—To add this statement to the configuration.

RELATED DOCUMENTATION

Commit Preparation and Activation Overview | 179

Committing Junos OS Configurations in Two Steps: Preparation and Activation | 180

# export-format

**Syntax**

```
export-format {
    json {
        ietf;
    }
}
```

**Hierarchy Level**

```
[edit system]
```

**Release Information**

Statement introduced in Junos OS Release 16.1.

**Description**

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.

**Options**

**json**—Define which implementation of the serialization to use for configuration data emitted in JavaScript Object Notation (JSON) format.

Acceptable values include:

- **ietf**—JSON data is emitted according to the encoding rules defined in Internet drafts draft-ietf-netmod-yang-json-09, *JSON Encoding of Data Modeled with YANG*, and draft-ietf-netmod-yang-metadata-06, *Defining and Using Metadata with YANG*.

**Default:** ietf

> **NOTE:** Starting in Junos OS Release 17.3R1, OpenConfig supports the operational state emitted by daemons directly in JSON format in addition to XML format. To configure JSON compact format, use the following command: **set system export-format state-data json compact**.
>
> This CLI command converts XML format to compact JSON format. Else, it emits the JSON in non-compact format.

**Required Privilege Level**

maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

## RELATED DOCUMENTATION

- *Mapping Junos OS Command Output to JSON in the CLI*
- *Mapping Junos OS Configuration Statements to JSON*

# 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.

> **NOTE:** Junos OS does not support configuring statements corresponding to third-party YANG data models, for example, OpenConfig or custom data models, under the **[edit groups]** hierarchy.

### 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**—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.

- **chassis** *chassis-id*—Specify the chassis type of the router. Valid types include SCC0, SCC1, LCC0, LCC1 ... LCC3.
- **member** *member-id*—Specify the name of the member of the virtual chassis.
- **model** *model-id*—Specify the model name of the router, such as m7i or tx100.
- **node** *node-id*—Specify the cluster node.
- **peers** *names-of-peers*—Specify the names of the MC-LAG peers participating in commit synchronization.
- **routing-engine** *routing-engine-id*—Specify the type of Routing Engine, re0 or re1.
- **time** *start-time* **[to** *end-time***]**—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 uses the format yyyy-mm-dd.hh:mm, hh:mm, or hh.

***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.

On routers that support multiple Routing Engines, you can also specify two special group names:

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**.

- **lcc*n*-re0**—Configuration statements applied to the Routing Engine in slot 0 of the specified T640 router that is connected to a TX Matrix router.
- **lcc*n*-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.

**Required Privilege Level**
configure—To enter configuration mode.

RELATED DOCUMENTATION

Creating a Junos OS Configuration Group | **135**
apply-groups | **301**
apply-groups-except | **302**

# no-hidden-commands

**Syntax**

```
no-hidden-commands;
```

**Hierarchy Level**

```
[edit system]
```

**Release Information**
Statement introduced in Junos OS Release 16.1R1.

**Description**
Hidden commands are Junos OS commands that are not published but could be run on a router. Hidden commands serve a specific purpose, but for most part are not expected to be used, and as such are not actively supported. The **no-hidden-commands** statement allows you to block all hidden commands to all users except the root users.

**Default**
Hidden commands are enabled by default.

**Options**
This command has no options.

**Required Privilege Level**
admin—To view this statement in the configuration.
admin-control—To add this statement to the configuration.

# server (Batch Commits)

**Syntax**

```
server {
    commit-interval number-of-seconds-between-commits;
    commit-schedule-profile;
    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;
    redirect-completion-status;
    retry-attempts;
    retry-interval;
    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 in seconds between commits.

**days-to-keep-error-logs**—Configure the number of days to keep log entries. Valid range is from 1 to 366 days.

**maximum-aggregate-pool**—Configure the maximum number of commits to aggregate together. The valid range is 1 through 4294967295.

**maximum-entries**—Configure the maximum number of commit entries.

**redirect-completion-status**—Configure the redirect asynchronous commit status to server configured here.

**retry-attempts**—Configure the retry attempts for commit failure due to db lock error. The default is 5 retries.

**retry-interval**—Configure the retry interval in seconds for commit failure. The default is 20 seconds.

The remaining statements are explained separately. See CLI Explorer.

**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 | **188**

traceoptions | **321**

# synchronize

### Syntax

```
synchronize;
```

### Hierarchy Level

```
[edit system commit]
```

### Release Information

Statement introduced in Junos OS Release 7.4.
Statement introduced in Junos OS Release 10.4 for EX Series switches.

### Description

For devices with multiple Routing Engines only. Configure the **commit** command to automatically perform a **commit synchronize** action between dual Routing Engines within the same chassis. The Routing Engine on which you execute the **commit** command (the requesting Routing Engine) copies and loads its candidate configuration to the other (the responding) Routing Engine. Each Routing Engine then performs 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.

**NOTE:** If you configure the **commit synchronize** statement at the **[edit system]** hierarchy level and issue a **commit** in the master Routing Engine, the master configuration is automatically synchronized with the backup. However, if the backup Routing Engine is down when you issue the **commit**, the Junos OS displays a warning and commits the candidate configuration in the master Routing Engine. When the backup Routing Engine comes up, its configuration will automatically be synchronized with the master. A newly inserted backup Routing Engine automatically synchronizes its configuration with the master Routing Engine configuration.

**NOTE:** When you configure nonstop active routing (NSR), you must configure the **commit synchronize** statement. Otherwise, the commit operation fails.

> **NOTE:** Starting in Junos OS Release 20.2R1, when the **commit synchronize** statement is configured and the backup Routing Engine synchronizes its configuration with the master Routing Engine, for example, when it is newly inserted, brought back online, or during a change in mastership, it also synchronizes the ephemeral configuration database.

On the TX Matrix router, synchronization only occurs between the Routing Engines within the same chassis. When synchronization is complete, the new configuration is then distributed to the Routing Engines on the T640 routers. That is, the master Routing Engine on the TX Matrix router distributes the configuration to the master Routing Engine on each T640 router. Likewise, the backup Routing Engine on the TX Matrix router distributes the configuration to the backup Routing Engine on each T640 router.

On the TX Matrix Plus router, synchronization only occurs between the Routing Engines within the switch-fabric chassis and when synchronization is complete, the new configuration is then distributed to the Routing Engines on the line-card chassis (LCC). That is, the master Routing Engine on the TX Matrix Plus router distributes the configuration to the master Routing Engine on each LCC. Likewise, the backup Routing Engine on the TX Matrix Plus router distributes the configuration to the backup Routing Engine on each LCC.

In EX Series Virtual Chassis configurations:

- On EX4200 switches in Virtual Chassis, synchronization occurs between the switch in the master role and the switch in the backup role.
- On EX8200 switches in a Virtual Chassis, synchronization occurs only between the master and backup XRE200 External Routing Engines.

**Options**

**and-quit**—(Optional) Quit configuration mode if the commit synchronization succeeds.

**at**—(Optional) Time at which to activate configuration changes.

**comment**—(Optional) Write a message to the commit log.

**force**—(Optional) Force a commit synchronization on the other Routing Engine (ignore warnings).

**scripts**—(Optional) Push scripts to the other Routing Engine.

**Required Privilege Level**

system—To view this statement in the configuration.
system-control—To add this statement to the configuration.

RELATED DOCUMENTATION

*Synchronizing the Routing Engine Configuration*

Configuring Multiple Routing Engines to Synchronize Committed Configurations Automatically | **254**

# traceoptions (Batch Commits)

**Syntax**

```
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 server],
[edit system commit synchronize server]
```

**Release Information**

Statement introduced in Junos OS Release 12.1.

**Description**

For Junos OS batch commits, configure tracing operations.

**Options**

**file *name***—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.

**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 | **188**

CHAPTER 7

# Junos OS CLI Commands


activate | 326

annotate | 327

clear log | 329

clear system commit | 331

clear system commit prepared | 333

commit | 335

configure | 341

copy | 343

deactivate | 344

delete | 345

edit | 347

exit | 348

file | 349

help | 350

insert | 351

load | 352

| (pipe) | 355

protect | 359

quit | 360

rename | 361

replace | 363

request | 364

request system commit server pause | 366

request system commit server queue cleanup | 367

request system commit server start | 369

request system configuration rescue delete | 370

request system configuration rescue save | 371

restart | 372

rollback | 385

run | 386

save | 387

set | 389

set cli complete-on-space | 390

set cli directory | 392

set cli idle-timeout | 393

set cli prompt | 395

set cli restart-on-upgrade | 396

set cli screen-length | 397

set cli screen-width | 398

set cli terminal | 399

set cli timestamp | 400

set date | 402

show | 404

show cli | 405

show cli authorization | 407

show cli directory | 411

show cli history | 412

show configuration | 413

show | display inheritance | 417

show | display omit | 420

show | display set | 421

show | display set relative | 423

show groups junos-defaults | 425

show system commit | 427

show system commit server queue | 431

show system commit server status | 436

show system configuration archival | 438

show system configuration rescue | 439

show system rollback | 441

status | 443

test configuration | 444

top | 446

unprotect | 447

up | 448

update | 449

wildcard delete | 450

# activate

**Syntax**

```
activate <statement | identifier >
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Remove the **inactive:** 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 **commit** command.

