

# Junos® OS

---

## CLI User Guide for Junos OS

Published  
2025-12-19

Juniper Networks, Inc.  
1133 Innovation Way  
Sunnyvale, California 94089  
USA  
408-745-2000  
[www.juniper.net](http://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 for Junos OS*

Copyright © 2025 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 This Guide | xi

## 1

### Overview

About the CLI Guide | 2

CLI Overview | 2

Introducing the Command-Line Interface | 3

CLI Modes, Commands, and Statement Hierarchies—An Overview | 5

Other Tools to Configure and Monitor Juniper Networks Devices | 7

Configure Junos OS in a FIPS Environment | 7

## 2

### Getting Started

Getting Started: A Quick Tour of the CLI | 10

Get Started with the Command-Line Interface | 10

Switch Between Operational Mode and Configuration Mode | 12

Use Keyboard Sequences to Navigate and Edit the CLI | 14

Configure a User Account on a Juniper Networks Device | 16

Use the CLI Editor in Configuration Mode | 19

Check the Status of a Juniper Networks Device | 22

Roll Back Configuration Changes | 25

Configure a Routing Protocol | 27

Shortcut | 28

Longer Configuration | 28

Make Changes to a Routing Protocol Configuration | 31

Online Help in the CLI | 35

Get Online Help from the Command-Line Interface | 35

CLI Online Help Features | 39

CLI Explorer Overview | 41

## CLI Environment Settings | 42

Customize the CLI Environment | 42

Set the CLI Screen Length and Width | 46

Enable Configuration Breadcrumbs | 47

## Using Configuration Statements to Configure a Device

### CLI Configuration Mode Overview | 51

Understanding CLI Configuration Mode | 51

Enter and Exit CLI Configuration Mode | 58

Relative Configuration Mode Commands | 60

Command Completion in Configuration Mode | 61

Notational Conventions Used in Configuration Hierarchies | 64

### Overview of the `configure` Command | 65

Forms of the `configure` Command | 65

How to Use the `configure` Command | 67

How to Use the `configure exclusive` Command | 68

How to Work with the Correct Configuration | 71

### Modify the Configuration of a Device | 71

Display Users Currently Editing the Configuration | 72

How to Modify the Juniper Networks Device Configuration | 73

How to Add Configuration Statements and Identifiers | 74

How to Delete a Statement from a Device Configuration | 76

Example: Delete a Statement from the Device Configuration | 79

Copy a Statement in the Configuration | 81

Example: Copy a Statement in the Configuration | 81

Requirements | 81

Overview | 82

Configuration | 82

## Example: Replace a Configuration | 85

- Requirements | 85

- Overview | 85

- Configuration | 86

## Insert a New Identifier in a Device Configuration | 92

### Example: Insert a New Identifier in a Device Configuration | 92

- Requirements | 93

- Overview | 94

- Configuration | 94

## Deactivate and Reactivate Statements and Identifiers in a Device Configuration | 97

### Example: Deactivate and Reactivate Statements and Identifiers in a Device Configuration | 98

- Requirements | 98

- Overview | 98

- Configuration | 98

## How to Make Global Changes in the Device Configuration | 100

### Common Regular Expressions to Use with the replace Command | 101

### Example: How to Use Global Replace in a Device Configuration—the \n Back Reference | 103

- Requirements | 103

- Overview | 104

- Configuration | 105

### Example: Global Replace in a Device Configuration—Replacing an Interface Name | 106

- Requirements | 107

- Overview | 107

- Configuration | 107

### Example: Global Replace in a Device Configuration—the upto Option | 109

## Add Comments in a Device Configuration | 112

- Add Comments in the CLI | 112

- Add Comments in a File | 113

### Example: Include Comments in a Device Configuration by Using the CLI | 114

- Requirements | 115

- Overview | 115
- Configuration | 115

Example: Use the Wildcard Command with the Range Option | 117

- Requirements | 118
- Overview | 118
- Configuration | 118

## Use Configuration Groups to Quickly Configure Devices | 127

Configuration Groups Overview | 128

Configure Configuration Groups | 129

Create a Configuration Group | 129

How to Apply a Configuration Group | 131

Example: Create and Apply Configuration Groups | 133

Example: Disable Inheritance of a Configuration Group | 134

Example: Use the `junos-defaults` Configuration Group | 136

Example: Use Wildcards with Configuration Groups | 139

How to Improve Commit Time When Using Configuration Groups | 142

Example: Configure Sets of Statements with Configuration Groups | 143

Example: Configure Interfaces Using Configuration Groups | 144

Example: Use Configuration Groups to Configure a Consistent IP Address for the Management Interface | 147

Example: Use Configuration Groups to Configure Peer Entities | 149

Example: Use Configuration Groups to Establish Regional Configurations | 151

Example: Configure Wildcard Configuration Group Names | 152

Example: Reference the Preset Statement from the Defaults Group | 154

Example: View Default Statements That Have Been Applied to the Configuration | 155

Set Up Routing Engine Configuration Groups | 156

How to Use Conditions to Apply Configuration Groups | 158

Example: Configure Conditions for Applying Configuration Groups | 159

Requirements | 159

Overview | 159

Configuration | 160

## **View the Configuration | 163**

Display the Current Configuration | 163

Example: Display the Current Configuration | 164

Display Additional Information About the Configuration | 166

Display set Commands from the Configuration | 169

## **Verify the Device Configuration | 173**

### **Commit the Configuration | 174**

The Commit Model for Configurations | 174

Commit a Device Configuration | 176

Commit Operation When Multiple Users Configure the Software | 178

Commit Preparation and Activation Overview | 178

Commit Device Configurations in Two Steps: Preparation and Activation | 180

Activate a Device Configuration with Confirmation | 182

Schedule a Commit Operation | 184

Monitor the Commit Process | 185

Add a Comment to Describe the Committed Configuration | 186

Batch Commits Overview | 188

Example: Configure Batch Commit Server Properties | 189

Requirements | 189

Overview | 189

Configuration | 189

Verification | 193

Back Up the Committed Configuration on the Alternate Boot Drive | 200

## **Managing Configurations**

### **Configuration Files Overview | 202**

Configuration Files Overview | 202

Device Configuration Storage Overview | 204

## Managing Configurations | 204

The show | compare | display xml Command Output | 205

Returning to the Most Recently Committed Configuration | 214

Returning to a Previously Committed Configuration | 215

Example of Returning to a Previous Configuration | 215

Example of Displaying Previous Configurations | 215

About Comparing Configuration Versions | 217

Using Configuration Revision Identifiers | 219

Saving a Configuration to a File | 221

About Compressing the Current Configuration File | 222

Free Up System Storage Space | 223

Clean Up Files with the CLI | 225

## Autoinstallation of Configuration Files Overview | 227

Configuration File Autoinstallation—An Overview | 227

Configuring Autoinstallation of Configuration Files (CLI Procedure) | 230

## Loading Configuration Files | 232

Examples for Loading a Configuration from a File or the Terminal | 233

How Character Encoding Works on Juniper Networks Devices | 236

About Specifying Statements and Identifiers | 237

About Loading a Configuration from a File | 242

Upload a Configuration File | 246

Load JSON Configuration Data With Unordered List Entries | 247

## Back Up Configurations to an Archive Site | 250

Configure the Transfer of the Active Configuration | 250

## Factory Default Configuration Overview | 253



Restore the Default Factory Configuration | 253

## Rescue Configuration | 254

Creating and Returning to a Rescue Configuration | 254

## Encrypt and Decrypt Configuration Files | 255

Encrypt Configuration Files | 256

Decrypt Configuration Files | 258

Modify the Encryption Key | 259

## Example: Protecting the Junos OS Configuration from Modification or Deletion | 259

Requirements | 260

Overview | 260

Protecting a Parent-Level Hierarchy | 261

Protecting a Child Hierarchy | 262

Protecting a Configuration Statement Within a Hierarchy | 263

Protecting a List of Identifiers for a Configuration Statement | 263

Protecting an Individual Member from a Homogenous List | 264

Unprotecting a Configuration | 265

Verification | 266

## Synchronizing Configurations Across Routing Engines | 271

Routing Engine Synchronization Overview | 271

Configure Multiple Routing Engines to Synchronize Committed Configurations Automatically | 275

# 5

## Using Operational Commands to Monitor Devices

### CLI Operational Mode Overview | 279

CLI Operational Mode Command Overview | 279

Display Options of the show Command—An Overview | 282

Interface Naming Conventions Used in Operational Commands | 283

About Group Interface Names Using Wildcard Characters | 285

### Using Operational Commands to Monitor a Device | 286

CLI Command Completion Example | 287

Operational Mode Commands: Overview of Controlling the Scope | 288

Viewing Files and Directories on a Device | 290

Directories on the Device | 290

List Files and Directories | 291

Filenames and URLs | 294

Display Operating System Information | 296

Managing Programs and Processes Using Operational Mode Commands | 296

Show Software Processes | 296

Restart the Software Process | 298

Stop the Software | 299

Reboot the Software | 300

CLI Comment Character # for Operational Mode Commands | 301

## Filtering Operational Command Output | 301

About Using the Pipe ( | ) Symbol to Filter Command Output | 302

Example: Use Regular Expressions with the Pipe ( | ) Symbol to Filter Command Output | 303

Example: Pipe ( | ) Filter Functions in the Command-Line Interface | 305

Filter Operational Mode Command Output in a QFabric System | 319

Use Suppress-Zero Filter with the Pipe ( | ) Symbol to Filter Zero Values in Command Output | 320

## Configuration Statements and Operational Commands

Junos CLI Reference Overview | 325

# About This Guide

The Junos OS command-line interface (CLI) is a command shell specific to Juniper Networks. This command shell runs on top of the FreeBSD UNIX-based operating system kernel for Junos OS. Using industry-standard tools and utilities, the CLI provides a powerful set of commands that you can use to monitor and configure Juniper Networks devices running Junos OS. This guide contains information about the CLI for Junos OS.

## RELATED DOCUMENTATION

---

[Day One: Exploring the Junos CLI](#)

[Day One: Junos for IOS Engineers](#)

# 1

CHAPTER

## Overview

---

### IN THIS CHAPTER

- [About the CLI Guide | 2](#)
  - [CLI Overview | 2](#)
-

# About the CLI Guide

The Junos OS CLI Guide explains how to use the command-line interface (CLI). This guide also describes advanced concepts and device configuration when working with Juniper Networks devices running Junos OS.

In this guide, you will learn about:

- 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 and describes the security, configuration, monitoring, and management of network devices.

Another useful learning resource is [Day One: Exploring the Junos CLI](#).

## CLI Overview

### IN THIS SECTION

- [Introducing the Command-Line Interface | 3](#)
- [CLI Modes, Commands, and Statement Hierarchies—An Overview | 5](#)
- [Other Tools to Configure and Monitor Juniper Networks Devices | 7](#)
- [Configure Junos OS in a FIPS Environment | 7](#)

The CLI is the software interface used to access your device. You use the CLI to configure the device, monitor its operations, and adjust the configuration as needed. You access the CLI through a console connection interface or through a network connection.

## Introducing the Command-Line Interface

### IN THIS SECTION

- [Key Features of the CLI | 3](#)

The Junos OS CLI is a command shell specific to Juniper Networks that runs on top of the operating system kernel. Through industry-standard tools and utilities, the CLI provides a powerful set of commands that you can use to monitor and to configure devices running Junos OS.

The CLI has two modes:

- **Operational mode**—Use this mode to display the current status of the device. In operational mode, you enter commands to monitor and to troubleshoot the network operating system, devices, and network connectivity.
- **Configuration mode**—Use this mode 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. Junos OS stores a configuration as a hierarchy of configuration statements.

When you enter configuration mode, you are viewing and changing a file called the *candidate configuration*. You use the candidate configuration file, you make configuration changes without causing operational changes to the current operating configuration, called the *active configuration*. The device does not implement the changes you added to the candidate configuration file until you commit the changes. Committing the configuration changes activates the revised configuration on the device. Candidate configurations enable you to alter your configuration without damaging your current network operations.

### Key Features of the CLI

The CLI commands and statements follow a hierarchical organization and have a regular syntax. The 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 you 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—The CLI provides information about available commands at each level of the 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 are already familiar with 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 CLI displays possible completions. 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 operational mode or configuration mode, the CLI displays `<[Enter]>` as one of the choices when you type a question mark (?). This output 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 reach a specific hierarchy level in the configuration mode and do not need 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 CLI. For example, you can:
  - Use regular expression matching to locate and to replace values and identifiers in a configuration, to filter command output, and to 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 the CLI environment and create a UNIX C shell or Bourne shell to navigate the file system, manage router processes, and so on.

## CLI Modes, Commands, and Statement Hierarchies—An Overview

### IN THIS SECTION

- [CLI Command Hierarchy | 5](#)
- [Configuration Statement Hierarchy | 5](#)
- [Move Among Hierarchy Levels | 6](#)

The Junos OS CLI commands and statements are organized under two command modes and various hierarchies. The following sections provide an overview of the CLI command modes and the command and statement 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 under the `show system` command. All commands that display information about the routing table are 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`.

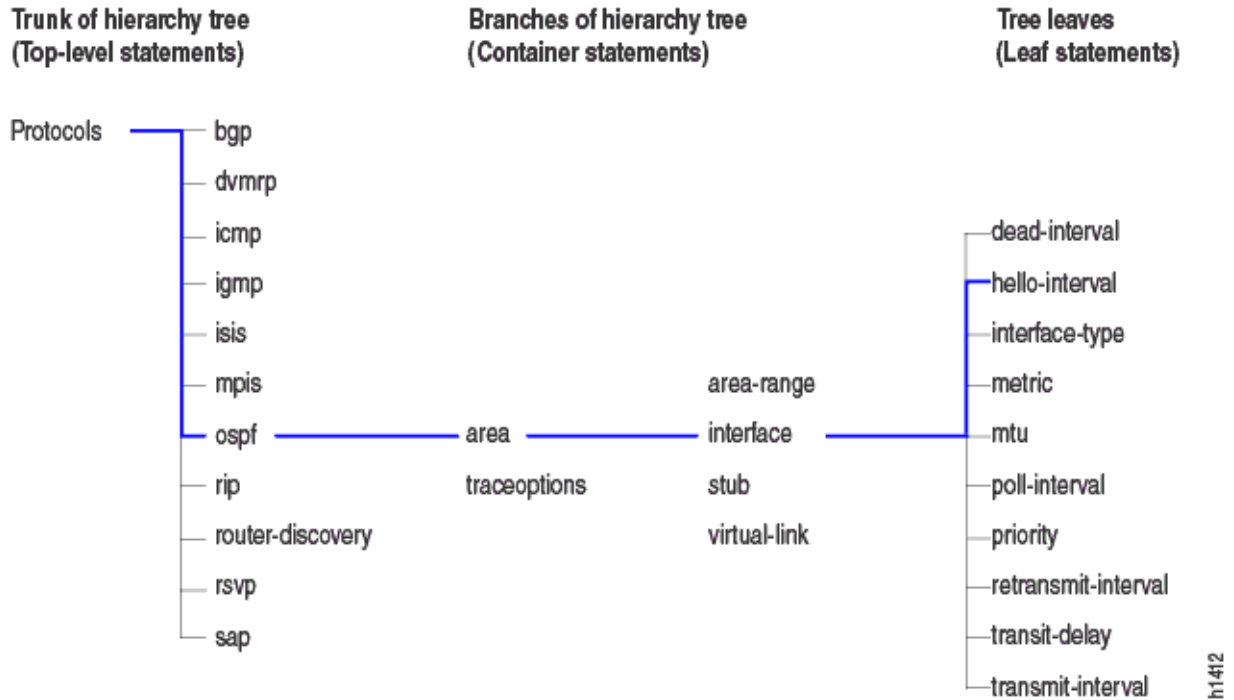
### Configuration Statement Hierarchy

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 statements 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; that is, they are branches of the hierarchy tree. The `hello-interval` statement is a leaf on the tree.

**Figure 1: Configuration Statement Hierarchy Example**





## Move Among Hierarchy Levels

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

**Table 1: CLI Configuration Mode Navigation Commands**

Command	Description
<code>edit <i>hierarchy-level</i></code>	Moves to an existing configuration statement hierarchy or creates a hierarchy and moves to that level.
<code>exit</code>	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 command and the quit command are interchangeable.
<code>up</code>	Moves up the hierarchy one level at a time.
<code>top</code>	Moves directly to the top level of the hierarchy.

## Other Tools to Configure and Monitor Juniper Networks Devices

Apart from the CLI, Junos OS also supports the following applications, scripts, and utilities that enable you to configure and monitor Juniper Networks devices:

- **J-Web GUI**—Available on select Juniper Networks devices, the J-Web GUI enables you to monitor, configure, troubleshoot, and manage the device by means of a browser with HTTP or HTTPS enabled. For more information, see the *J-Web Interface User Guide*.
- **Junos XML management protocol**—The Junos XML management protocol enables you to monitor and configure Juniper Networks devices. For more information, see the *Junos XML Management Protocol Developer Guide*.
- **NETCONF API**—You can also use the NETCONF XML management protocol to monitor and configure Juniper Networks devices. For more information, see the [NETCONF XML Management Protocol Developer Guide](#).
- **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 [Junos OS Automation Scripting User Guide](#).
- **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 [Junos OS Network Management Administration Guide for Routing Devices](#).

## Configure Junos OS in a FIPS Environment

With Junos-FIPS you can configure a network of Juniper Networks devices in a 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 Juniper Networks devices so the secure environment complies with the requirements of public sector certifications such as Common Criteria 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](#)

# 2

CHAPTER

## Getting Started

---

### IN THIS CHAPTER

- [Getting Started: A Quick Tour of the CLI | 10](#)
  - [Online Help in the CLI | 35](#)
  - [CLI Environment Settings | 42](#)
-

# Getting Started: A Quick Tour of the CLI

## IN THIS SECTION

- [Get Started with the Command-Line Interface | 10](#)
- [Switch Between Operational Mode and Configuration Mode | 12](#)
- [Use Keyboard Sequences to Navigate and Edit the CLI | 14](#)
- [Configure a User Account on a Juniper Networks Device | 16](#)
- [Use the CLI Editor in Configuration Mode | 19](#)
- [Check the Status of a Juniper Networks Device | 22](#)
- [Roll Back Configuration Changes | 25](#)
- [Configure a Routing Protocol | 27](#)

The following topics can help you (the network administrator) 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 assumes that you are familiar with Junos OS concepts and operation principles.

## Get Started with the Command-Line Interface

This topic shows you how to start the Junos OS CLI, view the command hierarchy, and make minor configuration changes.



**NOTE:** Before you begin, make sure that 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 that 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. 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.

CLI commands can vary by platform and software release. The CLI includes several ways to get help about available 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

```

## Switch Between Operational Mode and Configuration Mode

When you monitor and configure a device running Junos OS, you may need to switch between modes . When you switch between operational mode and configuration mode, the command prompt also

changes. The operational mode prompt is a right-angle bracket (>). 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 and press Enter, 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 in CLI operational mode. The prompt in brackets (`[edit]`), also known as a *banner*, shows that you are in configuration 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#`, showing that you are in configuration mode, and a banner appears to indicate the hierarchy level.

3. You can exit configuration mode and 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. 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 ...
```

## Use Keyboard Sequences to Navigate and Edit the CLI

You can use keyboard sequences in the Junos OS CLI to navigate 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 2: 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.

Table 2: CLI Keyboard Shortcuts (*Continued*)

Keyboard sequence	Action
Ctrl+n	Scroll forward through the list of recently executed commands.
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+ <i>number sequence</i> or Alt+ <i>number sequence</i>	Specify the number of times to execute a keyboard sequence.

## Configure a User Account on a Juniper Networks Device

This topic describes how to use a root account to log in to a Juniper Networks device 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. In the example, `user1` represents a username:

```
[edit system login]
root@host# edit user user1
```

This example adds an account `user1`.



**NOTE:** User account names can contain a period (.). For example, you can have a user account `user.1`. 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 user1]
root@host# set full-name "User One"
```

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

```
[edit system login user user1]
root@host# set class super-user
```

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

```
[edit system login user user1]
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 user1]
root@host# commit
commit complete
```

Configuration changes are not active 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 user1]
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: user1
Password: password
---JUNOS 17.2B1.8 built 2018-05-09 23:41:29 UTC
user1@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.



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

## Use the CLI Editor in Configuration Mode

This topic describes basic commands that you can use to enter configuration mode in the CLI editor. The topic also describes commands that you use to navigate the configuration hierarchy, get help, and commit or revert the changes that you make during the configuration session.

Task	Command/ Statement	Example
<b>Edit Your Configuration</b>		
<p>Enter configuration mode.</p> <p>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 <code>user@host&gt;</code> to <code>user@host#</code>, and the hierarchy level appears in square brackets.</p>	<code>configure</code>	<pre>user@host&gt; configure  [edit] user@host#</pre>
<p>Create a statement hierarchy.</p> <p>You can use the <code>edit</code> command to simultaneously create a hierarchy and move to that new level in the hierarchy. You cannot use the <code>edit</code> command to change the value of identifiers.</p>	<code>edit <i>hierarchy-level value</i></code>	<pre>[edit] user@host# edit security zones security-zone myzone  [edit security zones security-zone myzone] user@host#</pre>
<p>Create a statement hierarchy, and set identifier values.</p> <p>The <code>set</code> command is like <code>edit</code>, except that your current level in the hierarchy does not change.</p>	<code>set <i>hierarchy-level value</i></code>	<pre>[edit] user@host# set security zones security-zone myzone  [edit] user@host#</pre>
<b>Navigate the Hierarchy</b>		

*(Continued)*

Task	Command/ Statement	Example
Navigate down to an existing hierarchy level.	<code>edit hierarchy-level</code>	<pre>[edit] user@host# edit security zones  [edit security zones] user@host#</pre>
Navigate up one level in the hierarchy.	<code>up</code>	<pre>[edit security zones] user@host# up  [edit security] user@host#</pre>
Navigate to the top of the hierarchy.	<code>top</code>	<pre>[edit security zones] user@host# top  [edit] user@host#</pre>
<b>Commit or Revert Changes</b>		
Commit your configuration.	<code>commit</code>	<pre>[edit] user@host# commit  commit complete</pre>

*(Continued)*

Task	Command/ Statement	Example
<p>Roll changes back from the current session.</p> <p>Use the rollback command to revert all changes from the current configuration session. When you run the rollback command before you exit your session or commit changes, the software loads the most recently committed configuration onto the device. You must enter the rollback statement at the edit level in the hierarchy.</p>	rollback	<pre>[edit] user@host# rollback  load complete</pre>
<b>Exit Configuration Mode</b>		
Commit the configuration, and exit configuration mode.	commit and-quit	<pre>[edit] user@host# <b>commit and-quit</b>  user@host&gt;</pre>
<p>Exit configuration mode without committing your configuration.</p> <p>You must navigate to the top of the hierarchy using the up or top commands before you can exit configuration mode.</p>	exit	<pre>[edit] user@host# exit  The configuration has been changed but not committed Exit with uncommitted changes? [yes,no] (yes)</pre>
<b>Get Help</b>		



(Continued)

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

SEE ALSO

| [Understanding CLI Configuration Mode](#) | 51

Check the Status of a Juniper Networks Device

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 Operational Mode Commands | 296](#)

[Viewing Files and Directories on a Device | 290](#)

## Roll Back 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 Juniper Networks device 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 Configuration | 214](#)

# Configure a Routing Protocol

## IN THIS SECTION

- [Shortcut | 28](#)
- [Longer Configuration | 28](#)
- [Make Changes to a Routing Protocol Configuration | 31](#)

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 so-0/0/0
[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 go 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 that 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#

```

## Make Changes to a Routing Protocol Configuration

Suppose you decide to use different dead intervals 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 that 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. Maybe you decide to use the default values for the hello intervals and dead intervals on your remaining interface but want OSPF to run on that interface. In that case, delete the hello interval timer and dead interval timer:

```
[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 hierarchy consists of the path of statements from the top inward, as well as one or more statements at the bottom of the hierarchy. Setting multiple statements at the same time can reduce considerably the number of commands you must enter.

4. To go back to the original hello interval timer and dead interval timer 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 re-create 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-0/0/0 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#
```

# Online Help in the CLI

## IN THIS SECTION

- [Get Online Help from the Command-Line Interface | 35](#)
- [CLI Online Help Features | 39](#)
- [CLI Explorer Overview | 41](#)

## Get Online Help from the Command-Line Interface

### IN THIS SECTION

- [Getting Help About Commands | 35](#)
- [Getting Help About a String in a Statement or Command | 37](#)
- [Getting Help About Configuration Statements | 38](#)
- [Getting Help About System Log Messages | 38](#)

The CLI has a context-sensitive online help feature that enables you to access information about commands and statements.

### Getting Help About Commands

CLI commands and options can vary by platform and software release. Each level of the CLI command hierarchy provides information about available commands. 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. Use the string 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 that need input data type as STRING, the supported characters set is as follows:

- Any printable ASCII characters
- For characters with space, enclose it 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.

## CLI Online Help Features

### IN THIS SECTION

- [Help for Omitted Statements | 39](#)
- [Using CLI Command Completion | 39](#)
- [Using Command Completion in Configuration Mode | 40](#)
- [Displaying Tips About CLI Commands | 40](#)

### 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 the operating system 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 display.

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

[CLI Command Completion Example](#) | 287

## CLI Explorer Overview

CLI Explorer is a Web application that helps you to explore Junos OS configuration statements and commands. CLI Explorer lists all the configuration statements and commands the Junos OS supports across different platforms and software releases.

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](#).

# CLI Environment Settings

## IN THIS SECTION

- [Customize the CLI Environment | 42](#)
- [Set the CLI Screen Length and Width | 46](#)
- [Enable Configuration Breadcrumbs | 47](#)

In operational mode, you (the network administrator) can customize the Junos OS CLI environment to suit your specific preferences and requirements.

## Customize the CLI Environment

### IN THIS SECTION

- [Display the Current CLI Settings | 43](#)
- [Set the Terminal Type | 43](#)
- [Set the CLI Prompt | 44](#)
- [Set the CLI Directory | 44](#)
- [Set the CLI Timestamp | 44](#)
- [Set the Idle Timeout | 44](#)
- [Set the CLI to Prompt for Restart After a Software Upgrade | 45](#)
- [Set CLI Command Completion | 45](#)

In operational mode, you can customize the 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
  tenant            Set default tenant
  terminal           Set terminal type
  timestamp          Timestamp CLI output
```



**NOTE:** Some values are already set 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.

## Display 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
```

## Set 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`.

## Set 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 to `user@host`.

## Set the CLI Directory

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

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

The *directory* must be the full pathname of the desired working directory. After entering this command, the CLI switches to the specified directory.

## Set 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).

## Set the Idle Timeout

By default, a CLI session never times out after extended idle time unless you have included the `idle-timeout` statement 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.

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

By default, the CLI prompts users 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
```

## Set 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>
```



## Set the CLI Screen Length and Width

### IN THIS SECTION

- [Set the Screen Length | 46](#)
- [Set the Screen Width | 46](#)

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

### Set the Screen Length

The default CLI screen length is 24 lines. If output is longer than this, the display scrolls to the configured screen length and then displays a `more` prompt. You can press Enter to display the next line, or press the Spacebar to show the next full screen. Alternatively, you can press `h` to view 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. This setting causes the screen to 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.

### Set 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
```

## Enable Configuration Breadcrumbs

You can configure the output of `show configuration` operational mode commands and `show configuration` mode commands to display configuration breadcrumbs. These breadcrumbs help you identify the exact location in the configuration hierarchy for the output you are viewing.

Before you enable 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%)-----
```

The output does not clearly indicate the section of the configuration being viewed.

To enable the configuration breadcrumbs feature:

1. Launch configuration mode in the CLI.
2. Define a class at the `[edit system login]` hierarchy level, and set 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 breadcrumb 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
```

Upon 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 the user is viewing. In this case, user1 is currently viewing the interface configuration of a group.



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

# 3

CHAPTER

## Using Configuration Statements to Configure a Device

---

### IN THIS CHAPTER

- CLI Configuration Mode Overview | 51
  - Overview of the Configure Command | 65
  - Modify the Configuration of a Device | 71
  - Use Configuration Groups to Quickly Configure Devices | 127
  - View the Configuration | 163
  - Verify the Device Configuration | 173
  - Commit the Configuration | 174
-

# CLI Configuration Mode Overview

## IN THIS SECTION

- [Understanding CLI Configuration Mode | 51](#)
- [Enter and Exit CLI Configuration Mode | 58](#)
- [Relative Configuration Mode Commands | 60](#)
- [Command Completion in Configuration Mode | 61](#)
- [Notational Conventions Used in Configuration Hierarchies | 64](#)

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 CLI Configuration Mode

### IN THIS SECTION

- [Configuration Mode Commands | 52](#)
- [Configuration Statements and Identifiers | 54](#)
- [Configuration Statement Hierarchy | 55](#)

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 "[Understanding the Junos OS CLI Modes, Commands, and Statement Hierarchies](#)" on page 5 describes, a device configuration is stored as a hierarchy of statements. In configuration mode, you create a set of configuration statements to use. When you finish entering the configuration statements and 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, load it on the device, and commit it.

## Configuration Mode Commands

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

**Table 3: Summary of Configuration Mode Commands**

Command	Description
activate	Remove the <code>inactive:</code> tag from a statement. Statements or identifiers that have been activated take effect when you next issue the <code>commit</code> 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 <code>inactive:</code> 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 <code>commit</code> 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 before the last edit command, or exit from configuration mode. The <code>quit</code> and <code>exit</code> commands are equivalent.
extension	Manage configurations that SDK application packages contribute. Manage them by either displaying or deleting user-defined configurations that the named SDK application package contributed. A configuration defined in any native Junos OS package is never deleted by the extension command.

**Table 3: Summary of Configuration Mode Commands (Continued)**

Command	Description
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.
quit	Exit the current level of the statement hierarchy, returning to the level before the last edit command, or exit from configuration mode. The quit and exit commands are equivalent.
rename	Rename an existing <i>configuration statement</i> 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 action 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 command 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.



**Table 3: Summary of Configuration Mode Commands (Continued)**

Command	Description
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, the operating system 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 4 on page 54](#) lists top-level configuration statements. See [CLI Explorer](#) for information about each configuration statement.

**Table 4: Configuration Mode Top-Level Statements**

Statement	Description
access	Configure the Challenge Handshake Authentication Protocol (CHAP).
accounting-options	Configure accounting statistics data collection for interfaces and firewall filters.
chassis	Configure properties of the router chassis, including conditions that activate alarms and SONET/SDH framing and concatenation properties.

**Table 4: Configuration Mode Top-Level Statements** *(Continued)*

Statement	Description
class-of-service	Configure class-of-service parameters.
firewall	Configure filters that select packets based on their contents.
forwarding-options	Configure forwarding options, including traffic sampling options.
groups	Configure configuration groups.
interfaces	Configure interface information, such as encapsulation, interfaces, virtual channel identifiers (VCIs), and data-link connection identifiers (DLCIs).
policy-options	Configure routing policies, which enable you to filter and set properties in incoming and outgoing routes.
protocols	Configure routing protocols, including BGP, IS-IS, LDP, MPLS, OSPF, RIP, and RSVP.
routing-instances	Configure one or more routing instances.
routing-options	Configure protocol-independent routing options, such as static routes, autonomous system numbers, confederation members, and global tracing (debugging) operations to log.
security	Configure IP Security (IPsec) services.
snmp	Configure SNMP community strings, interfaces, traps, and notifications.
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.

## Configuration Statement Hierarchy

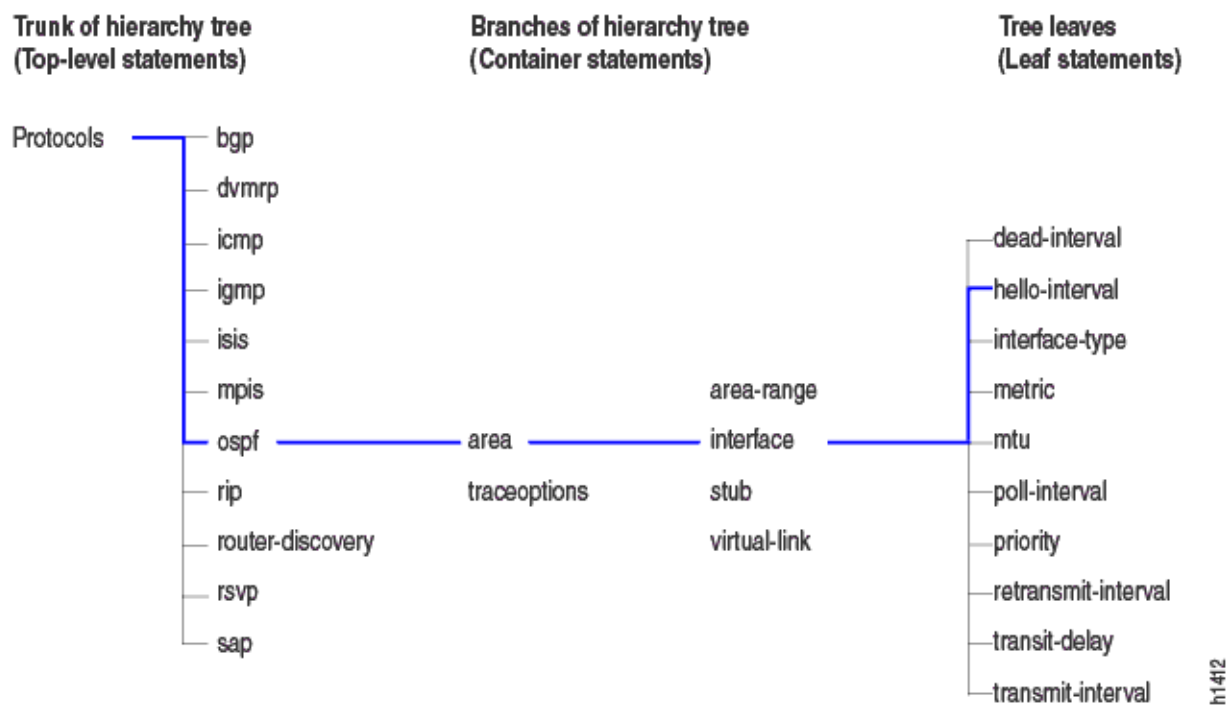
The Junos OS configuration consists of a hierarchy of statements. There are two types of statements:

- 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 56](#):

[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 configuration mode:

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

## Enter and Exit 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, commit them before you exit. If you exit configuration mode without committing configuration changes, you lose the intended changes.

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 fconfiguration mode using the `exit` command and the configuration contains changes that you have not committed, you see the following 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

[Switch Between Operational Mode and Configuration Mode | 12](#)

[How to Use the `configure exclusive` Command | 68](#)

[How to Work with the Correct Configuration | 71](#)

[How to Modify the Juniper Networks Device Configuration | 73](#)

[Display set Commands from the Configuration | 169](#)

[Commit Operation When Multiple Users Configure the Software | 178](#)

[Managing Programs and Processes Using Operational Mode Commands | 296](#)

## Relative 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 in the hierarchy, and then specify a configuration command. For example:

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

## SEE ALSO

| [Display the Current Configuration](#) | 163

## Command Completion in Configuration Mode

This topic shows you how to access command help and to use basic command completion in CLI configuration mode. In each case, you access help by using the question mark (?) character, either alone or with a partial command or configuration statement.

To list the configuration mode commands, use the `?` command alone:

```
[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, see 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, letter, or both, and then enter the ? character. This example shows 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 example shows all configured xe- interfaces. You can display these interfaces by using the first two letters of the abbreviation (ex) and the ? character:

```

user@host# edit interfaces ex?
<interface_name>      Interface name
ex-0/2/0              Interface name
ex-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

[How to Add Configuration Statements and Identifiers](#) | 74

## Notational Conventions Used in Configuration Hierarchies

When you are working in 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 a 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. An exception occurs if a subhierarchy is so long that it might be difficult to determine where it ends and its next peer statement begins. In case of a very long subhierarchy, the subhierarchy appears at the end of its parent hierarchy instead of in alphabetical order. In this exception scenario, a placeholder appears in the alphabetical position where the subhierarchy would have been listed.

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 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.

## Overview of the Configure Command

### IN THIS SECTION

- [Forms of the configure Command | 65](#)
- [How to Use the configure Command | 67](#)
- [How to Use the configure exclusive Command | 68](#)
- [How to Work with the Correct Configuration | 71](#)

You (the network administrator) use the `configure` command to enter CLI configuration mode. You can also use it to gather other information, such as which other users are currently in configuration mode.

### Forms of the configure Command

Junos OS supports three forms of the `configure` command: `configure`, `configure private`, and `configure exclusive`. These forms control how users edit and commit configurations. You can use this command to coordinate the work of multiple users who manage the network and device configuration.

**Table 5: Forms of the configure Command**

Command	Edit Access	Commit Access
configure	<ul style="list-style-type: none"> <li>• No one can lock the configuration. All users can make configuration changes.</li> <li>• When you enter configuration mode, the CLI displays the following information: <ul style="list-style-type: none"> <li>• A list of other users editing the configuration</li> <li>• Hierarchy levels the users are viewing or editing</li> <li>• Whether the configuration has been changed, but not committed</li> </ul> </li> <li>• When more than one user makes changes to a configuration, the most recent changes take precedence when the configuration is committed.</li> </ul>	<ul style="list-style-type: none"> <li>• All users can commit any changes to the configuration.</li> <li>• If you and another user make changes and the other user commits changes, your changes are committed as well.</li> </ul>
configure exclusive	<ul style="list-style-type: none"> <li>• One user locks the configuration and makes changes without interference from other users.</li> <li>• 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.</li> <li>• 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 <a href="#">CLI Explorer</a>.</li> </ul>	<ul style="list-style-type: none"> <li>• Only the user who has locked the configuration can commit it.</li> <li>• 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.</li> </ul>

**Table 5: Forms of the configure Command (Continued)**

Command	Edit Access	Commit Access
configure private	<ul style="list-style-type: none"> <li>• Multiple users can edit the configuration at the same time.</li> <li>• Each user has a private candidate configuration to edit independently of other users.</li> <li>• When multiple users enter conflicting configurations, the first commit operation takes precedence over subsequent commit operations.</li> </ul>	<ul style="list-style-type: none"> <li>• When you commit the configuration, the device does not immediately accept your private candidate configuration as the new operational configuration. Before the device accepts your configuration, it verifies that no other user has modified the operational (running) configuration .</li> <li>• If another user has modified the configuration, you can merge the modifications into your private candidate configuration and attempt to commit again.</li> </ul>

**SEE ALSO**


---

[Commit a Device Configuration | 176](#)


---

[Display set Commands from the Configuration | 169](#)


---

[Display Users Currently Editing the Configuration | 72](#)


---

## How to Use the configure Command

Up to 32 users can work in configuration mode simultaneously; 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 that you and other users have made. For this reason, you must keep track of who is in configuration mode with you.

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

- Use the configure command to enter CLI configuration mode.

If other users are in configuration mode, 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#
```

## How to Use the `configure exclusive` Command

If you enter configuration mode using the `configure exclusive` command, you lock the candidate global configuration for as long as you remain in configuration mode. (The candidate global configuration is also known as the shared configuration or shared configuration database.) Using the `configure exclusive` command, you can 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. Also, any attempted changes by other users while the configuration is in the locked state are discarded as soon as the other users 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 while another user is also in configuration mode and has locked the configuration, a message identifies the user. The message also identifies the portion of the configuration that the user is viewing or editing. For example, in the following example, 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 back the configuration to its previously committed state. The `no` option enables you to continue editing or to commit your changes in `configure exclusive` mode.