**Options**

***identifier***—Identifier from which you are removing the inactive tag. It must be an identifier at the current hierarchy level.

***statement***—Statement from which you are removing the inactive tag. It must be a statement at the current hierarchy level.

**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

deactivate | 344

Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration | 114

# 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

***statement***—Statement to which you are attaching the comment.

***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.

### 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 | 127

# clear log

**Syntax**

```
clear log filename
<all>
```

**Release Information**

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 OCX Series switches.

**Description**

Remove contents of a log file.

**Options**

***filename***—Name of the specific log file to delete. Note that the file name cannot contain any special characters, including: ![=;|(){}]

**all**—(Optional) Delete the specified log file and all archived versions of it.

**Required Privilege Level**

clear

RELATED DOCUMENTATION

*show log*

**List of Sample Output**

clear log on page 329

**Output Fields**

See *file list* for an explanation of output fields.

## Sample Output

**clear log**

The following sample commands list log file information, clear the contents of a log file, and then display the updated log file information:

```
user@host> file list lcc0-re0:/var/log/sampled detail
```

```
lcc0-re0:
--------------------------------------------------------------------------
-rw-r-----  1 root  wheel      26450 Jun 23 18:47 /var/log/sampled
total 1
```

```
user@host> clear log lcc0-re0:sampled
```

```
lcc0-re0:
--------------------------------------------------------------------------
```

```
user@host> file list lcc0-re0:/var/log/sampled detail
```

```
lcc0-re0:
--------------------------------------------------------------------------
-rw-r-----  1 root  wheel         57 Sep 15 03:44 /var/log/sampled
total 1
```

# clear system commit

**Syntax**

```
clear system commit
<synchronize-server pending-jobs>
```

**Release Information**

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.
Option **synchronize-server** introduced in Junos OS Release 17.2R1.

**Description**

Clear pending commit operations.

**Options**

**none**—Clear pending commit operations on the static configuration database.

**synchronize-server pending-jobs**—(Optional) Clear pending commit synchronize operations for all instances of the ephemeral configuration database on an MX Series Virtual Chassis or a device with dual Routing Engines. This option can only be executed on the master Routing Engine of the Virtual Chassis master router or the dual Routing Engine system.

**NOTE:** The **clear system commit synchronize-server pending-jobs** command should be used only under the supervision of the Juniper Networks Technical Assistance Center (JTAC) during a troubleshooting session.

**Required Privilege Level**

maintenance (or the actual user who scheduled the commit)

RELATED DOCUMENTATION

show system commit | 427

**List of Sample Output**

clear system commit on page 332
clear system commit (None Pending) on page 332

clear system commit (User Does Not Have Required Privilege Level) on page 332
clear system commit synchronize-server pending-jobs on page 332

**Output Fields**

When you enter this command, you are provided feedback on the status of your request.

## Sample Output

**clear system commit**

```
user@host> clear system commit
```

```
Pending commit cleared.
```

**clear system commit (None Pending)**

```
user@host> clear system commit
```

```
No commit scheduled.
```

**clear system commit (User Does Not Have Required Privilege Level)**

```
user@host> clear system commit
```

```
error: Permission denied
```

**clear system commit synchronize-server pending-jobs**

```
user@host> clear system commit synchronize-server pending-jobs
```

# clear system commit prepared

**Syntax**

```
clear system commit prepared
```

**Release Information**

Command introduced in Junos OS Release 17.3.

**Description**

Clear the prepared commit. This initiates cleanup of the saved database data structures and the necessary files that are generated as a result of the commit preparation stage and unlinks the pending activation file. A log message is generated upon successful clearing of the pending commit.

**Options**

This command has no options.

**Required Privilege Level**

Maintenance (or the actual user who scheduled the commit)

RELATED DOCUMENTATION

clear system commit | 331

**List of Sample Output**

clear system commit prepared on page 333
clear system commit prepared (None Prepared) on page 334

**Output Fields**

When you enter this command, you are provided feedback on the status of your request.

## Sample Output

**clear system commit prepared**

```
user@host> clear system commit prepared
```

```
Prepared commit cleared.
```

**clear system commit prepared (None Prepared)**

```
user@host> clear system commit prepared
```

```
No commit prepared.
```

# commit

**Syntax**

```
commit
<activate>
<and-quit>
<at "string">
<check>
<comment <comment-string>
<confirmed>
<peers-synchronize>
<prepare>
<scripts>
<synchronize | no-synchronize>
< | >
```

**Release Information**

Command introduced before Junos OS Release 7.4.
Command introduced in Junos OS Release 11.1 for the QFX Series.
Option **synchronize scripts** introduced in Junos OS Release 13.2.
Command introduced in Junos OS Release 14.1X53-D20 for the OCX Series.
Option **no-synchronize** introduced in Junos OS Release 17.2R1

**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 the QFX Series Virtual Chassis.

The **peers-synchronize** option is not supported in SRX Series devices.

**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.

In Junos OS Evolved, if an FPC or PIC is brought offline, neither will be started when you enter a **commit** command that configures an element of the offline FPC or PIC.

**Options**

**none**—Execute the commit command without any options to commit the configuration changes to the configuration database.

**activate (Optional)**—Complete commit in two steps of preparing the configuration for commit and later activating the configuration. This enables you configure a number of devices and simultaneously activate the configurations on multiple devices.

**and-quit (Optional)**—Commit the configuration and, if the configuration contains no errors and the commit succeeds, exit from configuration mode.

**at *string***—(Optional) Save software configuration changes and activate the configuration at a future time, or upon reboot. The variable *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 by at least one minute 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 device.
- 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 "2018-03-10 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 mode 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 system commit** command to cancel a scheduled commit configuration, see *clear system commit*.

**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 user".**

**confirmed *in 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.
- The timeout for the **commit confirmed** command is calculated based on the system time, when the **commit confirmed** command is issued. In case the system time is modified while a **commit confirmed** is pending, the remaining time until commit execution might get shortened (in case the old system time is behind) or prolonged (in case the old system time is ahead) from the intended interval.
- In Junos OS Release 11.4 and later, you can also use the **commit confirmed** command in the **[edit private]** configuration mode.

**no-synchronize**—(Optional) Configure the **commit** command to run without synchronization. This can be useful in situations, for example, where a Routine Engine configuration is corrupted such that a commit synchronization is not possible or will block the commit.

- This option allows you to commit only on the current Routing Engine even if **set system commit synchronize** is configured.

- This option overrides the **commit peer-synchronize** configuration as well. If you have configured the commit synchronize using **set system commit synchronize** and then use the command **commit no-synchronize**, the commit will happen only on the device issuing the command.
- When using **commit synchronize**, the commit is first done in the other Routing Engine and then in the current one. If the other Routine Engine is corrupted, the commit will fail. In such cases, you can use **commit no-synchronize**. This command cannot be configured using **set**. It can only be run.

**peers-synchronize**—(Optional) Configure the commit command to automatically perform a peers-synchronize action between peers. The local peer (or requesting peer) on which you enable the peers-synchronize statement copies and loads its configuration to the remote (or 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**—(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.

The synchronize option has the following two additional options:

- force—(Optional) Enforce commit synchronization on the Routing Engines by using the **force** option.

  The **commit synchronize** command does not work if the responding Routing Engine has uncommitted configuration changes. 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 the configuration is synchronized with that on the Routing Engine from which you issued the command.

- scripts—(Optional) Synchronize all commit, event, lib, op, and SNMP scripts from the requesting Routing Engine to the responding Routing Engine and commit and synchronize 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 at the **[edit system scripts]** hierarchy level 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:** 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 sychronize** command again.

**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 139.

The responding Routing Engine must use Junos OS Release 5.0 or later.

**prepare**—(Optional) Prepare the configuration to activate at a later stage. During the preparation stage, all the required files and databases are generated and the configuration is validated. A file is created that indicates if the commit is pending for activation. In the event of failure during the preparation stage, the log message commit preparation failed is generated.

**scripts**—(Optional) Commit newly enabled scripts during the commit operation and push scripts to the other Routing Engine.

**| (pipe)**—(Optional) Use the | (pipe)) options to filter the output of the **commit** command.

**Additional Information**

**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.

**| 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.

**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


- Adding a Comment to Describe the Committed Configuration | **186**
- *Committing Configurations on a Routing Matrix with a TX Matrix Plus Router*
- *Commit Script Overview*
- Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration | **114**
- Monitoring the Junos OS Commit Process | **185**
- Scheduling a Junos OS Commit Operation | **183**
- Verifying the Junos OS Configuration | **174**, Committing a Configuration | **174**

# configure

**List of Syntax**

Syntax on page 341
Syntax (Junos OS Evolved) on page 341

**Syntax**

```
configure
<batch>
<dynamic>
<exclusive>
<private>
```

**Syntax (Junos OS Evolved)**

```
configure
<batch>
<exclusive>
<private>
```

**Release Information**

Command introduced before Junos OS Release 7.4.
Command introduced in Junos OS Release 9.0 for EX Series switches.
The **dynamic** option of the **configure** command is deprecated for Junos OS Evolved.