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

Another rollback can happen if you enter configuration mode with the `configure exclusive` command and issue the `commit confirmed` command, but without confirming the commit within the specified interval. By not confirming the commit within the specified interval, you trigger an automatic rollback. After an automatic rollback occurs, the operating system removes the exclusive lock from your session. As a result, the error message “access has been revoked” appears. This error message appears because the session is no longer an exclusive session. This means that the configuration is back to the default state: anyone with access can edit the configuration, commit it, or both. To re-lock the configuration, you must 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 the `commit confirmed` command, and confirm the commit, your session retains the exclusive lock. 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

| [How to Add Configuration Statements and Identifiers](#) | 74

## How to Work with the Correct 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 update your private candidate configuration, that 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 the `configure private` command to configure statements corresponding to third-party YANG data models such as OpenConfig data models or custom YANG data models.



**NOTE:** When you issue the `configure` command in a chassis cluster, it takes you to `configure private` mode.

## Modify the Configuration of a Device

### IN THIS SECTION

- [Display Users Currently Editing the Configuration | 72](#)
- [How to Modify the Juniper Networks Device Configuration | 73](#)
- [How to Add Configuration Statements and Identifiers | 74](#)
- [How to Delete a Statement from a Device Configuration | 76](#)
- [Example: Delete a Statement from the Device Configuration | 79](#)
- [Copy a Statement in the Configuration | 81](#)
- [Example: Copy a Statement in the Configuration | 81](#)

- [Example: Replace a Configuration | 85](#)
- [Insert a New Identifier in a Device Configuration | 92](#)
- [Example: Insert a New Identifier in a Device Configuration | 92](#)
- [Deactivate and Reactivate Statements and Identifiers in a Device Configuration | 97](#)
- [Example: Deactivate and Reactivate Statements and Identifiers in a Device Configuration | 98](#)
- [How to Make Global Changes in the Device Configuration | 100](#)
- [Common Regular Expressions to Use with the replace Command | 101](#)
- [Example: How to Use Global Replace in a Device Configuration—the \n Back Reference | 103](#)
- [Example: Global Replace in a Device Configuration—Replacing an Interface Name | 106](#)
- [Example: Global Replace in a Device Configuration—the upto Option | 109](#)
- [Add Comments in a Device Configuration | 112](#)
- [Example: Include Comments in a Device Configuration by Using the CLI | 114](#)
- [Example: Use the Wildcard Command with the Range Option | 117](#)

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

## Display Users Currently Editing the 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 | 65](#)

[Schedule a Commit Operation | 184](#)

[How to Use the configure Command | 67](#)

## How to Modify the Juniper Networks Device Configuration

To configure a Juniper Networks device or to modify an existing 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. You then continue creating the hierarchy with statements that move progressively lower in the hierarchy.

To modify the hierarchy, you use two configuration mode commands. Select the relevant command based on what you want to accomplish:

- **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>
```

The hierarchy to the configuration statement and the statement itself is *statement-path*. If you have already moved to the statement's hierarchy level, you can omit the statement path. The configuration statement itself is *statement*. The *identifier* string 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

[Relative Configuration Mode Commands | 60](#)

[How to Use the \*\*configure exclusive\*\* Command | 68](#)

[How to Work with the Correct Configuration | 71](#)

[Display the Current Configuration | 163](#)

## How to Add Configuration Statements and Identifiers

You configure all properties of a Juniper Networks device by including statements in the configuration. A statement consists of a keyword, which is fixed text. You can also include an identifier in a statement. An identifier is an identifying name that you define, such as the name of an interface or a username, and that enables you and the CLI to discriminate among a collection of statements.

For example, the following list shows the statements available at the top level in 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 identifiers in quotation marks (double quotes). You must also use quotation marks to enclose identifiers and any strings that include a space, a 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 must type an area number to complete the command:

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

## SEE ALSO

[Display the Current Configuration | 163](#)

[About Specifying Statements and Identifiers | 237](#)

## How to Delete a Statement from a Device Configuration

You delete a statement or identifier from a device configuration using 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, use the `delete` command without specifying a statement or an identifier. 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 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 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
```

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 delete only certain parts of the configuration where you normally put multiple items, such as interfaces. However, you cannot delete “groups” of different items, as shown in this 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, as follow:

```
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.

## How to Delete Interfaces from the Configuration

You can delete multiple T1 interfaces in the range from `t1-0/0/0:0` through `t1-0/0/0:23` by using this syntax:

```
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
```

## How to Delete Routes from the Configuration

You can delete static routes in the range from 172.0.0.0 to 172.255.0.0 by using this syntax:

```
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: Delete a Statement from the Device 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 example, remove 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;
}
```

## Copy a Statement in the Configuration

When you have many similar statements in a device 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:

1. Use the configuration mode `copy` command:

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

2. Immediately after you have copied a portion of the configuration, check the validity of the new configuration.
3. If the configuration is invalid, modify either the copied portion or the original portion to produce a valid configuration.

## Example: Copy a Statement in the Configuration

### IN THIS SECTION

- [Requirements | 81](#)
- [Overview | 82](#)
- [Configuration | 82](#)

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 them 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` in 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, you add a virtual connection that is very similar to a virtual connection already configured.

## Configuration

### IN THIS SECTION

- [CLI Quick Configuration | 83](#)
- [Configure by Copying | 83](#)
- [Results | 84](#)

## 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
```

## Configure 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: Replace a Configuration

### IN THIS SECTION

- [Requirements | 85](#)
- [Overview | 85](#)
- [Configuration | 86](#)

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 command 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 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 try some of the same commands under slightly different circumstances.

## Configuration

### IN THIS SECTION

- [Use the Copy, Rename, and Replace Commands to Modify a Loopback Interface Configuration | 86](#)
- [Compare the Copy Command at the Top-Level Configuration Hierarchy Level | 89](#)

## Use 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 lpbck interface";
```

3. Duplicate the loopback interface you have just created, errors included, 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 lpbck interface";
[edit]
user@host# show interfaces lo0 unit 101
description "this is a lpbck 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 unit 102 to verify that the word *loopback* is now spelled correctly.

```
[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. In configuration mode, use the `rollback` command to return the device configuration to the state it was in before you executed the previous steps.

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

## Results

In configuration mode, use the `show interfaces lo0 unit 101` and `show interfaces lo0 unit 102` commands to ensure that the device configuration is in 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 19](#).

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 occurrences, you must navigate 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. In configuration mode, use the `rollback` command to return the device configuration to the state it was in before you executed the previous steps.

```

[edit]
user@host# rollback

```

## Results

In configuration mode, use the `show interfaces et-2/0/0` and `show interfaces et-2/1/0` commands to ensure that the device configuration is in the state it was in before you executed the steps in this example.

```

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

```

You should not see any results from this command.

```

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

```

You should not see any results from this command.

## Insert a New Identifier in a Device Configuration

When configuring a Juniper Networks device, 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, in a few cases the order 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 configure the identifier, the identifier is placed at the end of the list of similar identifiers.

## Example: Insert a New Identifier in a Device Configuration

### IN THIS SECTION

- [Requirements | 93](#)
- [Overview | 94](#)
- [Configuration | 94](#)

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 statements is important in the cases in which the order 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 use it with 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 OS removes empty terms automatically.

## 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 that you have the hierarchy configured correctly:

```
[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

### IN THIS SECTION

- [CLI Quick Configuration | 94](#)
- [Configure to Insert Terms | 95](#)
- [Results | 96](#)

### 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` in 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
```

## Configure to Insert Terms

### Step-by-Step Procedure

1. Determine the order in which your configuration terms need to go. Consider 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;
}
```

## Deactivate and Reactivate Statements and Identifiers in a Device 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 the *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 function, it is reactivated when you issue a `commit` command but is treated as though it is down or administratively disabled.

## Example: Deactivate and Reactivate Statements and Identifiers in a Device Configuration

### IN THIS SECTION

- [Requirements | 98](#)
- [Overview | 98](#)
- [Configuration | 98](#)

This example shows a common use case in which you use the deactivate and activate configuration mode commands. It involves dual Routing Engines, primary 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 upgrade, you need to deactivate the GRES feature, upgrade each of the Routing Engines, and then activate GRES again.

### Configuration

#### IN THIS SECTION

- [Configure the Deactivation and Reactivation of GRES | 99](#)

## Configure 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;
        }
    }
}
```

## How to Make Global Changes in the Device Configuration

You can make global changes to variables and identifiers in the device 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>
```

The pattern *pattern1* option is a text string or regular expression that defines the identifiers and values you want to replace in the configuration.

The *pattern2* option is a text string or regular expression that replaces the identifiers and values located within *pattern1*.

The CLI 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 supported.

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 6: 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.



**Table 6: Common Regular Expressions to Use with the replace Command (Continued)**

Operator	Function
\$	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.
*	Denotes 0 or more terms.
+	Denotes one or more terms.
.	Denotes any character except for a space (" ").
\	A backslash escapes special characters to suppress their special meaning. For example, \. matches . (period symbol).
\n	Back reference that matches the <i>n</i> th group.
&	Back reference that matches the entire match.

The following table lists some replacement examples.

**Table 7: Replacement Examples**

Command	Result
replace pattern myrouter with router1	Match: myrouter Result: router1

Table 7: Replacement Examples *(Continued)*

Command	Result
replace pattern "192\.168\.(.*)/24" with "10.2.1/28"	Match: 192.168.3.4/24 Result: 10.2.3.4/28
replace pattern "1.1" with "abc&def"	Match: 1.1 Result: abc1.1def
replace pattern 1.1 with " abc&def"	Match: 1#1 Result: abc&def

## Example: How to Use Global Replace in a Device Configuration—the \n Back Reference

### IN THIS SECTION

- [Requirements | 103](#)
- [Overview | 104](#)
- [Configuration | 105](#)

This example shows how you can use a back reference 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 back reference. 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

### IN THIS SECTION

- [Configuring a Replacement Using a Back Reference in the Command](#) | 105
- [Results](#) | 105

## Configuring a Replacement Using a Back Reference in the Command

### Step-by-Step Procedure

To replace a pattern in a Junos OS configuration using a back reference:

- 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 back reference (`\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: Global Replace in a Device Configuration—Replacing an Interface Name

### IN THIS SECTION

- [Requirements | 107](#)
- [Overview | 107](#)
- [Configuration | 107](#)

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 108](#).

```
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

### IN THIS SECTION

- [CLI Quick Configuration | 108](#)
- [Configuring an Interface Name Change | 108](#)
- [Results | 108](#)

## 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.

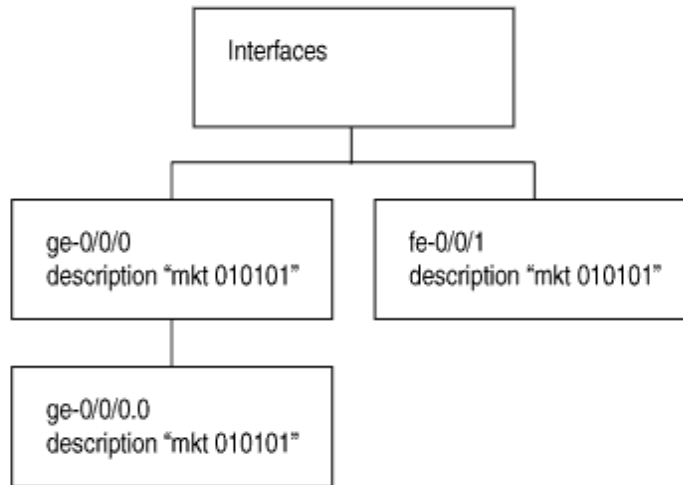
## Example: Global Replace in a Device Configuration—the upto Option

Consider the hierarchy shown in [Figure 3 on page 110](#). 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 device configuration:



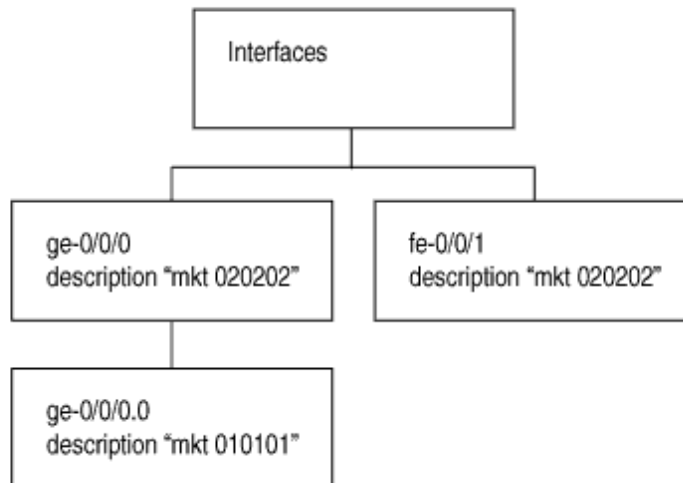
Figure 3: Replacement by Object

Current Configuration:



user@host > **replace pattern 01 with pattern 02 upto 2**

Resulting Configuration:



g017228

An upto 2 option in the replace command converts 01 to 02 for two object instances. The objects under the main interfaces ge-0/0/0 and fe-0/0/1 will be replaced first (since these are siblings in the hierarchy level). Because of the upto 2 restriction, the replace command replaces patterns in the first and second instance in the hierarchy (siblings), but not the third instance (child of the first instance).

```

user@host# show interfaces
ge-0/0/0 {
    description "mkt 010101"; #First instance in the hierarchy
    unit 0 {

```

```

        description "mkt 010101"; #Third instance in the hierarchy (child of the first
        instance)
    }
}
fe-0/0/1 {
    description "mkt 010101"; #second instance in the hierarchy (sibling of the first
    instance)
    unit 0 {
        family inet {
            address 200.200.20.2/24;
        }
    }
}
[edit]
user@host# replace pattern 01 with 02 upto 2
[edit]
user@host# commit
commit complete

```

```

[edit]
user@host# show interfaces
ge-0/0/0 {
    description "mkt 020202"; #First instance in the hierarchy
    unit 0 {
        description "mkt 010101"; #Third instance in the hierarchy (child of the first
        instance)
    }
}
fe-0/0/1 {
    description "mkt 020202"; #second instance in the hierarchy (sibling of the first
    instance)
    unit 0 {
        family inet {
            address 200.200.20.2/24;
        }
    }
}

```

## Add Comments in a Device Configuration

### IN THIS SECTION

- [Add Comments in the CLI | 112](#)
- [Add Comments in a File | 113](#)

You can include comments in a device 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..

### Add 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:** 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;
    }
  }
}
```

## Add 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: Include Comments in a Device Configuration by Using the CLI

### IN THIS SECTION

- [Requirements | 115](#)
- [Overview | 115](#)
- [Configuration | 115](#)

Adding comments to a device configuration makes the configuration file readable and more readily understood by users. 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

### IN THIS SECTION

- [CLI Quick Configuration | 116](#)
- [Including Comments in the CLI Configuration Mode | 116](#)

## 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.

## Example: Use the Wildcard Command with the Range Option

### IN THIS SECTION

- [Requirements | 118](#)
- [Overview | 118](#)



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 wildcard command along with ranges in `activate`, `deactivate`, `delete`, `protect`, `set`, `show`, and `unprotect` configuration commands.



**NOTE:** The wildcard command cannot create a configuration hierarchy. You use it to modify existing statements. Expect CLI errors of the form `warning: statement not found` when you try to set parameters, such as `deactivate`, on an empty hierarchy.

## Requirements

Junos OS Release 12.1 or later running on the device.

## Overview

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

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

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

## Configuration

### IN THIS SECTION

- Use the wildcard command with range statements for a variety of configuration tasks. | 119

- [Using the Range Option for Configuring a Series of Named Identifiers for a Configuration Statement | 120](#)
- [Specifying Multiple Ranges in the Syntax | 121](#)
- [Specifying a Range and Unique Numbers In the Syntax | 123](#)
- [Excluding Some Values from a Range | 124](#)
- [Specifying a Range with a Step Number | 125](#)
- [Use Wildcard to Delete Policies p1-p10 | 126](#)

Use the wildcard command with range statements for a variety of configuration tasks.

### Step-by-Step Procedure

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

```
[edit]
user@host# wildcard range ?
Possible completions:
  activate      Remove the inactive tag from a statement
  annotate      Annotate the statement with a comment
  deactivate     Add the inactive tag to a statement
  delete        Delete a data element
  protect       Protect the statement
  set           Set a parameter
  show          Show a parameter
  unprotect     Unprotect the statement
```

2. The wildcard statement can be qualified with a full configuration hierarchy. When parked at a configuration hierarchy, the wildcard command is relative to the options supported at that hierarchy.