**Description**

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.

**Options**

**none**—Enter configuration mode.

**batch**—(Optional) Work in the batch commit mode where commit operations are executed in batches.

**dynamic**—(Optional) (Not available for Junos OS Evolved) 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.

**exclusive**—(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.

**private**—(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. This mode does not support configuring statements corresponding to third-party YANG data models, for example, OpenConfig or custom YANG data models.

**Additional Information**

For more information about the different methods of entering configuration mode and the restrictions that apply, see the *Junos OS Administration Library*.

**Required Privilege Level**

configure

RELATED DOCUMENTATION

show configuration | 413

**List of Sample Output**

configure on page 342

**Output Fields**

When you enter this command, you are placed in configuration mode and the system prompt changes from ***hostname>*** to ***hostname#***.

## Sample Output

**configure**

```
user@host> configure
```

```
Entering configuration mode
[edit]
user@host#
```

# copy

**Syntax**

```
copy existing-statement to new-statement
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Make a copy of an existing statement in the configuration.

**Options**

***existing-statement***—Statement to copy.

***new-statement***—Copy of the statement.

**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

Copying a Junos OS Statement in the Configuration | 99

# deactivate

**Syntax**

```
deactivate (statement | identifier)
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

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.

**Options**

***identifier***—Identifier to which you are adding the **inactive:** tag. It must be an identifier at the current hierarchy level.

***statement***—Statement to which you are adding the **inactive:** tag. It must be a statement at the current hierarchy level.

**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

activate | 326

delete | 345

Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration | 114.

# delete

**Syntax**

```
delete <statement-path> <identifier>
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Delete a statement or identifier. All subordinate statements and identifiers contained within the specified statement path are deleted with it.

Deleting a statement or an identifier effectively “unconfigures” or disables the functionality associated with that statement or identifier.

If you do not specify ***statement-path*** or ***identifier***, the entire hierarchy, starting at the current hierarchy level, is removed.

> **NOTE:** For Junos OS Evolved, if you use the **delete** configuration command at the top level of the configuration, you cannot commit the resulting empty configuration. At a minimum, the root authentication password is required.

**Options**

***statement-path***—(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.

***identifier***—(Optional) Name of the statement or identifier to delete.

**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

deactivate | 344

Deleting a Statement from a Junos OS Configuration | 94

**List of Sample Output**

**delete (Junos OS Evolved) on page 346**

## Sample Output

**delete (Junos OS Evolved)**

```
[edit]
```

```
user@host# delete
```

```
This will delete the entire configuration
Delete everything under this level? [yes,no] (no) yes
```

If you then try to commit the change, you get this:

```
user@host# commit
```

```
error: cannot commit an empty configuration
```

# edit

**Syntax**

```
edit statement-path
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Move inside the specified statement hierarchy. If the statement does not exist, it is created.

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

**Options**

***statement-path***—Path to the statement.

**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

Displaying the Current Junos OS Configuration | 165

# exit

**Syntax**

```
exit <configuration-mode>
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

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.

**Options**

**none**—Return to the previous edit level. If you are at the top of the statement hierarchy, exit configuration mode.

**configuration-mode**—(Optional) Exit from configuration mode.

**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

Displaying the Current Junos OS Configuration | 165

# file

**Syntax**

```
file <archive | checksum | compare | copy | delete | list | rename | show | source address>
```

**Release Information**
Command introduced before Junos OS Release 7.4.

**Description**
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.

**Options**

**archive (Optional)**—Archive, and optionally compress, one or multiple local system files as a single file, locally or at a remote location.

**checksum (Optional)**—Calculate the Message Digest 5 (MD5) checksum of a file.

**compare (Optional)**—Compare two local files and describe the differences between them in default, context, or unified output styles.

**copy (Optional)**—Copy files from one place to another on the local device or between the local device and a remote system.

**delete (Optional)**—Delete a file on the local device.

**list (Optional)**—Display a list of files on the local device.

**rename (Optional)**—Rename a file on the local device.

**show (Optional)**—Display the contents of a file.

**source address (Optional)**—Specify the source address of the local file.

**Required Privilege Level**
maintenance

RELATED DOCUMENTATION

Viewing Files and Directories on a Device Running Junos OS | 268

# help

**Syntax**

```
help < (apropos string | reference statement-name| syslog syslog-tag| tip cli number | topic word)>
```

**Release Information**
Command introduced before Junos OS Release 7.4.
The **apropos** option was added in Junos OS Release 8.0.

**Description**

Display help about available operational commands, configuration statements, or general information about getting help. Entering the **help** command without an option provides introductory information about how to use the **help** and **?** commands.

**Options**

**apropos** *string*—(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.

**reference** *statement-name*—(Optional) Display summary information for a configuration statement. This information is based on summary descriptions that appear in the Junos configuration guides.

**syslog** *syslog-tag*—(Optional) Display information about system log messages.

**tip cli** *number*—(Optional) Display a tip about using the CLI. Specify the number of the tip you want to view.

**topic** *word*—(Optional) Display usage guidelines for a topic or configuration statement. This information is based on subjects that appear in the Junos configuration guides.

**Required Privilege Level**
None

RELATED DOCUMENTATION

Getting Online Help from the Junos OS Command-Line Interface | 55

# insert

**Syntax**

```
insert <statement-path> identifier1 (before | after) identifier2
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Insert an identifier in to an existing hierarchy.

**Options**

***statement-path***—(Optional) Path to the existing identifier.

***identifier1***—The existing identifier.

**after**—Place *identifier1* after *identifier2*.

**before**—Place *identifier1* before *identifier2*.

***identifier2***—The new identifier to insert.

**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

Inserting a New Identifier in a Junos OS Configuration | 109

# load

**Syntax**

```
load (factory-default | merge | override | patch | replace | set | update) (filename | terminal)
<json>
<relative>
```

**QFX Series**

```
load (dhcp-snooping filename)
```

**Release Information**

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.
**json** option introduced in Junos OS Release 16.1.

**Description**

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 first configuration for the router or switch 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 "Viewing Files and Directories on a Device Running Junos OS" on page 268.

**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*.

**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

Loading a Configuration from a File or the Terminal | 229

## Sample Output

The following is an example of a load scenario using Secure Copy (scp).

**To Load a Configuration File Using Secure Copy Protocol (scp) with 'source-address' and 'routing-instance' options**

To load a configuration file using the scp command with the **source-address** and **routing-instance** options, enter the following command:

`root@host#` **load merge scp://***`user@hostname/path/filename`* **source-address** *`address`* **routing-instance** ***instance-name***

The scp options **source-address** and **routing-instance** are supported for **load override**, **load patch**, **load replace**, **load set**, and **load update** options also.

# | (pipe)

**Syntax**

```
| (compare | count |
display (changed | commit-scripts | detail | inheritance | json | merge | omit | set | translation-scripts <configured-delta
   | translated-config | translated-delta> | xml) |
except pattern | find pattern | hold | last lines | match pattern | no-more | refresh interval |
request message (all | account@terminal)
resolve <full-names> | save filename | append filename | tee | trim columns )
```

**Release Information**

Command introduced before Junos OS Release 7.4.
**display commit-scripts** option added in Junos OS Release 7.4.
**tee** option added in Junos OS Release 14.1.
**display json** option added in Junos OS Release 14.2.
**compare | display xml** option added in Junos OS Release 15.1.
**display translation-scripts** option added in Junos OS Release 16.1.
**display merge** option added in Junos OS Release 16.2R2.
**display merge** option deprecated in Junos OS Release 18.2R1.

**Description**

Filter the output of an operational mode or a configuration mode command.

**Options**

**append *filename***—Append the output to a file.

**compare (filename | rollback *n* )**—Compare configuration changes with another configuration file. In operational mode, use the **show configuration** command. In configuration mode, use the **show** command.

**compare | display xml**—Compare configuration changes with the active configuration and display them in XML format. In operational mode, use the **show configuration** command. In configuration mode, use the **show** command.

**count**—Display the number of lines in the output.

**display**—Display additional information about the configuration contents.

**changed**—Tag changes with **junos:changed attribute** (XML only).

**commit-scripts**—(Configuration mode only) Display all statements that are in a configuration, including statements that were generated by transient changes.

**detail**—(Configuration mode only) Display configuration data detail.

**inheritance <brief | default | no-comments |groups | terse>**—(Configuration mode only) Display inherited configuration data and source group.

**json**—Display the output for operational commands and configuration data in JavaScript Object Notation (JSON) format.

**merge**—Use with the **show ephemeral-configuration** command to display the merged view of the static and ephemeral configuration databases in Junos OS Release 18.1 and earlier releases. Issuing the **show ephemeral-configuration | display merge** command displays the configuration data from all instances of the ephemeral configuration database merged with the complete post-inheritance configuration.