```
[edit]
user@host# wildcard range set interfaces ?
Possible completions:
  <interface-name>  Interface name
  + apply-groups     Groups from which to inherit configuration data
  + apply-groups-except Don't inherit configuration data from these groups
```

```

> interface-range      Interface ranges configuration
> interface-set        Logical interface set configuration
    lo0                Interface name
> stacked-interface-set Stacked interface set configuration
> traceoptions         Interface trace options
    xe-0/0/0:0         Interface name
    xe-0/0/0:1         Interface name

user@host# edit interfaces
[edit interfaces]
wildcard range set ?
Possible completions:
    <interface-name>    Interface name
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
> interface-range      Interface ranges configuration
> interface-set        Logical interface set configuration
    lo0                Interface name
> stacked-interface-set Stacked interface set configuration
> traceoptions         Interface trace options
    xe-0/0/0:0         Interface name
    xe-0/0/0:1         Interface name

```

In the first case the full path to the interfaces hierarchy is specified with the wildcard statement. In the second, the user is parked at the interfaces hierarchy. The options supported are the same with either method.

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

### Step-by-Step Procedure

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

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

2. Display the result. The range wildcard range is expanded to 24 different set commands to configure interfaces with port numbers ranging from 0 through 23. In this case the interfaces are created in

order to add the vpls protocol family. If the interfaces are already defined then just the family is added.

```

root@R1# show | compare
[edit interfaces]
+   ge-0/0/0 {
+       unit 0 {
+           family vpls;
+       }
+   }
+   ge-0/0/1 {
+       unit 0 {
+           family vpls;
+       }
+   }
+   ge-0/0/2 {
+       unit 0 {
+           family vpls;
+       }
+   }
+   . . .
+   ge-0/0/23 {
+       unit 0 {
+           family vpls;
+       }
+   }

```

## Specifying Multiple Ranges in the Syntax

### Step-by-Step Procedure

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



**NOTE:** For the remaining examples you create a temporary copy of your configuration with ten policy statements. Having these statements avoids CLI errors that are reported when you try to manipulate attributes in a null portion of the hierarchy.

```

[edit]
user@device# set policy-options policy-statement p1 then reject
set policy-options policy-statement p10 then reject

```

```

set policy-options policy-statement p2 then reject
set policy-options policy-statement p3 then reject
set policy-options policy-statement p4 then reject
set policy-options policy-statement p5 then reject
set policy-options policy-statement p6 then reject
set policy-options policy-statement p7 then reject
set policy-options policy-statement p8 then reject
set policy-options policy-statement p9 then reject

```

```

[edit]
user@device# save temp
Wrote 254 lines of configuration to 'temp'

```

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

```

[edit]
user@host# wildcard range deactivate policy-options policy-statement p[1-3,5-7,6-9]

```

2. Display the result. The wildcard range expands to add the deactivate statement to policies that have names matching the specified ranges. Of note is the absence of *p4* and *p10* in the list of changes to the candidate configuration.



**NOTE:** An error is returned if the policy name specified does not already exist in the configuration. This is because you cannot add the deactivate statement to an empty portion of the configuration hierarchy.

```

[edit]
show | compare
[edit policy-options]
!   inactive: policy-statement p1 { ... }
!   inactive: policy-statement p2 { ... }
!   inactive: policy-statement p3 { ... }
!   inactive: policy-statement p5 { ... }
!   inactive: policy-statement p6 { ... }
!   inactive: policy-statement p7 { ... }
!   inactive: policy-statement p8 { ... }
!   inactive: policy-statement p9 { ... }

```

## Specifying a Range and Unique Numbers In the Syntax

### Step-by-Step Procedure

You can specify a combination of a range and unique numbers in the syntax of the wildcard range command.

1. Reload the ["temporary configuration" on page 121](#).

```
[edit]
user@host# load override temp
load complete
```

1. To specify a range and a unique number, separate them with a comma. In this example 10 is added as a unique number to a series of ranges.

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

2. Display the result. The wildcard range expands to add the deactivate statement to policies that have names matching the specified ranges and also to the unique number. As a result the *p10* policy is also deactivated in this example.



**NOTE:** An error is returned if the policy name specified does not already exist in the configuration. This is because you cannot add the deactivate statement to a empty configuration hierarchy.

```
[edit]
user@host# show | compare
[edit policy-options]
!   inactive: policy-statement p1 { ... }
!   inactive: policy-statement p10 { ... }
!   inactive: policy-statement p2 { ... }
!   inactive: policy-statement p3 { ... }
!   inactive: policy-statement p5 { ... }
!   inactive: policy-statement p7 { ... }
```

## Excluding Some Values from a Range

### Step-by-Step Procedure

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

1. Reload the ["temporary configuration" on page 121](#).

```
[edit]
user@host# load override temp
load complete
```

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

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

3. Display the result. The wildcard range expands to add the deactivate statement to policies that have names matching the specified ranges. Note that policies *p3* and *p4* are absent in the list of modifications made to the candidate configuration. They have been excluded from the wildcard range.



**NOTE:** An error is returned if the policy name specified does not already exist in the configuration. This is because you cannot add the deactivate statement to a empty configuration hierarchy.

```
[edit]
user@host# show | compare
[edit policy-options]
!   inactive: policy-statement p1 { ... }
!   inactive: policy-statement p2 { ... }
!   inactive: policy-statement p5 { ... }
```

## Specifying a Range with a Step Number

### Step-by-Step Procedure

You can provide a step number to have a constant interval in the range. The step value counts the first match, and then skips subsequent matches up to the specified step value.

1. Reload the ["temporary configuration" on page 121](#).

```
[edit]
user@host# load override temp
load complete
```

2. To provide a step, include the step value in the syntax preceded by a forward slash (/). Here we specify a range of 1-10, skipping 1 number between each match using a step value of 2. The effect is to skip all even numbered matches as we begin by matching an odd number (1).

```
[edit]
user@host# wildcard range deactivate policy-options policy-statement p[1-10/2]
```

3. Display the result. The wildcard range expands to add the deactivate statement to every other matching statement, starting with the first match.



**NOTE:** An error is returned if the policy name specified does not already exist in the configuration. This is because you cannot add the deactivate statement to a empty configuration hierarchy.

```
[edit]
user@host# show | compare
[edit policy-options]
!   inactive: policy-statement p1 { ... }
!   inactive: policy-statement p3 { ... }
!   inactive: policy-statement p5 { ... }
!   inactive: policy-statement p7 { ... }
!   inactive: policy-statement p9 { ... }
```



To better illustrate the step function, the temporary configuration is restore and the wildcard command is repeated, this time with a step value of 4. Now 3 entries are skipped after each initial match.

```
[edit]
user@host# load override temp
load complete
user@host## wildcard range deactivate policy-options policy-statement p[1-10/4]
\
```

4. Display the result. The wildcard range expands to add the deactivate statement to every fourth matching statement, starting with the first match.

```
[edit]
user@host# show | compare
[edit policy-options]
!   inactive: policy-statement p1 { ... }
!   inactive: policy-statement p5 { ... }
!   inactive: policy-statement p9 { ... }
```

## Use Wildcard to Delete Policies p1-p10

### Step-by-Step Procedure

Delete the temporary policies added to test the wildcard function.

1. Use the delete statement with a wildcard range statement to delete policies *p1* through *p10*.

```
[edit]
user@host# wildcard range delete policy-options policy-statement p[1-10]
```

2. Display the result. Our policy-options stanza contained only policies *p1* through *p10*. This results in an empty policy-options configuration hierarchy.

```
[edit]
user@host# show policy-options
```

# Use Configuration Groups to Quickly Configure Devices

## IN THIS SECTION

- [Configuration Groups Overview | 128](#)
- [Configure Configuration Groups | 129](#)
- [Create a Configuration Group | 129](#)
- [How to Apply a Configuration Group | 131](#)
- [Example: Create and Apply Configuration Groups | 133](#)
- [Example: Disable Inheritance of a Configuration Group | 134](#)
- [Example: Use the junos-defaults Configuration Group | 136](#)
- [Example: Use Wildcards with Configuration Groups | 139](#)
- [How to Improve Commit Time When Using Configuration Groups | 142](#)
- [Example: Configure Sets of Statements with Configuration Groups | 143](#)
- [Example: Configure Interfaces Using Configuration Groups | 144](#)
- [Example: Use Configuration Groups to Configure a Consistent IP Address for the Management Interface | 147](#)
- [Example: Use Configuration Groups to Configure Peer Entities | 149](#)
- [Example: Use Configuration Groups to Establish Regional Configurations | 151](#)
- [Example: Configure Wildcard Configuration Group Names | 152](#)
- [Example: Reference the Preset Statement from the Defaults Group | 154](#)
- [Example: View Default Statements That Have Been Applied to the Configuration | 155](#)
- [Set Up Routing Engine Configuration Groups | 156](#)
- [How to Use Conditions to Apply Configuration Groups | 158](#)
- [Example: Configure Conditions for Applying Configuration Groups | 159](#)

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

## Configuration Groups Overview

### IN THIS SECTION

- [How Configuration Groups Work | 128](#)
- [Inheritance Model | 128](#)

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

### How Configuration Groups Work

Configuration groups enable 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. 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 Juniper Networks devices. For example, you can group statements that are repeated in many places in the configuration, such as when configuring interfaces. By grouping statements, you can limit configuration updates to just the group.

You can also use wildcards in a configuration group. Any object that matches the wildcard expression inherits the group configuration data.

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 you can use throughout the configuration but that are known only to the 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. The target automatically inherits data values that you change in the configuration group. The target does not need to contain the inherited information. However, the inherited values can be overridden in the target without affecting the source from which they were inherited.

This inheritance model enables you to see only the instance-specific information without seeing the inherited details. A command pipe in configuration mode enables you to display the inherited data.

## Configure Configuration Groups

For areas of your configuration to inherit configuration statements, you must first put the statements into a configuration group. You 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 the configuration group from step 1 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.

## Create a Configuration Group

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

When you apply the group in different sections of the configuration, that part of the configuration inherits the statements configured in the 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 inherited target reflects the changes automatically.

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

This inheritance model enables you to see only the instance-specific information without seeing the inherited details. A command pipe in configuration mode enables 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 configure all of your `ge-0/0/1` interfaces for the MTU value of 1500:

1. 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;
      }
    }
  }
}
```

2. Next, you apply the group in the interface configuration.

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

3. 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;
  }
}
```

If you want to configure MTU value for interface `ge-0/0/1` in different parts of the configuration, you can apply the group statement using the `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 use a configuration group, you can change the group configuration, thereby automatically updating all associated interfaces.

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;
      }
    }
  }
}
```

## How to Apply a Configuration Group

If you want a Juniper Networks device configuration to inherit the statements from a configuration group, include the `apply-groups` statement in the configuration.

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

If you specify more than one group name, you must list the names in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.

For devices that support multiple Routing Engines, you can specify `re0` and `re1` 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.

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. It then inherits configuration data from group two and group 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 ];
      }
    }
  }
}

```

The root level is the default logical system. When you configure a group defined for the root level, you cannot successfully apply that group to a nondefault logical system under the `[edit logical-systems logical-system-name]` hierarchy level. Although the device 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.



**NOTE:** In Junos Evolved, Routing Engine-specific groups `re0` and `re1` that are used to apply configuration selectively to Routing Engine slot 0 and slot 1 are not supported.

## Example: Create and Apply Configuration Groups

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

You gain multiple advantages by placing the system-specific configuration (SNMP contact) into a configuration group, thus separating it from the normal configuration hierarchy:

- You can replace either section without discarding data from the other, by using the `load replace` command.
- You can set a contact for a specific box because the group data is hidden by the device-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;
  }
}
```



## Example: Disable Inheritance of a Configuration Group

You can disable inheritance of a configuration group at any level except the top level of the hierarchy. To disable inheritance, you include the `apply-groups-except` statement in the configuration:

```
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: Disable 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 device, because a device can inherit configuration data from configuration groups. To view the actual values used by the device, you 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, you 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 may not be able to view information about passwords or other important data where ## is used. To view the complete configuration details with all the information

(without just the comments marked with ##), you 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;
    }
}
```

## Example: Use 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 device. 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, you 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, you include the selected `junos- default-name` statement at the applicable hierarchy level.

To view the list of applications from the `junos-defaults` group, you issue the `show configuration groups junos-defaults applications`. The applications that begin with `junos-` are configured by Juniper Networks by default. The following example displays a partial list of Junos defaults groups applications.

```

user@host>show configuration groups junos-defaults applications
## protect: groups junos-defaults
##
#
# File Transfer Protocol
#
application junos-ftp {
  application-protocol ftp;
  protocol tcp;
  destination-port 21;
}
#

```

```

# Trivial File Transfer Protocol
#
application junos-ftp-data {
    application-protocol ftp-data;
    protocol tcp;
    destination-port 20;
}
application junos-tftp {
    application-protocol tftp;
    protocol udp;
    destination-port 69;
}
#
# Two-Way Active Measurement Protocol
#
application junos-twamp {
    application-protocol twamp;
    protocol tcp;
    destination-port 862;
}
#
# Real Time Streaming Protocol
#
application junos-rtsp {
    application-protocol rtsp;
    protocol tcp;
    destination-port 554;
}
#
# Network Basic Input Output System - networking protocol used on
# Windows networks    session service port
#
application junos-netbios-session {
    protocol tcp;
    destination-port 139;
}
application junos-smb-session {
    protocol tcp;
    destination-port 445;
}
application junos-ssh {
    protocol tcp;
    destination-port 22;
}

```

```

}
application junos-telnet {
    protocol tcp;
    destination-port 23;
}

```

## Example: Use Wildcards with Configuration Groups

You can use wildcards to identify names and allow one statement to provide data for a variety of statements.

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 an open bracket `[` rather than introducing a character class.
- A character class matches any of the characters between the square brackets. Within a configuration group, you must enclose in quotation marks an interface name that includes a character class.
- Hyphen ( `-` )—Specifies a range of characters.
- Exclamation point ( `!` )—You can complement the character class 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 using an identifier inside the groups hierarchy, start the identifier name with something other than `<`. However, if you are defining a wildcard statement, you can use `<` because the wildcard statement must have a closing `>`.

Using wildcards in configuration groups follows the same rules as using them for normal configuration. However, `<` and `>` have a special meaning when used under the groups hierarchy. In the groups hierarchy,

you must enclose in angle brackets any term using a wildcard pattern *<pattern>* to differentiate it from other wildcards 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 enable you to pass normal wildcards 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\*>*.
- 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;
    }
}
}

```

## How to Improve Commit Time When Using Configuration Groups

You use configuration groups to apply configurations across other hierarchies without re-entering configuration data. You can specify every configuration detail in a configuration groups. You can also use wildcards in configuration groups to configure ranges of data, without detailing each configuration line. Another way to use configuration groups is to create 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 affected 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 enables the system to build the inheritance path for each configuration group inside the database rather than in the process memory. This change can improve commit time performance. However, it can also increase the database size.

## Example: Configure 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: Configure 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 <*> {

```



```

    ##
    ## "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 Operational Commands](#) | 283

## Example: Use Configuration Groups to Configure a Consistent IP Address for the Management Interface

On devices with multiple Routing Engines, each Routing Engine is configured with a separate IP address for the management interface. To access the primary Routing Engine, you must know which Routing Engine is active and use the appropriate IP address.

Another option for consistent access to the primary Routing Engine is to configure an additional IP address. You then 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 primary Routing Engine. During switchover, the address moves to the new primary Routing Engine.

This example configures address 10.17.40.131 for both Routing Engines and includes a `master-only` statement. With this configuration, the 10.17.40.131 address is active only on the primary Routing Engine. The address remains consistent regardless of which Routing Engine is active. Address 10.17.40.132 is

assigned to fxp0 on global interfaces re0:mgmt-0, and 10.17.40.133 is assigned to fxp0 on global interfaces re1:mgmt-0.

```
[edit groups global interfaces re0:mgmt-0 interfaces fxp0]
unit 0 {
    family inet {
        address 10.17.40.131/25 {
            master-only;
        }
        address 10.17.40.132/25;
    }
}
[edit groups global interfaces re1:mgmt-0 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:

- You must assign unique IP addresses for two interfaces that have duplicate addresses on private and public interfaces. When graceful Routing Engine switchover (GRES) is enabled, the CLI displays an appropriate commit error message if it finds identical addresses. This error can occur 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.
- The em0 management Ethernet interface is used for the TX Matrix Plus router, T1600 routers in a routing matrix, and PTX Series Packet Transport Routers. Junos OS automatically creates the device's management Ethernet interface, em0.

## Example: Use Configuration Groups to Configure Peer Entities

This example creates a group `some-isp` that contains configuration data relating to another ISP. It then inserts `apply-group` statements at various points to allow those locations in the configuration hierarchy to inherit this data.

```
[edit]
user@host# show
groups {
  some-isp {
    interfaces {
      <xe-*> {
        gigether-options {
          flow-control;
        }
      }
    }
    protocols {
      bgp {
        group <*> {
          neighbor <*> {
            remove-private;
          }
        }
      }
      pim {
        interface <*> {
          version 1;
        }
      }
    }
  }
}
interfaces {
  xe-0/0/0 {
    apply-groups some-isp;
    unit 0 {
      family inet {
        address 10.0.0.1/24;
      }
    }
  }
}
```



```

    }
}
protocols {
    bgp {
        group main {
            neighbor 10.254.0.1 {
                apply-groups some-isp;
            }
        }
    }
    pim {
        interface xe-0/0/0.0 {
            apply-groups some-isp;
        }
    }
}
[edit]
user@host# show | display inheritance
interfaces {
    xe-0/0/0 {
        ##
        ## "igether-options" was inherited from group "some-isp"
        ##
        igether-options {
            ##
            ## "flow-control" was inherited from group "some-isp"
            ##
            flow-control;
        }
        unit 0 {
            family inet {
                address 10.0.0.1/24;
            }
        }
    }
}
protocols {
    bgp {
        group main {
            neighbor 10.254.0.1 {
                ##
                ## "remove-private" was inherited from group "some-isp"
                ##

```

```

        remove-private;
    }
}
}
pim {
    interface xe-0/0/0.0 {
        ##
        ## "1" was inherited from group "some-isp"
        ##
        version 1;
    }
}
}

```

## Example: Use Configuration Groups to Establish Regional Configurations

This example populates one group 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: Configure Wildcard Configuration Group Names

Wildcards are configuration group names that use special characters to create a pattern that you can apply to multiple statements. Wildcards are useful for copying one set of configuration options to many different configuration groups. You must set up your wildcard name properly to ensure that the wildcard configuration options get copied to the appropriate configuration groups.

This example configures 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 on to 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: Reference the Preset Statement from the Defaults Group

The following example is a preset statement from the 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 default statement from the defaults group, include the `junos-default-name` statement at the applicable hierarchy level. For example, to reference the 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: View Default Statements That Have Been Applied to the Configuration

To view the defaults that have been applied to the device configuration, you issue the `show | display inheritance defaults` command. This example displays the inherited 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 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.

## Set Up Routing Engine Configuration Groups

In a device with two Routing Engines, both Routing Engines should share one configuration. This setup 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 [Junos OS High Availability User Guide](#).

To set up a Routing Engine configuration group:

1. Create the configuration group `re0`. The `re0` group is a special group designator that RE0 uses, only 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 DNS server does not use the hostname that you specify in the device configuration to resolve to the correct IP address. The DNS server uses this hostname 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.

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.

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 ]
```

## How to Use Conditions to Apply Configuration Groups

You can use the `when` statement at the `[edit groups group-name]` hierarchy level to define conditions under which to apply a configuration group.

You can configure a group to apply based on the start and optional end time of day or date.

For example, you could use the `when` statement to create a generic configuration group that is applied only during specific time windows, based on a scheduled start and end time.

## Example: Configure Conditions for Applying Configuration Groups

### IN THIS SECTION

- [Requirements | 159](#)
- [Overview | 159](#)
- [Configuration | 160](#)

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.

### Overview

You can configure your group configuration data at the `[edit groups group-name]` hierarchy level. You can then use the `when` statement to apply the group configuration based on conditions such as these: 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

### IN THIS SECTION

- [Verification | 162](#)

### 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
```

### Procedure

#### 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

In 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

### IN THIS SECTION

- [Check Group Inheritance with Conditional Data | 162](#)

### *Check Group Inheritance with Conditional Data*

#### Purpose

Verify that conditional data from a configuration group is inherited when applied.

#### Action

Issue the `show | display inheritance` operational command with the `when` data to display the conditional inheritance. Using this example, you can 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
```

# View the Configuration

## IN THIS SECTION

- [Display the Current Configuration | 163](#)
- [Example: Display the Current Configuration | 164](#)
- [Display Additional Information About the Configuration | 166](#)
- [Display set Commands from the Configuration | 169](#)

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

## Display the Current Configuration

To display the current configuration for a Juniper Networks device, 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 device, 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: Display the Current Configuration

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

Set and commit a configuration:

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

## Display Additional Information About the Configuration

In configuration mode only, to display additional information about the device 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 21.3-202107190949.0;
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
    ##
    xe-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;
    }
}
...

```

## Display set Commands from the Configuration

### IN THIS SECTION

- [Example: Display set Commands from the Configuration | 170](#)
- [Example: Display set Commands with the match Option | 172](#)

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 <explicit>
```

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`.

You can use the `<explicit>` option with the `| display set` command, to explicitly display all the configuration statements that the system internally creates, as a series of set commands, when you configure certain statements from the top level of the hierarchy.

For example, assume you issue the `set interfaces ge-0/0/0.0 family inet` configuration mode command. You then show the resulting configuration with the `show interfaces ge-0/0/0 | display set` command. The output displays the same set command you entered. If you include the `explicit` argument, the output also shows the configuration statements needed to create the hierarchy where the `family inet` statement is specified. Specifically for this example, the output therefore includes the `set interfaces ge-0/0/0 unit 0` statement in addition to the `set interfaces ge-0/0/0.0 family inet` statement.

## Example: Display set Commands from the Configuration

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

```
[edit interfaces xe-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 ge-0/0/0]
user@host# show | display set
set interfaces ge-0/0/0 unit 0 family inet address 192.107.1.230/24
set interfaces xe-0/0/0 unit 1 family inet address 10.0.0.1/8
deactivate interfaces xe-0/0/0 unit 1

[edit interfaces ge-0/0/0]
user@host# show | display set | explicit
set interfaces ge-0/0/0 unit 0 family inet address 192.107.1.230/24
set interfaces xe-0/0/0 unit 0 family iso
set interfaces xe-0/0/0 unit 0 family mpls
set interfaces ge-0/0/0 unit 0 family inet
set interfaces ge-0/0/0 unit 0
set interfaces xe-0/0/0 unit 1 family inet address 10.0.0.1/8
deactivate interfaces xe-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 `show | display set relative` option. You can use the `<explicit>` option to explicitly display, as a series of commands, all the configurations that the system internally creates when you configure certain statements from the current hierarchy level.

```

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

[edit interfaces xe-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

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

```

### Example: Display 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

```

## Verify the Device Configuration

To verify that the syntax of a Juniper Networks device 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

[Commit a Device Configuration](#) | 176



# Commit the Configuration

## IN THIS SECTION

- [The Commit Model for Configurations | 174](#)
- [Commit a Device Configuration | 176](#)
- [Commit Operation When Multiple Users Configure the Software | 178](#)
- [Commit Preparation and Activation Overview | 178](#)
- [Commit Device Configurations in Two Steps: Preparation and Activation | 180](#)
- [Activate a Device Configuration with Confirmation | 182](#)
- [Schedule a Commit Operation | 184](#)
- [Monitor the Commit Process | 185](#)
- [Add a Comment to Describe the Committed Configuration | 186](#)
- [Batch Commits Overview | 188](#)
- [Example: Configure Batch Commit Server Properties | 189](#)
- [Back Up the Committed Configuration on the Alternate Boot Drive | 200](#)

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

## The 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** exists, this file is also located 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](#)

## Commit a Device Configuration

To save device 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`. By default, the file's size is limited to the default maximum size of the configuration database. You can verify the database size on disk and the maximum database size by issuing the `show system configuration database usage` command. You can also view the file's size from configuration mode by issuing the `run file list /var/rundb detail` command. If the candidate configuration exceeds the maximum database size, the commit fails with the message `configuration database size limit exceeded`. To address the error, you can:

- Increase the maximum size of the configuration database by configuring the `extend-size` statement at the [edit system configuration-database] hierarchy level on devices that support this statement. You must reboot the device to make the change effective.
- Simplify the configuration and reduce the file size by creating configuration groups with wildcards or defining less specific match policies in your firewall filters.

After you commit the configuration and, if necessary, reboot, you can verify any updates to the database size on disk or maximum database size by issuing the `show system configuration database usage` command.



**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 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, the CLI checks and activates all changes by all users.

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, the CLI commits only your own changes. 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. This allows 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 user X entered those changes 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

You can complete the commit process in two steps. The two-step commit feature enables you to configure several devices and simultaneously activate the configurations. Two-step commit provides 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 that the commit is pending activation.

In the first step, 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` displays.

In the second step, 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:**

We recommend that you do not perform changes to the configuration 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 primary.

## Commit Device Configurations in Two Steps: Preparation and Activation

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
```

## Activate a Device Configuration with 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 device 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
```

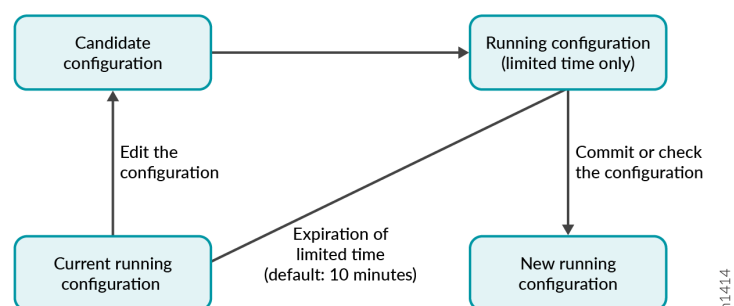
If the `commit` is not confirmed within a certain time (10 minutes by default), the operating system automatically rolls back to the previous configuration and a broadcast message is sent to all logged-in users.

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.

## Schedule a Commit Operation

You can schedule when you want your candidate configuration to become active. To save device configuration changes and activate the configuration on the device 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
```

*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 the device software 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.

## Monitor the Commit Process

To monitor the device configuration 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

```

## Add 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 characters 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:** You can also use the `commit confirmed` command in the `[edit private]` configuration mode.

Starting in Junos OS Release 24.2R1, Junos OS enforces you to issue a comment for each commit request. This helps to track changes made by multiple users or administrators at the time of commit.



**NOTE:** The `commit` command does not execute without the `comment` argument.

To enforce the user to add a comment for each commit request, configure `force-commit-log` option at the `[edit system commit]` hierarchy level.

## Batch Commits Overview

### IN THIS SECTION

- [Aggregation and Error Handling](#) | 188

Batch commit 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: Configure Batch Commit Server Properties

### IN THIS SECTION

- [Requirements | 189](#)
- [Overview | 189](#)
- [Configuration | 189](#)
- [Verification | 193](#)

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

### 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

### IN THIS SECTION

- [CLI Quick Configuration | 190](#)
- [Configuring the Commit Server Properties | 190](#)
- [Committing the Configuration from Batch Configuration Mode | 192](#)



## 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 | 193](#)
- [Checking the Batch Commit Status | 194](#)
- [Viewing the Patch Files in a Batch Commit Job | 195](#)
- [Viewing the Trace Files for Batch Commit Operations | 197](#)

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 committing 1008

## Meaning

Pending commits displays commit jobs that are added to the commit queue but are not committed yet.

Completed commits displays the list of commit jobs that are successful. Error commits are commits that failed because of an error.

## Viewing the Patch Files in a Batch Commit Job

### Purpose

View the timestamps, patch files, and the status of each of the commit jobs. Patch files show the configuration changes that occur in each commit operation that is added to the batch commit queue.

### Action

1. 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 <\*> {



```
+ 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 committing 1001
Nov  1 22:47:18 doing rollback
...
```

- To view log entries only for successful batch commit operations, issue the file `show /var/log/<filename>` command with the `| match committed` pipe option.

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 committing 1001
Nov  1 22:51:10 Error while committing 1003
Nov  1 22:52:15 Error while committing 1005
...
```

- To view log entries only for commit server events, issue the `file show /var/log/<filename>` command with the `| match "commit server"` pipe option.

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
...
```

## Back 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 is 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

[Overview of the Configure Command](#) | 65

# 4

CHAPTER

## Managing Configurations

---

### IN THIS CHAPTER

- Configuration Files Overview | 202
  - Managing Configurations | 204
  - Autoinstallation of Configuration Files Overview | 227
  - Loading Configuration Files | 232
  - Back Up Configurations to an Archive Site | 250
  - Factory Default Configuration Overview | 253
  - Rescue Configuration | 254
  - Encrypt and Decrypt Configuration Files | 255
  - Example: Protecting the Junos OS Configuration from Modification or Deletion | 259
  - Synchronizing Configurations Across Routing Engines | 271
-

# Configuration Files Overview

## IN THIS SECTION

- [Configuration Files Overview | 202](#)
- [Device Configuration Storage Overview | 204](#)

You use configuration files to configure devices and to streamline device configuration tasks. A configuration file stores the complete configuration of a device. Keep in mind these distinctions between configuration files:

- The active (running) configuration is the operational file of the device. These files control device behavior.
- The candidate configuration is the working copy that stores configuration updates. These are the files that you use to automatic device configuration.

## Configuration Files Overview

### IN THIS SECTION

- [Configuration File Terms | 203](#)

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.

The 50 most recently committed configuration files on a device are saved so that you can return to a previous configuration. The configuration files are named as follows:

- `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 enables you to make configuration changes without causing operational changes to the active configuration or causing potential damage to your current network operations. After you commit the changes you made to the candidate configuration, the system updates the active configuration.

## Configuration File Terms

**Table 8: Configuration File Terms**

Term	Definition
active configuration	Current committed configuration of a device.
candidate configuration	Working copy of the configuration that enables users to make configurational changes without causing any operational changes until this copy is committed.
configuration group	Group of configuration statements that the rest of the configuration can inherit.
commit a configuration	The act of checking a configuration for proper syntax, activating it, and marking as the current configuration file running on the device.
configuration hierarchy	A hierarchy of statements comprising the system configuration. The two types of statements are container and leaf: Container statements contain other statements. Leaf statements do not contain other statements. All the container and leaf statements together form the configuration hierarchy.
default configuration	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. Through the CLI, you set a current committed configuration to be the rescue configuration.
roll back a configuration	The act of returning to a previously committed configuration.

## Device Configuration Storage Overview

When you edit a Juniper Networks device configuration, you work in a copy of the current configuration to create a candidate configuration. The changes that you make to the candidate configuration are visible in the CLI immediately. Therefore, if multiple users are editing the configuration at the same time, all users can see all changes.

You commit your changes to cause a candidate configuration to take effect. 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 simultaneously, all changes made by all the users take effect when you commit the candidate configuration.

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. This current operational version is 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, the current configuration and three previous versions of the committed configuration are saved on the device CompactFlash card. The currently operational device 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.

## Managing Configurations

### IN THIS SECTION

- [The show | compare | display xml Command Output | 205](#)
- [Returning to the Most Recently Committed Configuration | 214](#)
- [Returning to a Previously Committed Configuration | 215](#)
- [Using Configuration Revision Identifiers | 219](#)
- [Saving a Configuration to a File | 221](#)
- [About Compressing the Current Configuration File | 222](#)

- [Free Up System Storage Space | 223](#)
- [Clean Up Files with the CLI | 225](#)

## The show | compare | display xml Command Output

### IN THIS SECTION

- [Add a Statement \(create Operation\) | 206](#)
- [Delete a Statement \(delete Operation\) | 207](#)
- [Change a Statement \(delete and create Operations\) | 209](#)
- [Change Metadata \(inactive Attribute and Operation\) | 209](#)
- [Add an Annotation \(comment Tag and create Operation\) | 211](#)
- [Change an Annotation \(comment Tag, and delete and create Operations\) | 212](#)
- [Add a Statement Inside a Container \(create Operation, and insert and key Attributes\) | 212](#)
- [Change the Order Inside a Container \(merge Operation, and insert and key Attributes\) | 213](#)

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 operating system 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.

## Add 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>

```

## Delete 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 shown in the 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>
```

## Change 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, but this is not shown in the 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>
```

## Change 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>

```

## Add 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>

```

## Change 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>

```

## Add 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>
```

## Change 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>
```

## Returning to the Most Recently Committed 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
```

## Returning to a Previously Committed Configuration

### IN THIS SECTION

- [Example of Returning to a Previous Configuration | 215](#)
- [Example of Displaying Previous Configurations | 215](#)
- [About Comparing Configuration Versions | 217](#)

This topic explains how you can return to an earlier configuration than the most recently committed one.

### Example of Returning to a Previous Configuration

To return to a previous configuration, you 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.

Example:

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

### Example of Displaying Previous Configurations

To display previous configurations, you use the rollback ? command. You include the rollback number, date, time, the name of the user who committed changes, and the method of commit.

Example:

```
[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]

```

## About Comparing Configuration Versions

In configuration mode only, when you have made changes to the configuration, you can compare the candidate configuration with a prior version. To compare versions, you use the `compare` command to display the configurations. The `compare` command compares the candidate configuration with either the current committed configuration or a configuration file. This command also displays the differences between the two configurations.

To compare configurations, you 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 system compares candidate configuration against the active configuration file (`/config/juniper.conf`).

The comparison output includes the following symbols in the prefix for statements that are:

- In the candidate configuration only: a plus sign (+).
- In the comparison file only: a minus sign (-).
- Unchanged; a single blank space ( ).

The following example shows various changes, followed by a comparison of the candidate configuration with the active configuration. The example shows 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;
    }

```

## Using Configuration Revision Identifiers

Every commit has a configuration revision identifier (CRI) associated with it. The CRI is a unique string that, unlike the rollback index, does not change when new configurations are committed.

Because the CRI for a given committed configuration is fixed, it has advantages over using a rollback index. Network management systems (NMS) can cache the CRI for a given commit. At a later date, the NMS can compare the cached value to the CRI of the current configuration on the network device to detect if other systems made out-of-band configuration changes to the device, for example, during a maintenance window.

Additionally, starting in Junos OS and Junos OS Evolved Release 20.4R1, you can use the CRI associated with a committed configuration to:

- View the configuration.
- Compare two configurations.
- Revert to the configuration.
- Retrieve the current rollback index associated with that configuration.

To view the CRI associated with each commit, use the `show system commit include-configuration-revision` command. This will display the system commit history and the CRI for each commit.

```
user@host> show system commit include-configuration-revision
0   2020-08-02 00:42:58 IST by user via cli re0-1596309177-4
1   2020-08-02 00:42:53 IST by user via cli re0-1596309173-3
2   2020-08-02 00:42:50 IST by user via cli re0-1596309170-2
3   2020-08-02 00:42:40 IST by user via other re0-1596309160-1
```

Alternatively, you can view the CRI for a specific rollback number by issuing the `show system rollback number configuration-revision` command.

```
user@host> show system rollback 0 configuration-revision
The corresponding configuration revision is: re0-1596309177-4
```

Once you have the CRI string for a specific commit, you can view that configuration with the `show system configuration revision cri-string` command.

```
user@host> show system configuration revision re0-1596309177-4
```

You can compare 2 configurations by using the `compare` option with both CRIs.

```
user@host> show system configuration revision compare re0-1596309177-4 re0-1596309173-3
```

You can also use view the rollback number for a specific CRI by including the `rollback-number cri-string` option.

```
user@host> show system configuration revision rollback-number re0-1596309160-1
The corresponding rollback number is: 3
```

Additionally, in configuration mode, you can roll back to a configuration by specifying the CRI instead of the rollback index.

```
[edit]
user@host# rollback revision re0-1596309160-1
load complete
```

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

## Saving a Configuration to a File

Saving a device 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.

Example:

```
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;
        ...
    }
}

```

## About Compressing the Current Configuration File

By default, the current operational configuration file is compressed and is stored in the file **juniper.conf.gz** in the **/config** file system. The operational configuration file is stored 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 operational 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, you 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 compression-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
```

## Free Up System Storage Space

### IN THIS SECTION

- [Problem | 224](#)
- [Solution | 224](#)

## Problem

### Description

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

The following error message appears 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. Issue a request to clean up (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-rtbdb/env.lck
 12.0K Jul 26 20:55 /var/tmp/if-rtbdb/env.mem
2688.0K Jul 26 20:55 /var/tmp/if-rtbdb/shm_usr1.mem
 132.0K Jul 26 20:55 /var/tmp/if-rtbdb/shm_usr2.mem
2048.0K Jul 26 20:55 /var/tmp/if-rtbdb/trace.mem
  155B Jul 26 20:55 /var/tmp/krt_gencfg_filter.txt
   0B Jul 26 20:55 /var/tmp/rtbdb/if-rtbdb
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. Select yes to delete the files.