Starting in Junos OS Release 18.2R1, to display the merged view of the static and ephemeral configuration databases, use the **show ephemeral-configuration merge** command instead.

**omit**—(Configuration mode only) Display configuration statements omitted by the **apply-flags omit** configuration statement.

**set**—Display the configuration as a series of configuration mode commands required to re-create the configuration.

**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.

**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.

**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 | **165**.

Using the Pipe ( | ) Symbol to Filter Junos OS Command Output | **281**

Comparing Configurations and Displaying the Differences in Text | **285**

Understanding the show | compare | display xml Command Output | **203**

*Automation Scripting User Guide*

Displaying Output Beginning with the Last Entries | **294**.

Viewing Files and Directories on a Device Running Junos OS | **268**.

# protect

**Syntax**

```
protect (hierarchy | statement | identifier)
```

**Release Information**

Command introduced in Junos OS Release 11.2.

**Description**

Protect a hierarchy, statement, or identifier from modification or deletion.

**Options**

***hierarchy***—(Optional) Protect a specific hierarchy.

***statement***—(Optional) Protect a specific statement.

***identifier***—(Optional) Protect a specific identifier.

none

**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

*Example: Protecting the Junos OS Configuration from Modification or Deletion*

unprotect | 447

# quit

**Syntax**

```
quit <configuration-mode>
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

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.

**Options**

**none**—Return to the previous edit level. If you are at the top of the statement hierarchy, exit configuration mode.

**configuration-mode**—(Optional) Exit from configuration mode.

**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

Displaying the Current Junos OS Configuration | 165

# 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-0/1/0.0** to **ge-0/1/10.0** 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*

# replace

**Syntax**

```
replace pattern pattern1 with pattern2 <upto n>
```

**Release Information**

Command introduced in Junos OS Release 7.6.

**Description**

Replace identifiers or values in a configuration.

**Options**

***pattern1***—Text string or regular expression that defines the identifiers or values you want to match.

***pattern2***—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.

**upto *n***—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. If you do not specify an upto option, all identifiers and values in the configuration that match *pattern1* are replaced.

**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

Using Global Replace in the Junos OS Configuration | 117

# 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*

# 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 | 188

## 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

**Syntax**

```
request system commit server queue cleanup <id commit-id | job-status (error | pending | success)>
```

**Release Information**
Command introduced in Junos OS Release 12.1.

**Description**
Clean up the batch commit queue. Note that the **id** argument cleans up batch commit operation messages for a specific commit ID, whereas **job-status** cleans up more broadly, based on categories of status messages. You can use either option, but not both.

**Options**
**id *commit-id***—(Optional) Clean up batch commit operation status messages for a specific commit ID.

**job-status**—(Optional) Clean up batch commit operation status messages for the following:

- **error**—Clean up status messages for batch commit operations that have errors.
- **pending**—Clean up status messages for batch commit operations that are pending.
- **success**—Clean up status messages for batch commit operations that are successful.

**Required Privilege Level**
view

RELATED DOCUMENTATION

Example: Configuring Batch Commit Server Properties | 188

**List of Sample Output**
**request system commit server queue cleanup id on page 368**
**request system commit server queue cleanup job-status success on page 368**

## Sample Output

When you enter the **request system commit server queue cleanup** command, you are provided feedback on the status of your request. The first example demonstrates cleaning up job ID 1008, while the second shows a queue clean up for all jobs marked as successfully completed.

**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

**Syntax**

```
request system commit server start
```

**Release Information**

Command introduced in Junos OS Release 12.1.

**Description**

Start the commit server.

**Options**

This command has no options.

**Required Privilege Level**

view

RELATED DOCUMENTATION

Example: Configuring Batch Commit Server Properties | 188

## 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.
```

# request system configuration rescue delete

**Syntax**

```
request system configuration rescue delete
```

**Release Information**

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 OCX Series switches.

**Description**

Delete an existing rescue configuration.

> **NOTE:** The **[edit system configuration]** hierarchy is not available on QFabric systems.

**Options**

This command has no options.

**Required Privilege Level**

maintenance

**List of Sample Output**

request system configuration rescue delete on page 370

**Output Fields**

This command produces no output.

## Sample Output

**request system configuration rescue delete**

```
user@host> request system configuration rescue delete
```

# request system configuration rescue save

**Syntax**

```
request system configuration rescue save
```

**Release Information**

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 OCX Series switches.

**Description**

Save the most recently committed configuration as the rescue configuration so that you can return to it at any time by using the **rollback** command.

> **NOTE:** The **[edit system configuration]** hierarchy is not available on QFabric systems.

**Options**

This command has no options.

**Required Privilege Level**

maintenance

**List of Sample Output**

request system configuration rescue save on page 371

**Output Fields**

This command produces no output.

## Sample Output

**request system configuration rescue save**

```
user@host> request system configuration rescue save
```

# restart

**List of Syntax**

Syntax on page 372
Syntax (ACX Series Routers) on page 372
Syntax (EX Series Switches) on page 373
Syntax (MX Series Routers) on page 373
Syntax (QFX Series) on page 373
Syntax (Routing Matrix) on page 374
Syntax (TX Matrix Routers) on page 374
Syntax (TX Matrix Plus Routers) on page 374
Syntax (QFX Series) on page 374
Syntax (Junos OS Evolved) on page 375

**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-health-monitoring |
   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 <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 | mountd-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-health-monitoring | kernel-replication | l2-learning |
   lacp | license-service | link-management | lldpd-service | mib-process | mountd-service | multicast-snooping | pgm
   | redundancy-interface-process | remote-operations | routing | secure-neighbor-discovery | service-deployment
   | sflow-service | snmp | vrrp | web-management>
```

**Syntax (MX Series Routers)**

```
restart
<adaptive-services | ancpd-service | application-identification | audit-process | auto-configuration | bbe-stats-service
   | 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-health-monitoring |
   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
   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>
<all-members>
<gracefully | immediately | soft>
<local>
<member member-id>
```

**Syntax (QFX Series)**

```
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 |logical-system-name>  | routing | sampling
   |secure-neighbor-discovery | service-deployment | snmp | usb-control | web-management>
<gracefully | immediately | soft>
```

**Syntax (Routing Matrix)**

```
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 logical-system-name>  | sampling |
   service-deployment | snmp>
<all | all-lcc | lcc number>
<gracefully | immediately | soft>
```

**Syntax (TX Matrix Routers)**

```
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 logical-system-name>
   | sampling | service-deployment | snmp| statistics-service>
<all-chassis | all-lcc | lcc number | scc>
<gracefully | immediately | soft>
```

**Syntax (TX Matrix Plus Routers)**

```
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 logical-system-name>
   | sampling | service-deployment | snmp| statistics-service>
<all-chassis | all-lcc | all-sfc | lcc number | sfc number>
<gracefully | immediately | soft>
```

**Syntax (QFX Series)**

```
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 |logical-system-name>  | routing | sampling
    |secure-neighbor-discovery | service-deployment | snmp | usb-control | web-management>
<gracefully | immediately | soft>
```

**Syntax (Junos OS Evolved)**

```
restart (BdL2Token | aft-sysinfo | agentd | alarmd | arpd | audit-process | bcmd_evo | bfdd | bios-manager | charonctl
    | chassis-control | class-of-service | clksynced | cmevod | command-handler | command-relay | configd | ddosd |
    dfwd-junos-relay | diskmgmt | distributor | dot1x-protocol | dot1xd-agent | edo | emfca |
    ethernet-connectivity-fault-management | ethernet-link-fault-management | event-processing | evo-aftmand-zx
    | evo-cda-zx | evo-cda-zx-diag | evo-jet-sdk-broker | evoaft-jvisiond | fabricHub | fabspoked-fchip | fabspoked-pfe
    | fabtokend | fibd | fibd-proxy | firewall | fpa | fwstatsd | gcd | hwddual | hwdfpc | hwdspmb | icmpd |
    idmd-dest-usage-class | idmd-src-usage-class | idmdbd | idmdcounter | idmdfabtoken | idmdfilter | idmdfilterterm
    | idmdfwgretunnel | idmdifd | idmdifl | idmdnh | idmdoffchip32 | idmdoffchip64 | idmdonchip | dmdpolicer | idmdrtb
    | idmdsensor |idmdsgid | idmdstp | ifstatsd | imgd | interface-control | jdhcpd | jinsightd | jsd | jstatsd | kfirewall-agent
    | l2agent | l2ald | l2cpd | l2cpd-agent | lacp | license-check | lldpd | mem-mgmt | mfilterd | mgd | mgd-api | mgd-pfe
    | mgmt-ethd | mib-process | mplsoamd | mstr | mstrzk | msvcsd | mstrzk | msvcsd | mustd | na-grpcd | na-mqtt |
    ndp | netdefaultsd | nlsd | objmon | objping-server | ofp | ofp-command | opticmand | orchestratord | packetio-zx
    | pccd | pci-agent | pdevmand | pfestatsd | picd | ppman | ppmd | ppmdagent | resild | routing | rpcserviced | rpdfw
    | securityd | sflowd | sinetd | smartd-agent-monitor | snmp | snmpd-subagent | svcsd | syscmdd | sysepochman |
    sysman | sysman-ui | trace-relay | trace-writer | xmlproxyd | ztp)
<gracefully | immediately | soft>
```

**Release Information**

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 12.2 for ACX Series routers.
Command introduced in Junos OS Release 14.1X53-D20 for the OCX Series.
Options added:

- **dynamic-flow-capture** in Junos OS Release 7.4.
- **dlsw** 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 Junos 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.
- **bbe-stats-service** in Junos OS Release 18.4R1 on MX Series routers.
- **kernel-health-monitoring** in Junos OS Release 19.1R1.
- Introduced in Junos OS Evolved Release 19.1R1.

**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.

For Junos OS Evolved, the **restart** command also triggers a restart of the dependent applications (apps). In order to inform you which dependent apps are being restarted the following message will be logged when the **restart** command is used:

**App restarting <app name>. Related apps that may be impacted - <related-app name>** . For example: **Jan 14 11:42:08 RE0 sysman[5100]: SYSTEM_APP_RESTARTING_WITH_RELAPPS_EVENT: App restarting re0-ifmand. Related apps that may be impacted - aggd**

Starting in Junos OS Evolved Release 20.1R1, if you specify **restart *app-name*** and the application is not supposed to run on the platform, the error message is as follows:

```
user@device> restart fabspoked-pfe

Restart failed for fabspoked-pfe on node re0. Application is not running.
```

The **restart** command expands all applications names including applications that are not required for the current platform. Therefore, a user could try to do a restart for an application that is not running for the current platform. This error message communicates that the restart failed because the application was not running on the system.

**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.

**bbe-stats-service**—(MX Series routers only) (Optional) Restart bbe-statsd, the BBE statistics collection and management 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.

**dlsw**—(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-health-monitoring**—(Optional) Restart the Routing Engine kernel health monitoring process, which enables health parameter data to be sent from kernel components to data collection applications. When you change the polling interval through **sysctl kern.jkhmd_polling_time_secs**, you must restart the kernel health monitoring process for the new polling interval to take effect.

**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.

**mountd-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*

**List of Sample Output**
**restart interface-control gracefully on page 383**
**restart interface-control (Junos OS Evolved) on page 383**

**Output Fields**
When you enter this command, you are provided feedback on the status of your request.

## Sample Output

**restart interface-control gracefully**

```
user@host> restart interface-control gracefully

Interface control process started, pid 41129
```

**restart interface-control (Junos OS Evolved)**

```
user@host> restart interface-control
```

```
interface-control restart requested
Restarted aggd on re0
Restarted ifmand on re0
```

# rollback

**Syntax**

```
rollback <number | rescue>
```

**Release Information**

Command introduced before Junos OS Release 7.4.
Command introduced in Junos OS Release 11.1 for the QFX Series.

**Description**

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 **commit** configuration command.

The currently operational Junos OS configuration is stored in the file **juniper.conf**, and the last three committed configurations are stored in the files **juniper.conf.1**, **juniper.conf.2**, and **juniper.conf.3**. These four files are located in the directory **/config**, which is on the router's flash drive. The remaining 46 previous committed configurations, the files **juniper.conf.4** through **juniper.conf.49**, are stored in the directory **/var/db/config**, which is on the router's hard disk.

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 **load update**).

**Options**

**none**—(Optional) Return to the most recently saved configuration.

***number***—(Optional) Configuration to return to. The range of values is from 0 through 49. The most recently saved configuration is number 0, and the oldest saved configuration is number 49. The default is 0.

**rescue**—(Optional) Return to the rescue configuration.

**Required Privilege Level**

rollback—To roll back to configurations other than the one most recently committed.

RELATED DOCUMENTATION

Returning to a Previously Committed Junos OS Configuration | 213
Creating and Returning to a Rescue Configuration | 246

# run

**Syntax**

```
run command
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Run a top-level CLI command without exiting from configuration mode.

**Options**

***command***—CLI top-level command.

**Required Privilege Level**

configure—To enter configuration mode.

RELATED DOCUMENTATION

Understanding Junos OS CLI Configuration Mode | 71

# save

**Syntax**

```
save filename
```

**QFX Series**

```
save (<dhcp- security-snoop | dhcpv6-security-snoop>filename)
```

**Release Information**

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.

**Description**

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.

For information on valid filename and URL formats, see *Format for Specifying Filenames and URLs in Junos OS CLI Commands*.

When saving a file to a remote system, the software uses the **scp**/**ssh** protocol.

**Options**

***filename***—Name of the saved file. You can specify a filename in one of the following ways:

- ***filename***—File in the user's home directory (the current directory) on the local flash drive.
- ***path/filename***—File on the local flash drive.
- **/var/*filename*** or **/var/*path/filename***—File on the local hard disk.
- **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 the string **%2F**; for example, **ftp://*hostname*/%2F*path*/*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@ftphostname.net//filename
```

Password for ***username***@ftp.***hostname***.net:

- **http://*hostname*/*path*/*filename***—File on a Hypertext Transfer Protocol (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.

Options for QFX Series

- **dhcp-security-snoop**—Save DHCP snooping entries
- **dhcpv6-security-snoop**—Save DHCPv6 snooping entries

**Required Privilege Level**
configure—To enter configuration mode.

RELATED DOCUMENTATION

Deactivating and Reactivating Statements and Identifiers in a Junos OS Configuration | 114

## Sample Output

The following is an example of a save scenario:

**Save a File Using Secure Copy Protocol (scp) with 'source-address' and 'routing-instance' options**

To use the scp command to save local file to a remote system with the **source-address** and **routing-instance** enter the following command:

`root@host#` **save scp://***`user@hostname/path/filename`* **routing-instance *instance-name*** **source-address** *`address`*

# set

### Syntax

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

### Release Information

Command introduced before Junos OS Release 7.4.

### Description

Create a statement hierarchy and set identifier values. This is similar to **edit** except that your current level in the hierarchy does not change.

### Options

***identifier***—Name of the statement or identifier to set.

***statement-path***—(Optional) Path to an existing statement hierarchy level. If that hierarchy level does not exist, it is created.

### 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

edit | 347

Displaying the Current Junos OS Configuration | 165

# set cli complete-on-space

**Syntax**

```
set cli complete-on-space (off | on)
```

**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 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.

**Options**

**off**—Turn off command completion.

**on**—Allow either a space or a tab to be used for command completion.

**Required Privilege Level**

view

RELATED DOCUMENTATION

> *CLI User Interface Overview*
>
> show cli | **405**

**List of Sample Output**

**set cli complete-on-space on page 390**

**Output Fields**

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

**Syntax**

```
set cli directory directory
```

**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 current working directory.

**Options**

***directory***—Pathname of the working directory.

**Required Privilege Level**

view

RELATED DOCUMENTATION

*CLI User Interface Overview*

show cli directory | **411**

**List of Sample Output**

**set cli directory on page 392**

**Output Fields**

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

**Syntax**

```
set cli idle-timeout <minutes>
```

**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 maximum time that an individual session can be idle before the user is logged off the router or switch. **set cli idle-timeout** holds good only for the session in use when you enter it. If you need to configure the idle timeout permanently for all the CLI sessions, then configure the **idle-timeout** statement at the **[edit system login]** hierarchy level.

**Options**

***minutes***—(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.

**Required Privilege Level**

view

RELATED DOCUMENTATION

*CLI User Interface Overview*

show cli | **405**

**List of Sample Output**

**set cli idle-timeout on page 393**

**Output Fields**

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

**Syntax**

```
set cli prompt string
```

**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 prompt so that it is displayed within the CLI.

**Options**

***string***—CLI prompt string. To include spaces in the prompt, enclose the string in quotation marks. By default, the string is *username@hostname*.

**Required Privilege Level**

view

RELATED DOCUMENTATION

> *CLI User Interface Overview*
>
> show cli | **405**

**List of Sample Output**

**set cli prompt on page 395**

**Output Fields**

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

**Syntax**

```
set cli restart-on-upgrade string (off | on)
```

**Release Information**
Command introduced before Junos OS Release 7.4.
Command introduced in Junos OS Release 9.0 for EX Series switches.

**Description**

For an individual session, set the CLI to prompt you to restart the router or switch after upgrading the software.

**Options**

**off**—Disables the prompt.

**on**—Enables the prompt.

**Required Privilege Level**
view

RELATED DOCUMENTATION

> *CLI User Interface Overview*
>
> show cli | **405**

**List of Sample Output**
**set cli restart-on-upgrade on page 396**

**Output Fields**

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

**Syntax**

```
set cli screen-length length
```

**Release Information**
Command introduced before Junos OS Release 7.4.

**Description**

Set terminal screen length.