3. Reboot the device.



#### **BEST PRACTICE:** Best Practice

We recommend that you regularly issue a request to clean up the system file storage. Cleaning up the system file storage space optimizes device performance.

## Clean 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 you can delete.

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 you can safely delete.

```
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 you can safely delete . The dry-run action lets you review the list before you issue the `request system storage cleanup` command to delete the files.

**NOTE:** On SRX Series Firewalls, the `/var` hierarchy is hosted in a separate partition (instead of the root partition). If the operating system 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.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
16.2R2	Beginning with Junos OS Release 16.2R2, the <code>show   compare   display xml</code> command omits the <code>&lt;configuration&gt;</code> 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.

# Autoinstallation of Configuration Files Overview

## IN THIS SECTION

- [Configuration File Autoinstallation—An Overview | 227](#)
- [Configuring Autoinstallation of Configuration Files \(CLI Procedure\) | 230](#)

Autoinstallation is the automatic configuration of devices over the network without manual intervention, including manual configuration. You (the network administrator) use autoinstallation to save time and to implement the same configuration consistently across devices.

## Configuration File Autoinstallation—An Overview

### IN THIS SECTION

- [Typical Uses for Autoinstallation | 228](#)
- [Autoinstallation Configuration Files and IP Addresses | 228](#)
- [Typical Autoinstallation Process on a New Device | 229](#)

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 network devices 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 work only with HTTP for autoinstallation. They do not support TFTP or FTP protocols. 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:

- Deploy and update multiple devices from a central location in the network.
- Update a device automatically, when 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. In addition, you must ensure that a service such as Dynamic Host Configuration Protocol (DHCP) is available to assign an IP address to the device.

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 on a switch. This file contains just enough configuration information for you to telnet to the device and configure it manually.
- **hostname.conf**—Host-specific configuration file for autoinstallation on a device. This file contains all the configuration information necessary for the device. In the filename, replace *hostname* with the hostname assigned to the device.
- **router.conf**—A default configuration file named `router.conf` with the minimum configuration necessary to enable you to telnet into the new device for further configuration.

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. You must attach this intermediate device directly to the new device so that the new device can send TFTP, Boot Protocol (BOOTP), and Domain Name System (DNS) requests through the intermediate device. In this case, you specify the IP address of the intermediate device as the location at which to receive TFTP autoinstallation requests.

## 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, 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.

## 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. A configuration server is typically a Trivial File Transfer Protocol (TFTP) server. You can use autoinstallation to deploy multiple devices automatically from a central location in the network.

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

To enable autoinstallation to run, complete the following steps:

1. Ensure that a service such as Dynamic Host Configuration Protocol (DHCP) is available to assign an IP address to the device.
2. Configure a DHCP server on your network to meet your network requirements. You can configure a switch to operate as a DHCP server.
3. 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.
4. Physically attach the device to the network using a Gigabit Ethernet port.
5. If you configured the DHCP server to provide only the TFTP server hostname, add an IP address-to-hostname mapping entry for the TFTP server. Map the TFTP server hostname to the DNS database file on the Domain Name System (DNS) server in the network.
6. 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.
7. 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.

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

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 such as **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.

Example:

```
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

- [Examples for Loading a Configuration from a File or the Terminal | 233](#)
- [How Character Encoding Works on Juniper Networks Devices | 236](#)
- [About Specifying Statements and Identifiers | 237](#)
- [About Loading a Configuration from a File | 242](#)
- [Upload a Configuration File | 246](#)
- [Load JSON Configuration Data With Unordered List Entries | 247](#)

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

## Examples for Loading a Configuration from a File or the Terminal

You can create a file containing configuration data for a Juniper Networks device, 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 the configuration at a later time.

You can also create a configuration while typing at the terminal and then load the configuration. Loading a configuration from the terminal is 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, you 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, you 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, you 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, you 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, you specify the `replace` option. The `load replace` operation looks for `replace:` tags that you added to the loaded file. The operation then replaces those 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 that 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 no statement of the same name exists, 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. In this scenario, the `override` or `merge` operation takes precedence and is performed.

If you are performing a `replace` operation, and if the file that you specify lacks `replace: tags`, the `replace` operation runs as a `merge` operation. The `replace` operation also runs as a `merge` operation if the text you type lacks `replace: tags`. This information might be useful if you are running automated scripts and cannot know in advance whether the scripts need to perform a `replace` operation 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 information 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, you specify the `update` option at any level of the hierarchy. The `load update` operation compares the candidate configuration and the new configuration data. This operation changes only those parts of the candidate configuration that are different from the new configuration. You would use this operation, 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 load JSON configuration data that contains unordered list entries, that is, list entries where the list key is not necessarily the first element in the list entry, see ["Load JSON Configuration Data With Unordered List Entries" on page 247](#).

To change part of the configuration with a patch file, you 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 before 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.

In this example, you first run the `show | compare` command.

Example:

```
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;
+ }
```

Continuing this example, you copy the output of the `show | compare` command to the clipboard, making sure to include the hierarchy levels. On router2, router3, and router4, you type `load patch terminal` and paste the output. You then press Enter and 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, you specify the `relative` option. This option loads the incoming configuration relative to your current edit point in the configuration hierarchy.

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 working 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
```

## How Character Encoding Works on Juniper Networks Devices

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, the software escapes and encodes these characters. The software escapes or encodes the characters using the equivalent UTF-8 decimal character reference.

The CLI attempts to display any non-ASCII characters in configuration data that is produced in text, set, or JSON format. The CLI also attempts to display these characters in command output that is produced in text format. In the exception cases, the CLI displays the UTF-8 decimal character reference instead. (Exception cases include configuration data in XML format and command output in XML or JSON format.) In NETCONF and Junos XML protocol sessions, you see a similar result if you request configuration data or command output that contains non-ASCII characters. In this case, 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.

```
[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, the ñ character maps to its equivalent UTF-8 decimal character reference `&#195;&#177;`. The same result occurs if you display the configuration in any format in a NETCONF or Junos XML protocol session.

```
[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, you can load non-ASCII characters using their equivalent UTF-8 decimal character references.

## About Specifying Statements and Identifiers

### IN THIS SECTION

● [Specifying Statements](#) | 238



This topic provides details about CLI container statements and leaf statements so that you know how to must specify them when creating ASCII configuration files. This topic 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. You use an identifier when a statement can be specified more than once in a configuration.

When specifying a statement, you must specify a statement name, 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 specify statements and identifiers. Each statement has a preferred style, and the CLI uses that style when displaying the configuration in response to a configuration mode `show` command. You can specify statements and identifiers in one of the following ways:

- 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 that you enter an address in a valid format. Otherwise, an error message indicates what you need to type. lists the data types the CLI checks. The following are CLI configuration input types:

**Table 9: CLI Configuration Input Types**

Data Type	Format	Examples
Physical interface name (used in the [edit interfaces] hierarchy)	<i>type-fpc/pic/port</i>	<b>Correct:</b> et-0/0/1 <b>Incorrect:</b> et-0
Full interface name	<i>type-fpc/pic/port&lt;:channel&gt;.logical</i>	<b>Correct:</b> et-0/0/1.0 <b>Incorrect:</b> et-0/0/1

Table 9: CLI Configuration Input Types (Continued)

Data Type	Format	Examples
Full or abbreviated interface name (used in places other than the [edit interfaces] hierarchy)	<i>type&lt;fpc&lt;/pic/port&gt;&gt;&lt;&lt;:channel&gt;.logical&gt;</i>	<b>Correct:</b> et, et-1, et-1/2/3:4.5
IP address	<i>0xhex-bytesoctet&lt;. octet&lt;.octet.&lt;octet&gt;&gt;&gt;</i>	<b>Correct:</b> 1.2.3.4, 0x01020304, 128.8.1, 128.8  <b>Sample translations:</b>  1.2.3 becomes 1.2.3.0 0x01020304 becomes 1.2.3.4 0x010203 becomes 0.1.2.3
IP address (destination prefix) and prefix length	<i>0xhex-bytes&lt;/length&gt;octet&lt;octet &lt;octet.&lt;octet&gt;&gt;&gt;&lt;/length&gt;</i>	<b>Correct:</b> 10/8, 128.8/16, 1.2.3.4/32, 1.2.3.4  <b>Sample translations:</b>  1.2.3 becomes 1.2.3.0/32 0x01020304 becomes 1.2.3.4/32 0x010203 becomes 0.1.2.3/32 default becomes 0.0.0.0/0
International Organization for Standardization (ISO) address	<i>hex-nibble&lt;hex-nibble ...&gt;</i>	<b>Correct:</b> 47.1234.2345.3456.00, 47123423453456.00, 47.12.34.23.45.34.56.00  <b>Sample translations:</b>  47123456 becomes 47.1234.56 47.12.34.56 becomes 47.1234.56 4712.3456 becomes 47.1234.56

Table 9: CLI Configuration Input Types *(Continued)*

Data Type	Format	Examples
OSPF area identifier (ID)	<i>0xhex-bytesoctet&lt;.octet&lt;.octet.&lt;octet &gt;&gt;&gt; decimal-number</i>	<b>Correct:</b> 54, 0.0.0.54, 0x01020304, 1.2.3.4  <b>Sample translations:</b>  54 becomes 0.0.0.54  257 becomes 0.0.1.1 128.8 becomes 128.8.0.0 0x010203 becomes 0.1.2.3

## About 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

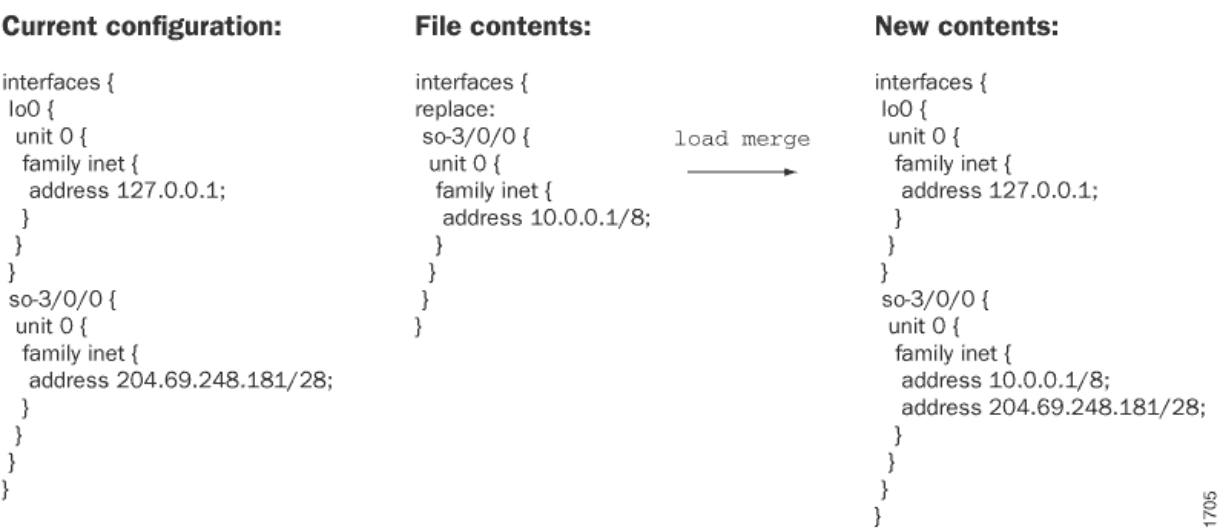


Figure 8: Using a Patch File

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
      }
    }
  }
}
}

```



# Upload 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, include one or more of the following options to perform the required action when the file is loaded.

**Table 10: Options for the load Command**

Options	Description
merge	Combines the current active configuration with either the configuration in the filename that you specify or the configuration 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. Loads either the configuration in the filename that you specify 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. 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 the existing statements with the incoming configuration. If no statement of the same name exists, the replace operation adds the statements marked with the replace tag to the active configuration.  <b>NOTE:</b> For this operation to work, you must include replace tags in the text file or in the configuration that you enter at 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.

## Load JSON Configuration Data With Unordered List Entries

The Junos schema defines certain configuration objects as lists. In JSON configuration data, a list instance is encoded as a name/array pair, and the array elements are JSON objects. Generally, the order of members in a JSON-encoded list entry is arbitrary because JSON objects are fundamentally unordered collections of members. However, the Junos schema requires that list keys precede any other siblings within a list entry and appear in the order specified by the schema.

For example, the `user` object at the `[edit system login]` hierarchy level is a list where `name` is the list key that uniquely identifies each user.

```
list user {
  key name;
  description "Username";
  uses login-user-object;
}
```

In the following sample configuration data, the list key (`name`) is the first element for each user. By default, when you load JSON configuration data, Junos devices require that the list keys precede any other siblings within a list entry and appear in the order specified by the schema.

```
{
  "configuration" : {
```

```

    "system" : {
      "login" : {
        "user" : [
          {
            "name" : "operator",
            "class" : "operator",
            "uid" : 3001
          },
          {
            "name" : "security-admin",
            "class" : "super-user",
            "uid" : 3002
          }
        ]
      }
    }
  }
}

```

Junos devices provide two options to load JSON configuration data that contains unordered list entries, that is, list entries where the list key is not necessarily the first element.

- Use the `request system convert-json-configuration operational mode` command to produce JSON configuration data with ordered list entries before loading the data on the device.
- Configure the `reorder-list-keys` statement at the `[edit system configuration input format json]` hierarchy level. After you configure the statement, you can load JSON configuration data with unordered list entries, and the device reorders the list keys as required by the Junos schema during the load operation.



**NOTE:** When you configure the `reorder-list-keys` statement, the load operation can take significantly longer to parse the configuration, depending on the size of the configuration and the number of lists. Thus, for large configurations or configurations with many lists, we recommend using the `request system convert-json-configuration` command instead of the `reorder-list-keys` statement.

For example, suppose the `user-data.json` file contains the following JSON configuration. If you tried to load the configuration, the device would emit a load error for `admin2` because the list key `name` is not the first element in that list entry.

```

user@host> file show /var/tmp/user-data.json
{

```

```

"configuration" : {
  "system" : {
    "login" : {
      "user" : [
        {
          "name" : "admin1",
          "class" : "super-user",
          "uid" : 3003
        },
        {
          "class" : "super-user",
          "name" : "admin2",
          "uid" : 3004
        }
      ]
    }
  }
}

```

If you use the request `system convert-json-configuration` command with the previous file as input, the command generates the specified output file with JSON configuration data that the Junos device can parse during the load operation.

```

user@host> request system convert-json-configuration /var/tmp/user-data.json output-filename
user-data-ordered.json

```

```

user@host> file show user-data-ordered.json

```

```

{
  "configuration":{
    "system":{
      "login":{
        "user":[
          {
            "name":"admin1",
            "class":"super-user",
            "uid":3003
          },
          {
            "name":"admin2",
            "class":"super-user",
            "uid":3004
          }
        ]
      }
    }
  }
}

```

```

    }
  ]
}
}
}
}
}

```

Alternatively, you can configure the `reorder-list-keys` configuration statement.

```

user@host# set system configuration input format json reorder-list-keys
user@host# commit

```

After you configure the statement, you can load the original JSON configuration file with unordered list entries, and the device handles the list entries when it parses the configuration.

```

user@host# load merge json /var/tmp/user-data.json
load complete

```

## Back Up Configurations to an Archive Site

### IN THIS SECTION

- [Configure the Transfer of the Active Configuration | 250](#)

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

### Configure the Transfer of the 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 active configuration by FTP, HTTP, secure copy (SCP), or SFTP periodically or after each commit.

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

```
[edit system archival configuration]
archive-sites {
    file:/path;
    file:///path;
    ftp://username@host[:port>//url-path;
    http://username@host[:port]/url-path;
    scp://username@host[:port]/url-path;
    sftp://username@host[:port]/url-path;
}
routing-instance routing-instance;
transfer-interval interval;
transfer-on-commit;
```

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, SCP-style remote file specification, or SFTP URL. The URL type **file:** is also supported. When you specify the archive site, do not add a forward slash (/) to the end of the URL.



**NOTE:**

- The URL type **file:** is supported only for local files.
- When using the FTP option, specify a double forward slash (//) after the host:port.  
For example: **ftp://username@host[:port>//url-path**

**file:/path/** is the minimal representation of a local file with no authority field and an absolute path that begins with a slash "/" as defined in RFC 8089.

**file:///path** is an example for a traditional file URI for a local file with an empty authority as defined in RFC 8089.



**NOTE:** When specifying a URL in a 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**"

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

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

If the network device reaches the archive server through a specific routing instance, configure the `routing-instance` statement at the `[edit system archival configuration]` hierarchy level, and specify the routing instance.

```
[edit system archival configuration]
routing-instance routing-instance;
```

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 in Coordinated Universal Time (UTC).

See [KB76300](#) for details on why SFTP transfers on commit fail if SSH-host keys are not configured.

# Factory Default Configuration Overview

## IN THIS SECTION

- [Restore the Default Factory Configuration | 253](#)

The default factory configuration contains the basic device configuration settings. This first configuration of the device is loaded automatically the first time you install the device and power it on.

## Restore the Default Factory Configuration

If for any reason the current active configuration fails, you can restore the default factory configuration. The default factory configuration contains the basic configuration settings and is sometimes referred to as the rescue configuration. This is the first configuration of the device and is loaded the first time you install the device and power it on.

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

To restore the switch to the default factory configuration:

1. Make sure you are in configuration mode.
2. Enter the following commands at the CLI:

```
[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 that preserve a Virtual Chassis configuration. A Virtual Chassis is a group of devices



configured to work together as if they were a single device. You can use the `load factory-default` command to restore the factory default configuration on a Virtual Chassis without removing anything needed to keep the Virtual Chassis working.

3. Make sure you configure the root (superuser) password. Set the root authentication password and commit. If you configure the plain-text-password option, you are prompted to enter and confirm the password.

```
[edit]
user@switch# set system root-authentication plain-text-password
New password: type password here
Retype new password: retype password here
user@switch# commit
```

## SEE ALSO

| *Root Password*

# Rescue Configuration

## IN THIS SECTION

- [Creating and Returning to a Rescue Configuration | 254](#)

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>
```

## Encrypt and Decrypt Configuration Files

### IN THIS SECTION

- [Encrypt Configuration Files | 256](#)
- [Decrypt Configuration Files | 258](#)
- [Modify the Encryption Key | 259](#)

You store configuration data and sensitive network information in configuration files. Encrypting configuration files enables you to secure the information they store. Decrypting means disabling the encryption of configuration files on a device and making the files readable to all.