```
user@host> set cli screen-length 75
```

```
Screen length set to 75
```

**Options**

*length*—Number of lines of text that the terminal screen displays. The range of values, in an integer number of lines, is 2 through 100,000. The default is 24.

The point at which the **---(more)---** prompt appears on the screen is a function of this setting and the settings for the **set cli screen-width** and **set cli terminal** commands.

**Required Privilege Level**
view

RELATED DOCUMENTATION


- Setting the Screen Length | 66
- Setting the Junos OS CLI Screen Length and Width | 65
- set cli screen-width | 398
- set cli terminal | 399
- show cli | 405

# 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 Junos OS CLI Screen Length and Width | 65
- set cli screen-length | 397
- set cli terminal | 399
- show cli | 405

# set cli terminal

**Syntax**

```
set cli terminal terminal-type
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Set the terminal type.

```
user@host> set cli terminal xterm
```

**Options**

*terminal-type*—Type of terminal that is connected to the Ethernet management port:

- **ansi**—ANSI-compatible terminal
- **pc**—PC screen command-prompt window
- **small-xterm**—Small xterm window (24 lines long)
- **vt100**—VT100-compatible terminal
- **xterm**—Large xterm window (65 lines long)

**Required Privilege Level**

view

RELATED DOCUMENTATION

Customizing the Junos OS CLI Environment | 62

# set cli timestamp

**Syntax**

```
set cli timestamp (format timestamp-format | disable)
```

**Release Information**

Command introduced before Junos OS Release 7.4.
Command introduced in Junos OS Release 9.0 for EX Series switches.

**Description**

Set a timestamp for CLI output.

**Options**

**format *timestamp-format***—Set the date and time format for the timestamp. The timestamp format you specify can include the following placeholders in any order:

- **%m**—Two-digit month
- **%d**—Two-digit date
- **%T**—Six-digit hour, minute, and seconds

**disable**—Remove the timestamp from the CLI.

> **NOTE:** A timestamp is displayed by default when no command output is generated.

**Required Privilege Level**

view

RELATED DOCUMENTATION

*CLI User Interface Overview*

show cli | 405

**List of Sample Output**

set cli timestamp on page 401

**Output Fields**

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

**Syntax**

```
set date (date-time | ntp <ntp-server> <key key> <source-address source-address>)
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

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
```

**Options**

- ***date-time***—Specify date and time in one of the following formats:
  - ***YYYYMMDDHHMM.SS***
  - **"*month DD*, *YYYY HH:MM*(am | pm)"**
- **ntp**—Configure the router to synchronize the current date and time setting with a Network Time Protocol (NTP) server.

  > **NOTE:** In Junos OS Evolved, if the ntpd server is running, the **set date ntp** command fails with the following error message: **error: ntpd is already running**. To use this command, you must first stop the ntpd server

- ***ntp-server***—(Optional) Specify the IP address of one or more NTP servers.
- **key *key***—Configure the key to authenticate the NTP server.
- **source-address *source-address***—(Optional) Specify the source address that is used by the router to contact the remote NTP server.

**Required Privilege Level**

view

RELATED DOCUMENTATION

*Setting the Date and Time Locally*

**List of Sample Output**

set date ntp (Junos OS) on page 403
set date ntp (Junos OS Evolved) on page 403

## Sample Output

**set date ntp (Junos OS)**

```
user@host> set date ntp

22 Jun 10:07:48 ntpdate[51123]: step time server 66.129.255.62 offset -0.013200
sec
```

**set date ntp (Junos OS Evolved)**

```
user@host> set date ntp

------------------------------
node: re0
------------------------------
error: ntpd is already running
```

# show

**Syntax**

```
show <statement-path> <identifier>
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Display the current configuration.

**Options**

none—Display the entire configuration at the current hierarchy level.

***identifier***—(Optional) Display the configuration for the specified identifier.

***statement-path***—(Optional) Display the configuration for the specified statement hierarchy path.

**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


- show | display inheritance | **417**
- show | display omit | **420**
- show | display set | **421**
- show | display set relative | **423**
- show groups junos-defaults | **425**
- Displaying the Current Junos OS Configuration | **165**

# show cli

**Syntax**

```
show cli
```

**Release Information**

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.

**Description**

Display configured CLI settings.

**Options**

This command has no options.

**Required Privilege Level**

view

RELATED DOCUMENTATION


- show cli authorization | 407
- show cli directory | 411
- show cli history | 412


**List of Sample Output**

**show cli on page 406**

**Output Fields**

Table 19 on page 405 lists the output fields for the **show cli** command. Output fields are listed in the approximate order in which they appear.

**Table 19: 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: **on** or **off**. |

**Table 19: show cli Output Fields *(continued)***

| Field Name | Field Description |
|---|---|
| **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 **disabled**. |
| **CLI restart-on-upgrade** | CLI is set to prompt you to restart the router or switch after upgrading the software: **on** or **off**. |
| **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: **enhanced**. |
| **CLI timestamp** | Date and time format for the timestamp. If the timestamp is not set, the state is **disabled**. |
| **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 authorization

**Syntax**

```
show cli authorization
```

**Release Information**

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.

**Description**

Display the permissions for the current user.

**Options**

This command has no options.

**Required Privilege Level**

view

**List of Sample Output**

show cli authorization on page 409

**Output Fields**

Table 20 on page 407 lists the output fields for the **show cli authorization** command. In the table, all possible permissions are displayed and output fields are listed in alphabetical order.

**Table 20: show cli authorization Output Fields**

| Field Name | Field Description |
|---|---|
| **access** | Can view access configuration information. |
| **access-control** | Can modify access configuration. |
| **admin** | Can view user account information. |
| **admin-control** | Can modify user account information. |
| **clear** | Can clear learned network information. |
| **configure** | Can enter configuration mode. |

Table 20: show cli authorization Output Fields *(continued)*

| Field Name | Field Description |
|---|---|
| **control** | Can modify any configuration. |
| **edit** | Can edit configuration files. |
| **field** | Reserved for field (debugging) support. |
| **firewall** | Can view firewall configuration information. |
| **firewall-control** | Can modify firewall configuration information. |
| **floppy** | Can read from and write to removable media. |
| **flow-tap** | Can view flow-tap configuration information. |
| **flow-tap-control** | Can configure flow-tap configuration information. |
| **idp-profiler-operation** | Can configure Profiler data. |
| **interface** | Can view interface configuration information. |
| **interface-control** | Can modify interface configuration information. |
| **maintenance** | Can perform system maintenance. |
| **network** | Can access the network by entering the **ping**, **ssh**, **telnet**, and **traceroute** commands. |
| **pgcp-session-mirroring** | Can view Packet Gateway Control Protocol session mirroring configuration. |
| **pgcp-session-mirroring-control** | Can modify Packet Gateway Control Protocol session mirroring configuration all-control. |
| **reset** | Can reset or restart interfaces and system processes. |
| **rollback** | Can roll back to previous configurations. |
| **routing** | Can view routing configuration information. |
| **routing-control** | Can modify routing configuration information. |
| **secret** | Can view passwords and authentication keys in the configuration. |

Table 20: show cli authorization Output Fields *(continued)*

| Field Name | Field Description |
|---|---|
| **secret-control** | Can modify passwords and authentication keys in the configuration. |
| **security** | Can view security configuration information. |
| **security-control** | Can modify security configuration information. |
| **shell** | Can start a local shell. |
| **snmp** | Can view SNMP configuration information. |
| **snmp-control** | Can modify SNMP configuration information. |
| **system** | Can view system configuration information. |
| **system-control** | Can modify system configuration information. |
| **trace** | Can view trace file settings information. |
| **trace-control** | Can modify trace file settings information. |
| **view** | Can view current values and statistics. |
| **view-configuration** | Can view all configuration information (not including secrets). |

## Sample Output

**show cli authorization**

`user@host>` **show cli authorization**

```
Current user: 'remote' login: 'user' class ''
Permissions:
    admin       -- Can view user accounts
    admin-control-- Can modify user accounts
    clear       -- Can clear learned network information
    configure   -- Can enter configuration mode
    control     -- Can modify any configuration
    edit        -- Can edit full files
    field       -- Special for field (debug) support
```

```
    floppy      -- Can read and write from 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 configuration
    secret-control-- Can modify secret configuration
    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 configure flow-tap service
Individual command authorization:
    Allow regular expression: none
    Deny regular expression: none
    Allow configuration regular expression: none
    Deny configuration regular expression: none
```

# show cli directory

**Syntax**

```
show cli directory
```

**Release Information**

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.

**Description**

Display the current working directory.

`user@host>` **show cli directory**

```
Current directory: /var/home/user
```

**Options**

This command has no options.

**Required Privilege Level**

view

# show cli history

**Syntax**

```
show cli history
<count>
```

**Release Information**

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.

**Description**

Display a list of previous CLI commands.

`user@host>` **show cli history**

```
11:14:14 -- show arp
11:22:10 -- show cli authorization
11:27:12 -- show cli history
```

**Options**

**none**—Display all previous CLI commands.

***count***—(Optional) Maximum number of commands to display.

**Required Privilege Level**

view

# show configuration

**Syntax**

```
show configuration
<statement-path>
```

**Release Information**

Command introduced before Junos OS Release 7.4.
Command introduced in Junos OS Release 9.0 for EX Series switches.

**Description**

Display the configuration that currently is running on the router or switch, which is the last committed configuration.

**Options**

**none**—Display the entire configuration.

***statement-path***—(Optional) Display one of the following hierarchies in a configuration. (Each ***statement-path*** option has additional suboptions not described here. See the appropriate user guide or EX Series switch documentation for more information.)

- **access**—Network access configuration.
- **access-profile**—Access profile configuration.
- **accounting-options**—Accounting data configuration.
- **applications**—Applications defined by protocol characteristics.
- **apply-groups**—Groups from which configuration data is inherited.
- **chassis**—Chassis configuration.
- **chassis network-services**—Current running mode.
- **class-of-service**—Class-of-service configuration.
- **diameter**—Diameter base protocol layer configuration.
- **ethernet-switching-options**—(EX Series switch only) Ethernet switching configuration.
- **event-options**—Event processing configuration.
- **firewall**—Firewall configuration.
- **forwarding-options**—Options that control packet sampling.
- **groups**—Configuration groups.
- **interfaces**—Interface configuration.

- **jsrc**—JSRC partition configuration.
- **jsrc-partition**—JSRC partition configuration.
- **logical-systems**—Logical system configuration.
- **poe**—(EX Series switch only) Power over Ethernet configuration.
- **policy-options**—Routing policy option configuration.
- **protocols**—Routing protocol configuration.
- **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.
- **vlans**—(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 | 165

Overview of Junos OS CLI Operational Mode Commands | 258

**List of Sample Output**

show configuration on page 415
show configuration policy-options on page 416

**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

**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 in-line comments marked with **##**.

**terse**—Display terse output with inheritance details as an in-line comment.

**Required Privilege Level**
view

RELATED DOCUMENTATION

Using the junos-defaults Configuration Group | 141

**List of Sample Output**
**show system login class readonly | display inheritance on page 418**
**show system login class readonly | display inheritance brief on page 418**
**show system ports | display inheritance defaults on page 418**
**show system login class readonly | display inheritance no-comments on page 418**
**show system login class readonly | display inheritance terse on page 419**

**Output Fields**
When you enter this command, you are provided feedback on the status of your request.

## Sample Output

**show system login class readonly | display inheritance**

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 ];
```

**show system login class readonly | display inheritance brief**

user@host#**show system login class readonly | display inheritance brief**

```
## '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 ];
```

**show system ports | display inheritance defaults**

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;
```

**show system login class readonly | display inheritance no-comments**

user@host# **show system login class readonly | display inheritance no-comments**

```
permissions [ interface network routing system trace view ];
```

**show system login class readonly | display inheritance terse**

user@host# **show system login class readonly | display inheritance terse**

```
permissions [ interface network routing system trace view ]; ## inherited from
group 'global'; inherited from group 'global'; inherited from group 'global';
inherited from group 'global'; inherited from group 'global'; inherited from group
 'global'
```

# 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;
        }
    }
```

Following is an example that shows how to set omit:

```
user@host#set system apply-flags omit
    [edit]
user@host# commit
    commit complete
```

**Required Privilege Level**

view

RELATED DOCUMENTATION

show | 404

# show | display set

**Syntax**

```
show | display set <explicit>
```

**Release Information**

Command introduced before Junos OS Release 7.4.
Option **explicit** command introduced in Junos OS Release 20.1R1.

**Description**

Display the configuration as a series of configuration mode commands required to re-create the configuration from the top level of the hierarchy as **set** commands.

**Options**

***explicit***—Display explicitly, as a series of commands, all the configurations that the system internally creates when you configure certain statements from the top level of the hierarchy.

**Required Privilege Level**

view

RELATED DOCUMENTATION

show | **404**
Displaying set Commands from the Junos OS Configuration | **171**

## Sample Output

Sample output for the **show | display set <*explicit*>** command:

```
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
```

```
user@host# show | display set explicit
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 inet
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 0
set interfaces fe-0/0/0 unit 1 family inet address 10.0.0.1/8
set interfaces fe-0/0/0 unit 1 family inet
deactivate interfaces fe-0/0/0 unit 1
```

# show | display set relative

**Syntax**

```
show | display set relative <explicit>
```

**Release Information**

Command introduced before Junos OS Release 7.4.
Option **explicit** command introduced in Junos OS Release 20.1R1.

**Description**

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

**Options**

***explicit***—Display explicitly, as a series of commands, all the configurations that the system internally creates when you configure certain statements from the current hierarchy level.

**Required Privilege Level**

view

RELATED DOCUMENTATION

Displaying set Commands from the Junos OS Configuration | 171

## Sample Output

Sample output for the **show | display set relative <*explicit*>** command:

```
[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
user@host# show | display set relative explicit
set unit 0 family inet address 192.168.1.230/24
set unit 0 family inet
set unit 0 family iso
set unit 0 family mpls
set unit 0
set unit 1 family inet address 10.0.0.1/8
set unit 1 family inet
deactivate unit 1
```

# 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.

# show system commit

**Syntax**

```
show system commit
<revision | server | synchronize-server pending-jobs>
```

**Release Information**

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.
Option **server** introduced in Junos OS Release 12.1 for the PTX Series router.
Option **revision** introduced in Junos OS Release 14.1.
Command introduced in Junos OS Release 14.1X53-D20 for OCX Series switches.
Option **synchronize-server** introduced in Junos OS Release 17.2R1.

**Description**

Display the system commit history and pending commit operations.

**Options**

**none**—Display the last 50 commit operations on the static configuration database, starting with the most recent.

**revision**—(Optional) Display the revision number of the active configuration of the Routing Engine(s).

**server**— (Optional) Display the commit server status.

**NOTE:** 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.

**synchronize-server pending-jobs**—(Optional) Display the pending commit synchronize operations for all instances of the ephemeral configuration database on an MX Series Virtual Chassis or a device with dual Routing Engines. This option can only be executed on the master Routing Engine of the Virtual Chassis master router or the dual Routing Engine system.

**Required Privilege Level**

view

RELATED DOCUMENTATION

clear system commit | 331

*show system commit revision*

**List of Sample Output**

show system commit on page 429
show system commit (At a Particular Time) on page 429
show system commit (At the Next Reboot) on page 429
show system commit (Rollback Pending) on page 429
show system commit (QFX Series) on page 429
show system commit synchronize-server pending-jobs on page 429

**Output Fields**

Table 21 on page 428 describes the output fields for the **show system commit** command. Output fields are listed in the approximate order in which they appear.

**Table 21: show system commit Output Fields**

| Field Name | Field Description | Level of Output |
|---|---|---|
| ***<number>*** | Displays the last 50 commit operations listed, most recent to first. The identifier ***<number>*** designates a configuration created for recovery using the **request system configuration rescue save** command. | **none** |
| ***<time-stamp>*** | Date and time of the commit operation. | **none** |
| ***<root>/<username>*** | User who executed the commit operation. | **none** |
| ***<method>*** | Method used to execute the commit operation:<br>• **CLI**—CLI interactive user performed the commit operation.<br>• **Junos XML protocol**—Junos XML protocol client performed the commit operation.<br>• **synchronize**—The **commit synchronize** command was performed on the other Routing Engine.<br>• **snmp**—An SNMP **set** request caused the commit operation.<br>• **button**—A button on the router or switch was pressed to commit a rescue configuration for recovery.<br>• **autoinstall**—A configuration obtained through autoinstallation was committed.<br>• **other**—When there is no login name associated with the session, the values for user and client default to root and other. For example, during a reboot after package installation, mgd commits the configuration as a system commit, and there is no login associated with the commit. | **none** |

## Sample Output

**show system commit**

```
user@host> show system commit
```

```
0   2003-07-28 19:14:04 PDT by root via other
1   2003-07-25 22:01:36 PDT by user via cli
2   2003-07-25 22:01:32 PDT by user via cli
3   2003-07-25 21:30:13 PDT by root via button
4   2003-07-25 13:46:48 PDT by user via cli
5   2003-07-25 05:33:21 PDT by root via autoinstall
...
rescue  2002-05-10 15:32:03 PDT by root via other
```

**show system commit (At a Particular Time)**

```
user@host> show system commit
```

```
commit requested by root via cli at Tue May  7 15:59:00 2002
```

**show system commit (At the Next Reboot)**

```
user@host> show system commit
```

```
commit requested by root via cli at reboot
```

**show system commit (Rollback Pending)**

```
user@host> show system commit
```

```
0 2005-01-05 15:00:37 PST by root via cli commit confirmed, rollback in 3mins
```

**show system commit (QFX Series)**

```
user@switch> show system commit
```

```
0   2011-11-25 19:17:49 PST by root via cli
```

**show system commit synchronize-server pending-jobs**

```
user@host> show system commit synchronize-server pending-jobs
```

```
Job Id                Commit Synchronize Model       Database
     Version
36                    Asynchronous Commit            ephemeral
     130
37                    Asynchronous Commit            ephemeral
     131
38                    Asynchronous Commit            ephemeral
     132
39                    Asynchronous Commit            ephemeral
     133
40                    Asynchronous Commit            ephemeral
     134
41                    Asynchronous Commit            ephemeral
     135
42                    Asynchronous Commit            ephemeral
     2
```

# show system commit server queue

**Syntax**

```
show system commit server queue
<id commit-id>
<job-status (all| error| pending| success)>
<patch (none | id commit-id) | (job-status (all | error | pending | success)>
```

**Release Information**

Command introduced in Junos OS Release 12.1.