**NOTE:** Encryption features are not available on all Juniper Networks devices. If these features are not available on one or more of your devices, the Junos OS CLI encryption-related commands described in this topic may be hidden or may not function. See your hardware documentation for details.

## Encrypt Configuration Files

To encrypt configuration files on a Juniper Networks device, you need an encryption key. You configure an encryption key in EEPROM and determine which encryption process is appropriate for your network.

To configure an encryption key, select the most appropriate `request system set-encryption-key` command in operational mode, as described in the following table.

**Table 11: The request system set-encryption-key CLI Commands**

CLI Command	Description
<code>request system set-encryption-key</code>	<p>Sets the encryption key and enables default configuration file encryption:</p> <ul style="list-style-type: none"> <li>• AES encryption for the Canada and U.S. version of the operating system</li> <li>• DES encryption for the international version of the operating system</li> </ul>
<code>request system set-encryption-key algorithm des</code>	Sets the encryption key and specifies configuration file encryption by DES.

**Table 11: The request system set-encryption-key CLI Commands** *(Continued)*

CLI Command	Description
request system set-encryption-key unique	<p>Sets the encryption key and enables default configuration file encryption with a unique encryption key that includes the chassis serial number of the device.</p> <p>When you encrypt configuration files with the unique key, you can decrypt the files on the current device only. You cannot copy encrypt configuration files to another device and decrypt them.</p>
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
```

## Decrypt Configuration Files

Decrypting configuration files means disabling the file encryption on a device, which makes the files readable to all.

To disable the encryption of configuration files on a device:

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.

Example:

```
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
```

## Modify 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.

## Example: Protecting the Junos OS Configuration from Modification or Deletion

### IN THIS SECTION

- [Requirements | 260](#)
- [Overview | 260](#)
- [Protecting a Parent-Level Hierarchy | 261](#)
- [Protecting a Child Hierarchy | 262](#)
- [Protecting a Configuration Statement Within a Hierarchy | 263](#)
- [Protecting a List of Identifiers for a Configuration Statement | 263](#)
- [Protecting an Individual Member from a Homogenous List | 264](#)

- [Unprotecting a Configuration | 265](#)
- [Verification | 266](#)

This example shows how to use the `protect` and `unprotect` commands in the configuration mode to protect and unprotect the CLI configuration.

## Requirements

This example uses the following hardware and software components:

- An MX Series or PTX Series device

## Overview

### IN THIS SECTION

- [Topology | 261](#)

The Junos OS enables you to protect the device configuration from being modified or deleted by other users. This can be accomplished by using the `protect` command in the configuration mode of the CLI. Likewise, you can also unprotect a protected configuration by using the `unprotect` command.

These commands can be used at any level of the configuration hierarchy—a top-level parent hierarchy or a configuration statement or an identifier within the lowest level of the hierarchy.

If a configuration hierarchy is protected, users cannot perform the following activities:

- Deleting or modifying a hierarchy or a statement or identifier within the protected hierarchy
- Inserting a new configuration statement or an identifier within the protected hierarchy
- Renaming a statement or identifier within the protected hierarchy
- Copying a configuration into a protected hierarchy

- Activating or deactivating statements within a protected hierarchy
- Annotating a protected hierarchy

## Topology

## Protecting a Parent-Level Hierarchy

### IN THIS SECTION

- [Procedure](#) | 261

## Procedure

### Step-by-Step Procedure

To protect a configuration at the top level of the hierarchy:

Identify the hierarchy that you want to protect and issue the `protect` command for the hierarchy at the `[edit]` hierarchy level.

For example, if you want to protect the entire `[edit access]` hierarchy level, use the following command:

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

## Results

Protects all elements under the parent hierarchy.



**NOTE:** If you issue the `protect` command for a hierarchy that is not used in the configuration, the Junos OS CLI displays the following error message:



```
[edit]
user@host# protect access
warning: statement not found
```

## Protecting a Child Hierarchy

### IN THIS SECTION

- [Procedure](#) | [262](#)

### Procedure

#### Step-by-Step Procedure

To protect a child hierarchy contained within a parent hierarchy:

Navigate to the parent container hierarchy. Use the `protect` command for the hierarchy at the parent level.

For example, if you want to protect the `[edit system syslog console]` hierarchy level, use the following command at the `[edit system syslog]` hierarchy level.

```
[edit system syslog]
user@host# protect console
```

### Results

Protects all elements under the child hierarchy.

## Protecting a Configuration Statement Within a Hierarchy

### IN THIS SECTION

- [Procedure](#) | 263

### Procedure

#### Step-by-Step Procedure

To protect a configuration statement within a hierarchy level:

Navigate to the hierarchy level containing the statement that you want to protect and issue the `protect` command for the hierarchy.

For example, if you want to protect the `host-name` statement under the `[edit system]` hierarchy level, use the following command:

```
[edit system]
user@host# protect host-name
```

### Results

## Protecting a List of Identifiers for a Configuration Statement

### IN THIS SECTION

- [Procedure](#) | 264

## Procedure

### Step-by-Step Procedure

Some configuration statements can take multiple values. For example, the address statement at the [edit system login deny-sources] hierarchy level can take a list of hostnames, IPv4 addresses, or IPv6 addresses. Suppose you have the following configuration:

```
[edit system login]
deny-sources {
  address [ 172.17.28.19 172.17.28.20 172.17.28.21 172.17.28.22];
}
```

To protect all the addresses for the address statement, use the following command at the [edit] level:

```
[edit]
user@host# protect system login deny-sources address
```

### Results

All the addresses ([172.17.28.19 172.17.28.20 172.17.28.21 172.17.28.22]) for the address statement are protected.

## Protecting an Individual Member from a Homogenous List

### IN THIS SECTION

- [Procedure | 265](#)

## Procedure

### Step-by-Step Procedure

Suppose you have the following configuration:

```
[edit groups ]
test1 {
  system {
    name-server {
      10.1.2.1;
      10.1.2.2;
      10.1.2.3;
      10.1.2.4;
    }
  }
}
```

To protect one or more individual addresses for the `name-server` statement, issue the following command at the `[edit]` level:

```
[edit]
user@host# protect groups test1 system name-server 10.1.2.1
user@host# protect groups test1 system name-server 10.1.2.4
```

### Results

Addresses 10.1.2.1 and 10.1.2.4 are protected.

## Unprotecting a Configuration

### IN THIS SECTION

- Procedure | 266

## Procedure

### Step-by-Step Procedure

Suppose you have the following configuration at the `[edit system]` hierarchy level:

```
protect: system {
  host-name bigping;
  domain-search 10.1.2.1;
  login {
    deny-sources {
      protect: address [ 172.17.28.19 172.17.28.173 172.17.28.0 174.0.0.0 ];
    }
  }
}
```

To unprotect the entire `[edit system]` hierarchy level, issue the following command at the `[edit]` level:

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

### Results

The entire `system` hierarchy level is unprotected.

## Verification

### IN THIS SECTION

- [Verify That a Hierarchy Is Protected Using the show Command | 267](#)
- [Verify That a Hierarchy Is Protected by Attempting to Modify a Configuration | 267](#)
- [Verify Usage of the protect Command | 268](#)
- [View the Configuration in XML | 269](#)

## Verify That a Hierarchy Is Protected Using the show Command

### Purpose

To check that a configuration hierarchy is protected.

### Action

In the configuration mode, issue the `show` command at the `[edit]` hierarchy level to see all the configuration hierarchies and configuration statements that are protected.



**NOTE:** All protected hierarchies or statements are prefixed with a `protect: string`.

```
...
protect: system {
  host-name bigping;
  domain-search 10.1.2.1;
  login {
    deny-sources {
      protect: address [ 172.17.28.19 172.17.28.173 172.17.28.0 174.0.0.0 ];
    }
  }
}
...
```

## Verify That a Hierarchy Is Protected by Attempting to Modify a Configuration

### Purpose

To verify that a configuration is protected by trying to modify the configuration using the `activate`, `copy`, `insert`, `rename`, and `delete` commands.

### Action

To verify that a configuration is protected:

1. Try using the `activate`, `copy`, `insert`, `rename`, and `delete` commands for a top-level hierarchy or a child-level hierarchy or a statement within the hierarchy.

For a protected hierarchy or statement, the Junos OS displays an appropriate warning that the command has not executed. For example:

```
protect: system {
  host-name a;
  inactive: domain-search [ a b ];
}
```

2. To verify that the hierarchy is protected, try issuing the activate command for the domain-search statement:

```
[edit system]
```

```
user@host# activate system domain-search
```

The Junos OS CLI displays an appropriate message:

```
warning: [system] is protected, 'system domain-search' cannot be activated
```

## Verify Usage of the protect Command

### Purpose

To view the protect commands used for protecting a configuration.

### Action

1. Navigate to the required hierarchy.
2. Issue the show | display set relative command.

```
user@host> show | display set relative
set system host-name bigping
set system domain-search 10.1.2.1
set system login deny-sources address 172.17.28.19
set system login deny-sources address 172.17.28.173
set system login deny-sources address 172.17.28.0
set system login deny-sources address 174.0.0.0
```

```
protect system login deny-sources address
protect system
```

## View the Configuration in XML

### Purpose

To check if the protected hierarchies or statements are also displayed in the XML. Protected hierarchies, statements, or identifiers are displayed with the `| display xml` attribute in the XML.

### Action

To view the configuration in XML:

1. Navigate to the hierarchy you want to view.
2. Use the `show` command with the pipe symbol and option `| display xml`:

```
[edit system]
```

```
user@host# show | display xml
[edit]
user@host# show system | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/11.2I0/junos">
  <configuration junos:changed-seconds="1291279234" junos:changed-localtime="2017-12-02
00:40:34 PST">
    <system protect="protect">
      <host-name>bigping</host-name>
      <domain-search>10.1.2.1</domain-search>
      <login>
        <message>

          \jnpr

          \tUNAUTHORIZED USE OF THIS ROUTER
          \tIS STRICTLY PROHIBITED!

        </message>
      </login>
    </system>
  </configuration>
</rpc-reply>
```



```

        <allow-commands>commit-synchronize</allow-commands>
        <deny-commands>commit</deny-commands>
    </class>
    <deny-sources>
        <address protect="protect">172.17.28.19</address>
        <address protect="protect">172.17.28.173</address>
        <address protect="protect">172.17.28.0</address>
        <address protect="protect">174.0.0.0</address>
    </deny-sources>
</login>
<syslog>
    <archive>
    </archive>
</syslog>
</system>
</configuration>
<cli>
    <banner>[edit]</banner>
</cli>
</rpc-reply>

```



**NOTE:** Loading an XML configuration with the `unprotect="unprotect"` tag unprotects an already protected hierarchy. For example, suppose you load the following XML hierarchy:

```

<protocols unprotect="unprotect">
    <ospf>
        <area>
            <name>0.0.0.0</name>
            <interface>
                <name>all</name>
            </interface>
        </area>
    </ospf>
</protocols>

```

The `[edit protocols]` hierarchy becomes unprotected if it is already protected.

# Synchronizing Configurations Across Routing Engines

## IN THIS SECTION

- [Routing Engine Synchronization Overview | 271](#)
- [Configure Multiple Routing Engines to Synchronize Committed Configurations Automatically | 275](#)

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

## Routing Engine Synchronization Overview

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 (the requesting Routing Engine) first commits the configuration. The requesting Routing Engine then copies and loads its candidate configuration to the responding Routing Engine. Each Routing Engine performs a syntax check on the candidate configuration file before committing it. 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 the commit fails on either Routing Engine, the commit process is rolled back on the other Routing Engine as well. This safeguard ensures that both Routing Engines have the same configuration.



**NOTE:** If your configuration includes a large amount of text or many `apply-groups`, commit times can be longer than desired.

For example, you may want both Routing Engines to have the same configuration. In this scenario, if you are logged in to `re1` (requesting Routing Engine), you issue the `commit synchronize` command on `re1`. Routing Engine `re1` copies and loads its candidate configuration to `re0` (responding Routing Engine). Both

Routing Engines then perform a syntax check on the candidate configuration file being committed. If no errors are found, the `re1` 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 Configuration Group" on page 131](#).

You can synchronize a Routing Engine's current operational configuration file with the other Routing Engine's configuration file. To do this, you log in to the Routing Engine from which you want to synchronize and issue the `commit synchronize` command.

Example:

```
[edit]
user@host# commit synchronize
re0:
configuration check succeeds
re1:
commit complete
re0:
commit complete
```



**NOTE:** The backup Routing Engine may be only partially committed due to invalid configuration during system reboot. In this case, the `commit synchronize` command with the `force` option from the primary 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 are terminated. When those sessions are terminated on the other Routing Engine, its configuration is synchronized with the configuration 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, you log in to the Routing Engine from which you want to synchronize. Then, you issue the command with the `force` option.

Example:

```
[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, you enter the `commit synchronize` command from the primary Routing Engine after you make any changes to the configuration. If you enter this command on the backup Routing Engine, the software displays a warning and commits the configuration.

Include the `fast-synchronize` statement at the `[edit system]` hierarchy level to have the synchronization occur simultaneously between the primary and the backup Routing Engines:

```
[edit system]
commit fast-synchronize;
```

The `fast-synchronize` configuration is valid on systems with two Routing Engines (Option not available for SRX Series). The `fast-synchronize` statement is also supported on EX Series Virtual Chassis and MX Series Virtual Chassis devices.



**NOTE:**

- Prior to Junos OS and Junos OS Evolved Release 25.4R1, when the `fast-synchronize` statement is configured, the commits on the primary 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, we recommend that you not 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. `re0` and `re1` are special groups

on dual Routing Engine devices. Whereas, member0 and member1 are special groups for EXVC.

- Prior to Junos OS and Junos OS Evolved Release 25.4R1, if fast-synchronize is enabled and both Routing Engines (primary and backup) run different software versions, the backup Routing Engine configuration may not be valid. This is true even if the primary Routing Engine validates the configuration. Therefore, ensure that the same operating system software version is running on both the Routing Engines.

The fast-synchronize commit model is the default commit synchronize model for all Junos OS dual Routing Engine devices, EX Series Virtual Chassis, and MX Series Virtual Chassis. In this model, the system executes commit check and activation simultaneously on both the Routing Engines, reducing commit latency and improving operational efficiency for Junos OS.

The configuration validation is performed on both the Routing Engines, unlike the fast-synchronize model prior to Junos OS Release 25.4R1.

You can disable fast-synchronize model by configuring no-fast-synchronize option at the [edit system commit] hierarchy level.

For MX Series Virtual Chassis systems, when fast-synchronize is the default commit model, the configuration is automatically synchronized across all members and their backup Routing Engines, regardless of whether the system commit synchronize option is configured or you explicitly issue the commit synchronize command.

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 you configure the load-scripts-from-flash statement for the requesting Routing Engine, the device synchronizes the scripts. 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.

Example:

```
[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. The scripts are still synchronized because the synchronization occurs before 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;
```

## Configure 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 primary 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 primary and backup

Routing Engines takes place one Routing Engine at a time. If the configuration has a large text size or many apply-groups, commit times can be longer than desired.

You can use the `commit fast-synchronize` statement to have the synchronization between the primary and backup Routing Engines occur simultaneously instead of sequentially. This can reduce the time needed for synchronization because the commits on the primary 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 primary 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.
- Prior to Junos OS Release 25.4R1, when the `fast-synchronize` statement is configured, the commits on the primary 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, we recommend limiting the number of 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.
- Prior to Junos OS Release 25.4R1, if `fast-synchronize` is enabled and if the primary Routing Engine and backup Routing Engines run different software versions, you cannot be sure that the backup Routing Engine configuration is valid. This is true even if the primary Routing Engine validates the configuration. Therefore, ensure that the operating system software version running on both the Routing Engines is the same. `re0` and `re1` are special groups on dual Routing Engine devices. Whereas `member0`, `member1`.. `memberN` are the special groups on EX Series Virtual Chassis devices.

The `fast-synchronize` is the default commit synchronize model for all Junos OS dual Routing Engine devices, EX Series Virtual Chassis, and MX Series Virtual Chassis. In this model, the system executes commit check and activation simultaneously on both the Routing Engines, reducing commit latency and improving operational efficiency for Junos OS.

The configuration validation is performed on both the Routing Engines, unlike the `fast-synchronize` model prior to Junos OS Release 25.4R1.

You can disable fast-synchronize model by configuring `no-fast-synchronize` option at the `[edit system commit]` hierarchy level.

For MX Series Virtual Chassis systems, when fast-synchronize is the default commit model, the configuration is automatically synchronized across all members and their backup Routing Engines, regardless of whether the `system commit synchronize` option is configured or you explicitly issue the `commit synchronize` command.

**Change History Table**

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
25.4R1	Starting in Junos OS Release 25.4R1, the fast-synchronize commit model is the default commit synchronize model for all Junos OS dual Routing Engine devices, EX Series Virtual Chassis, and MX Series Virtual Chassis.
19.4R1-S1	Starting in Junos OS Evolved Release 19.4R1-S1, <code>commit synchronize</code> is enabled by default on PTX10008. If you issue <code>commit</code> at the <code>[edit system]</code> hierarchy level from the primary routing engine, you see that the backup routing engine is automatically synchronized.
19.4R1	Starting in Junos OS Evolved Release 19.4R1, <code>commit synchronize</code> is enabled by default on PTX10008. If you issue <code>commit</code> at the <code>[edit system]</code> hierarchy level from the primary routing engine, you see that the backup routing engine is automatically synchronized.



# 5

CHAPTER

## Using Operational Commands to Monitor Devices

---

### IN THIS CHAPTER

- CLI Operational Mode Overview | 279
  - Using Operational Commands to Monitor a Device | 286
  - Filtering Operational Command Output | 301
-

# CLI Operational Mode Overview

## IN THIS SECTION

- [CLI Operational Mode Command Overview | 279](#)
- [Display Options of the show Command—An Overview | 282](#)
- [Interface Naming Conventions Used in Operational Commands | 283](#)
- [About Group Interface Names Using Wildcard Characters | 285](#)

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

## CLI Operational Mode Command Overview

### IN THIS SECTION

- [CLI Operational Mode Command Categories | 279](#)
- [Commonly Used Operational Mode Commands | 280](#)

You (the network administrator) can control all network operations using the Junos OS CLI operational mode commands described in this topic.

## CLI Operational Mode Command Categories

CLI operational mode commands fall into the following broad categories:

- 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 system processes, including the routing protocol, interface, and SNMP.
- A command—`request`—Perform system-level operations, including stopping and rebooting the router or switch and loading operating system images.
- A command—`start`—Exit the CLI and start a UNIX shell.
- A command—`configure`—Enter configuration mode, which provides a series of commands that configure the system, 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 enter `?` at the operational mode command prompt to view a list of available commands.