**Description**

Display the status of commit server queue transactions.

> **NOTE:** 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.

**Options**

**id *commit-id***—(Optional) Display the batch commit operation status messages for a specific commit ID.

**job-status**—(Optional) Display batch commit operation status messages for the following batch commit statuses:

- **all**—Status messages for all batch commit operations.
- **error**—Status messages for batch commit operations that have errors.
- **pending**—Status messages for batch commit operations that are pending.
- **success**—Status messages for batch commit operations that are successful.

**patch (none | id *commit-id*) | job-status (all |error | pending | success)**—(Optional) Display the patch file containing the configuration changes for all batch commit operations, a specific batch commit ID, or a specific job status.

**Required Privilege Level**

view

RELATED DOCUMENTATION

Example: Configuring Batch Commit Server Properties | 188

**List of Sample Output**

show system commit server queue on page 432
show system commit server queue job-status success on page 433
show system commit server queue patch on page 433

## 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 commiting 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 commiting 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.

> **NOTE:** 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.

**Options**

This command has no options.

**Required Privilege Level**

view

RELATED DOCUMENTATION

Example: Configuring Batch Commit Server Properties | 188

**List of Sample Output**

show system commit server status (When Server Is Inactive) on page 436
show system commit server status (When Server Is Active) on page 437

## 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
```

# show system configuration archival

**Syntax**

```
show system configuration archival
```

**Release Information**

Introduced in Junos OS Release 7.6.
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.

**Description**

Display directory and number of files queued for archival transfer.

> **NOTE:** The **[edit system configuration]** hierarchy is not available on QFabric systems.

**Options**

This command has no options.

**Required Privilege Level**

maintenance

**List of Sample Output**

show system configuration archival on page 438

## Sample Output

**show system configuration archival**

```
user@host> show system configuration archival

/var/transfer/config/:
total 8
```

# show system configuration rescue

**Syntax**

```
show system configuration rescue
```

**Release Information**

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 OCX Series switches.

**Description**

Display a rescue configuration, if one exists.

> **NOTE:** The **[edit system configuration]** hierarchy is not available on QFabric systems.

**Options**

This command has no options.

**Required Privilege Level**

maintenance

RELATED DOCUMENTATION

show system configuration archival | 438

**List of Sample Output**

**show system configuration rescue on page 439**

## Sample Output

**show system configuration rescue**

```
user@switch> show system configuration rescue
```

```
version "7.3"; groups {
    global {
        system {
            host-name router1;
            domain-name customer.net;
            domain-search [ customer.net ];
            backup-router 192.0.2.0;
            name-server {
                192.0.2.11;
                192.0.2.101;
                192.0.2.100;
                192.0.2.10;
            }
            login {
                user user1 {
                    uid 928;
                    class ;
                    shell csh;
                    authentication {
                        encrypted-password "$ABC123"; ## SECRET-DATA
                    }
                }
            }
            services {
                ftp;
                rlogin;
                rsh;
                telnet;
            }
        }
    }
....
```

# show system rollback

**Syntax**

```
show system rollback number
<compare number>
```

**Release Information**

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 14.1X53-D20 for OCX Series switches.
Command introduced in Junos OS Release 11.1 for the QFX Series.

**Description**

Display the contents of a previously committed configuration, or the differences between two previously committed configurations.

> **NOTE:** The **show system rollback** command is a purely operational mode command and cannot be issued with **run** from the configuration mode.

**Options**

***number***—Number of a configuration to view. The output displays the configuration. The range of values is 0 through 49.

**compare *number*** —(Optional) Number of another previously committed (rollback) configuration to compare to rollback ***number***. The output displays the differences between the two configurations. The range of values is 0 through 49.

**Required Privilege Level**

view

**List of Sample Output**

show system rollback compare on page 441

## Sample Output

**show system rollback compare**

```
user@host> show system rollback 3 compare 1
```

```
[edit]
+  interfaces {
+      ge-1/1/1 {
+          unit 0 {
+              family inet {
+                  filter {
+                      input mf_plp;
+                  }
+                  address 10.1.1.1/10;
+              }
+          }
+      }
+      ge-1/2/1 {
+          unit 0 {
+              family inet {
+                  filter {
+                      input mf_plp;
+                  }
+                  address 10.1.1.1/10;
+              }
+          }
+      }
+      ge-1/3/0 {
+          unit 0 {
+              family inet {
+                  filter {
+                      input mf_plp;
+                  }
+                  address 10.1.1.1/10;
+              }
+          }
+      }
+}
```

# status

**Syntax**

```
status
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Display the users currently editing the configuration.

**Options**

This command has no options.

**Required Privilege Level**

configure—Enters status configuration mode.

RELATED DOCUMENTATION

Displaying Users Currently Editing the Junos OS Configuration | 91

# test configuration

**Syntax**

```
test configuration filename
syntax-only
```

**Release Information**

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.
**syntax-only** option introduced in Junos OS Release 12.1.
Command introduced in Junos OS Release 14.1X53-D20 for OCX Series switches.

**Description**

Verify that the syntax of a configuration file is correct. If the configuration contains any syntax or commit check errors, a message is displayed to indicate the line number and column number in which the error was found. When using the *filename* option, this command only accepts text files.

**Options**

***filename***—Name of the configuration file. This file must be a text file and no other type.

**syntax-only**—(Optional) Check the syntax of a partial configuration file, without checking for commit errors.

**Required Privilege Level**

view

**List of Sample Output**

test configuration on page 444

**Output Fields**

When you enter this command, you are provided feedback on the status of your request.

## Sample Output

**test configuration**

```
user@host> test configuration terminal

[Type ^D to end input]
system {
host-name host;
```

```
test1;
login;
}
terminal:3:(8) syntax error: test
  [edit system]
    'test;'
      syntax error
terminal:4:(11) statement must contain additional statements: ;
  [edit system login]
    'login ;'
      statement must contain additional statements
configuration syntax failed
```

# top

**Syntax**

```
top <configuration-command>
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Return to the top level of configuration command mode, which is indicated by the **[edit]** banner.

**Options**

***configuration-command***—(Optional) Issue configuration mode commands from the top of the hierarchy.

**Required Privilege Level**

configure—To enter configuration mode.

RELATED DOCUMENTATION


Displaying the Current Junos OS Configuration | **165**

exit | **348**

up | **448**

# unprotect

**Syntax**

```
unprotect (hierarchy | statement | identifier)
```

**Release Information**

Command introduced in Junos OS Release 11.2.

**Description**

Unprotect a protected hierarchy, configuration statement, or an identifier, so that it can be modified or deleted.

**Options**

***hierarchy***—(Optional) Unprotect a specific protected hierarchy.

***statement***—(Optional) Unprotect a specific protected statement.

***identifier***—(Optional) Unprotect a specific protected identifier.

**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


protect | 359

top | 446

up | 448

Displaying the Current Junos OS Configuration | 165

# up

**Syntax**

```
up <number> <configuration-command>
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Move up one level in the statement hierarchy.

**Options**

**none**—Move up one level in the configuration hierarchy.

***configuration-command***—(Optional) Issue configuration mode commands from a location higher in the hierarchy.

***number***—(Optional) Move up the specified number of levels in the configuration hierarchy.

**Required Privilege Level**

configure—To enter configuration mode.

RELATED DOCUMENTATION


Displaying the Current Junos OS Configuration | **165**

exit | **348**

top | **446**

# 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.

**Options**

This command has no options.

**Required Privilege Level**

configure—To enter configuration mode.

RELATED DOCUMENTATION

Updating the configure private Configuration | 89.

# wildcard delete

**Syntax**

```
wildcard delete <statement-path> <identifier> <regular-expression>
```

**Release Information**

Command introduced before Junos OS Release 7.4.

**Description**

Delete a statement or identifier. All subordinate statements and identifiers contained within the specified statement path are deleted with it.

Deleting a statement or an identifier effectively “unconfigures” or disables the functionality associated with that statement or identifier.

If you do not specify *statement-path* or *identifier*, the entire hierarchy starting at the current hierarchy level is removed.

**Options**

***identifier***—(Optional) Name of the statement or identifier to delete.

***regular-expression***—(Optional) The pattern based on which you want to delete multiple items. When you use the **wildcard** command to delete related configuration items, the *regular-expression* must be the final statement.

***statement-path***—(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.

**Required Privilege Level**

configure—To enter configuration mode. Other required privilege levels depend on where the statement is located in the configuration hierarchy.

RELATED DOCUMENTATION

Example: Using Global Replace in a Junos OS Configuration—Using the upto Option | **125**.