## Commonly Used Operational Mode Commands

The following table lists some operational commands you may find useful for monitoring router or switch operation.

**Table 12: Commonly Used Operational Mode Commands**

Items to Check	Description	Command
Software version	Versions of software running on the router or switch	<code>show version</code>
Log files	Contents of the log files	<code>monitor</code>

**Table 12: Commonly Used Operational Mode Commands** *(Continued)*

Items to Check	Description	Command
	Log files and their contents and recent user logins	show log
Remote systems	Host reachability and network connectivity	ping
	The route to a network system	tracert
Configuration	Current system configuration	show configuration
File manipulation	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

## Display Options of the show Command—An Overview

The `show` command can include `brief`, `detail`, `extensive`, or `terse` options. You can use these and other options to control the amount and type of information 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. At any point in the CLI, you can use the `show` command with one of the following options to display the detail you need to view.

Figure 10: Command Output Options

Command output with the **brief** option.

```

user@host> show interfaces fe-1/1/1 brief
Physical interface: fe-1/1/1, Enabled, Physical link is Down
Link-level type: Ethernet, MTU: 1514, Speed: 100mbps, Loopback:
Disabled, Source filtering: Disabled,
Flow control: Enabled
Device flags : Present Running Down
Interface flags: Hardware-Down SNMP-Traps Internal: 0x4000
Link flags   : None

```

Command output with the **terse** option.

```

user@host> show interfaces fe-1/1/1 terse
Interface      Admin Link Proto  Local      Remote
fe-1/1/1       up      down

```

Command output with the **extensive** option.

```

user@host> show interfaces fe-1/1/1 extensive
Physical interface: fe-1/1/1, Enabled, Physical link is Down
Interface index: 141, SNMP ifIndex: 33, Generation: 24
Link-level type: Ethernet, MTU: 1514, Speed: 100mbps, Loopback:
Disabled, Source filtering: Disabled,
Flow control: Enabled
Device flags : Present Running Down
Interface flags: Hardware-Down SNMP-Traps Internal: 0x4000
Link flags   : None
CoS queues   : 4 supported, 4 maximum usable queues
Hold-times   : Up 0 ms, Down 0 ms
Current address: 00:90:69:d0:f8:9e, Hardware address: 00:90:69:d0:f8:9e
Last flapped : 2007-02-02 09:26:25 PST (2w0d 03:40 ago)
Statistics last cleared: Never
Traffic statistics:
Input bytes :                0                0 bps
Output bytes :                0                0 bps
Input packets:                0                0 pps
Output packets:                0                0 pps
---(more)---

```

## Interface Naming Conventions Used in Operational Commands

### IN THIS SECTION

- [Physical Part of an Interface Name | 283](#)
- [Logical Part of an Interface Name | 284](#)
- [Channel Identifier Part of an Interface Name | 285](#)

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

### Physical Part of an Interface Name

The physical interface naming conventions for Juniper Networks device platforms is as follows:

- On SRX Series Firewalls, 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 following identifiers: interface type, the slot in which the Flexible PIC Concentrator (FPC) is installed, the slot on the FPC in which the PIC is located, and the configured port number.

In the physical part of the interface name, a hyphen (-) separates the media type from the FPC number, and a slash (/) separates the FPC, PIC, and port numbers:

```
type-fpc/pic/port
```



**NOTE:** Exceptions to the *type-fpc/pic/port* physical description include the aggregated Ethernet and aggregated SONET/SDH interfaces, which use the syntax *aenumber* and *asnumber*, respectively.

## Logical Part of an Interface Name

The logical unit part of the interface name corresponds to the logical unit number, which can be a number from 0 through 16,384. You use logical unit numbers to uniquely identify physical storage systems or virtual storage systems within a network. In the virtual part of the name, a period (.) separates the port and logical unit numbers:

- SRX Series Firewalls:


```
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 an 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, you can specify up to three levels of channelization.

A colon (:) separates the physical and virtual parts of the interface name:

- SRX Series Firewalls:

```
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/lpic/port:channel:channel
type-fpc/pic/port:channel:channel:channel
```

## About Group Interface Names Using Wildcard Characters

You can use wildcard characters in 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 13: Wildcard Characters for Specifying Interface Names**

Wildcard Character	Description
* (asterisk)	Match any string of characters in that position in the interface name. For example, so* matches all SONET/SDH interfaces.



**Table 13: Wildcard Characters for Specifying Interface Names** *(Continued)*

Wildcard Character	Description
"[ <i>character</i> < <i>character</i> ...>]"	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.
"[! <i>character</i> < <i>character</i> ...>]"	Match all characters except those included in the brackets. For example, so-"[!03]"* matches all SONET/SDH interfaces except those in slots 0 and 3.
"[ <i>character1</i> - <i>character2</i> ]"	Match a range of characters. For example, so-"[0-3]" * matches all SONET/SDH interfaces in slots 0, 1, 2, and 3.
"[! <i>character1</i> - <i>character2</i> ]"	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.

## Using Operational Commands to Monitor a Device

### IN THIS SECTION

- [CLI Command Completion Example | 287](#)
- [Operational Mode Commands: Overview of Controlling the Scope | 288](#)
- [Viewing Files and Directories on a Device | 290](#)
- [Display Operating System Information | 296](#)
- [Managing Programs and Processes Using Operational Mode Commands | 296](#)
- [CLI Comment Character # for Operational Mode Commands | 301](#)

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

## CLI Command Completion Example

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

This example shows the result of issuing 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>
```

This example shows how to 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
...

```

## Operational Mode Commands: Overview of Controlling the Scope

### IN THIS SECTION

- [Routing Matrix Command Options | 289](#)

The Junos OS CLI operational commands include options that you can use to identify specific components on a device. For example:

- You use the `show interfaces` command to display information about all interfaces on the router.

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 offers 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

## Viewing Files and Directories on a Device

### IN THIS SECTION

- [Directories on the Device | 290](#)
- [List Files and Directories | 291](#)
- [Filenames and URLs | 294](#)

The operating system 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.

### Directories on the Device

The following table lists some standard directories on a Juniper Networks device.

**Table 14: Directories on the Device**

Directory	Description
/config	This directory is located on the device'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 device's hard drive and contains rollback files 4 through 49.
/var/tmp	This directory is located on the device's hard drive. It holds core files from the various processes on the Routing Engines. Core files are generated when a particular process crashes. Juniper Networks engineers use these core files to diagnose the cause of the failure.
/var/log	This directory is located on the device's hard drive. It contains files generated by both the device's logging function and the traceoptions command.

**Table 14: Directories on the Device *(Continued)***

Directory	Description
/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 software 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.

## List 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 the 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/msg/             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
...

```

## Filenames and URLs

You can include a filename in CLI commands and configuration statements such as these:

- file copy
- file archive,
- load,
- save
- set system login user
- *username*
- authentication
- *load-key-file*
- request system software add

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. Neither HTTP nor FTP supports wildcards.



**NOTE:** Only the `file` (`compare` | `copy` | `delete` | `list` | `rename` | `show`) commands support wildcards. When you issue the `file show` command with a wildcard, the command 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 the operating system. The default path is the user's home directory on the remote system. You can also specify *hostname* as *username@hostname*.
- *ftp://hostname/path/filename*—File on an FTP server. You can also specify *hostname* as *username@hostname* or *username:password@hostname*. The default path is the user's home directory.

To specify an absolute path, the path must start with `%2F`; for example, *ftp://hostname/%2Fpath/filename*.

To have the system prompt you for the password, specify `prompt` in place of the password. If a password is required, and you do not specify the password or prompt, an error message is displayed:

```
user@host> file copy ftp://username@ftp.hostname.net//filename
file copy ftp.hostname.net: Not logged in.
user@host> file copy ftp://username:prompt@ftp.hostname.net//filename
Password for username@ftp.hostname.net:
```

- *http://hostname/path/filename*—File on an HTTP server. You can also specify *hostname* as *username@hostname* or *username:password@hostname*. If a password is required and you omit it, you are prompted for it.
- *re0:/path/filename* or *re1:/path/filename*—File on a local Routing Engine.

You can also specify a file on a local Routing Engine for a specific T640 router on a routing matrix:

```
user@host> show log lcc0-re1:chassisd
```

## Display Operating System Information

You can display Junos OS version information and other status to determine if the version of the software that you are running supports specific features or hardware.

To display this 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 software packages installed on the router.  
  
If the Junos Crypto Software Suite is listed, the router has Canada and USA encrypted software. If the Junos Crypto Software Suite is not listed, the router is running worldwide nonencrypted software.
3. To display detailed version information, enter the `show version detail` command. This command displays the hostname and version information for software packages installed on your router. It also includes the version information for each software process.

## Managing Programs and Processes Using Operational Mode Commands

### IN THIS SECTION

- [Show Software Processes | 296](#)
- [Restart the Software Process | 298](#)
- [Stop the Software | 299](#)
- [Reboot the Software | 300](#)

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

### Show 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 15: The `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.

Table 15: The show system process extensive Command Output Fields (*Continued*)

Field	Description
USERNAME	Owner of the process.
WCPU	Weighted CPU usage.

## Restart the Software 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 can interrupt packet forwarding and cause data loss.

To restart a software process:

1. Make sure you are in operational mode.
2. Enter 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 the system 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.



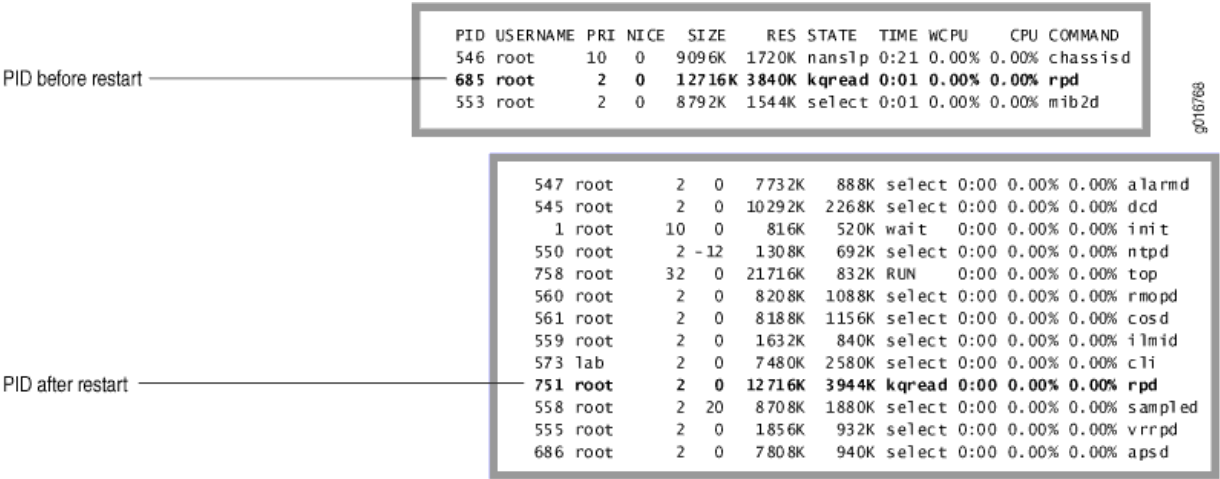
**NOTE:** The gracefully, immediately, and soft options for the restart command are optional and not required for executing the command.

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 identifier (PID) is updated.

Figure 11: Restarting a Process



## Stop the Software



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

You must stop the software on a device through a direct console connection, not through the network. As the software shuts down, the network will go down, and if you were connected that way, you will not see the results output.

To stop the software:

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.
```

## Reboot the Software

After a software upgrade or to recover (occasionally) from an error condition, you must reboot the software. As with the `shutdown` command, you must reboot through a direct console connection otherwise you will not see the command output when the network goes down during the reboot process.

To reboot the software:

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 rebootReboot 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...
```

## CLI Comment Character # for Operational Mode Commands

The comment character enables you to copy operational mode commands that include comments from a file and paste them into the CLI. A pound or hash symbol (#) at the beginning of the command line indicates a comment line. This command is useful for describing frequently used operational mode commands, such as a user's work instructions on how to monitor the network. To add a comment to a command file, you must place # as the first character of the line. When you start a command with #, the operating system disregards the rest of the line.



**NOTE:** The device configuration does not save the comments you enter in the CLI, whether individually or by pasting in the contents of a configuration file. Comments entered at the CLI are ignored.

To add comments in operational mode, you 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

● [About Using the Pipe \(|\) Symbol to Filter Command Output | 302](#)



- [Example: Use Regular Expressions with the Pipe \( | \) Symbol to Filter Command Output | 303](#)
- [Example: Pipe \( | \) Filter Functions in the Command-Line Interface | 305](#)
- [Filter Operational Mode Command Output in a QFabric System | 319](#)
- [Use Suppress-Zero Filter with the Pipe \( | \) Symbol to Filter Zero Values in Command Output | 320](#)

The pipe | symbol lets you (the network administrator) filter the command output in both operational and configuration modes.

## About Using the Pipe ( | ) Symbol to Filter Command Output

You can filter command output by adding the pipe ( | ) symbol when you enter the command.

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 you can use 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, you can combine the pipe symbol and question mark to display an additional compare filter:

```
user@host> show configuration | ?
Possible completions:
  compare          Compare configuration changes with prior version
  ...
```

You can enter any of the pipe filters in combination. For example:

```
user@host> command | match regular-expression | save filename
```



**NOTE:** This topic describes *only* the filters that you can use for *operational mode command* output.

## Example: Use Regular Expressions with the Pipe ( | ) Symbol to Filter Command Output

You use the `except`, `find`, and `match` filters with the pipe symbol to employ regular expressions to filter output. Juniper Networks uses the regular expressions as defined in POSIX 1003.2. If a regular expression contains spaces, operators, or wildcard characters, enclose the expression in quotation marks.

**Table 16: 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 to denote where a match should begin.
\$	Used at the end of an expression to denote 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              REV 09  711-031594  ZW0568        MX80
Midplane            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
```

## Example: Pipe ( | ) Filter Functions in the Command-Line Interface

### IN THIS SECTION

- [Examples of Configurations and Their Differences in Text | 306](#)
- [Examples of Configurations and Their Differences in XML | 308](#)
- [Example of Counting the Number of Lines of Output | 309](#)
- [Example of Output Displayed in XML Tag Format | 309](#)
- [Example of Displaying Static Configuration Data | 309](#)
- [Example of Displaying Ephemeral Configuration Data | 310](#)
- [Example of Displaying Output in JSON Format | 310](#)
- [Example of Displaying the Configuration with YANG Translation Scripts Applied | 311](#)
- [Example of Displaying the RPC Tags for a Command | 312](#)
- [Example of Ignoring Output That Does Not Match a Regular Expression | 313](#)
- [Example of Displaying Output from the First Match of a Regular Expression | 313](#)
- [Example of Retaining Output After the Last Screen | 314](#)
- [Example of Displaying Output Beginning with the Last Entries | 314](#)
- [Example of Displaying Output That Matches a Regular Expression | 315](#)

- [Example of Preventing Output from Being Paginated | 315](#)
- [Example of Sending Command Output to Other Users | 315](#)
- [Example of Resolving IP Addresses | 316](#)
- [Example of Saving Output to a File | 316](#)
- [Example of Appending Output to a File | 317](#)
- [Example of Displaying Output on Screen and Writing to a File | 317](#)
- [Example of Trimming Output by Specifying the Starting Column | 318](#)
- [Example of Refreshing the Output of a Command | 318](#)

This topic describes and provides examples of the pipe ( | ) filter functions that the Junos OS CLI supports.

## Examples of Configurations and Their Differences in Text

The `compare` filter compares the candidate configuration with either the current committed configuration or a configuration file. It also displays the differences between the two configurations with text characters.

To compare configuration files, you enter `compare` after the pipe ( | ) symbol, as follows:

```
show | compare [rollback n | filename]
```

The `rollback n` variable 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`), which is the same as comparing to rollback index 0.

The full path (or URL) to a configuration file is *filename*.

The comparison output uses the following conventions:

- Statements that are in the candidate configuration only are prefixed with a plus sign (+).
- Statements that are in the comparison file only are prefixed with a minus sign (-).
- Statements that are unchanged are prefixed with a single blank space ( ).

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;
+ }
```

We have enhanced output from the `show | compare` command to more accurately reflect configuration changes. This enhancement includes more intelligent handling of order changes in lists. For example, consider group names that are reordered as follows:

```
groups {
    groups {
        group_xmp;      group_xmp;
        group_cmp;      group_grp;
        group_grp;      group_cmp;
    }
}
```

In early 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 {...}
```

## Examples of Configurations and Their 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, you 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 before using the `| compare` command. In configuration mode, you can navigate to a hierarchy where the command is applied.

## Example of Counting the Number of Lines of Output

To count the number of lines in command output, enter `count` after the pipe symbol (`|`). For example:

```
user@host> show configuration | count
Count: 269 lines
```

## Example of Output Displayed in XML Tag Format

To display command output in XML tag format, you 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 CLI displays the equivalent UTF-8 decimal character reference for those characters in the XML output.

## Example of Displaying Static Configuration Data

You can 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. To view this data, you 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 {
  ...
}
```



## Example of Displaying Ephemeral Configuration Data

Juniper Extension Toolkit (JET) applications, Network Configuration Protocol (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 {
...
}
```

## Example of Displaying Output in JSON Format

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 CLI displays the equivalent UTF-8 decimal character reference for those characters in the JSON output.

## Example of Displaying the Configuration with YANG Translation Scripts Applied

You can load YANG modules onto devices running Junos OS to augment the configuration hierarchy with data models that Junos OS does not support natively. Junos OS does support translation of these models. The active configurations and candidate configurations contain the configuration data for non-native YANG data models in the syntax defined by that model. These configurations 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, you use the `| display translation-scripts translated-config` filter in either operational mode or configuration mode.

```
user@host> show configuration | display translation-scripts translated-config
```

In configuration mode, you can display just the configuration differences in the hierarchies corresponding to non-native YANG data models before or after translation scripts are applied. To display those differences, you 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 that the YANG data model defines. The `| display translation-scripts` filter displays the non-native configuration data in both the syntax defined by the YANG data model and the translated Junos OS syntax. Both commands display the entire configuration, which has been truncated for brevity in this example. However, the `show` command returns the pre-inheritance configuration, whereas the `show | display translation-scripts` command returns the post-inheritance configuration.

```
user@host# show
...
myint:intconfig {
```

```

    interfaces {
        interface ge-0/0/0 {
            config {
                description test;
            }
        }
    }
}
...

```

```

user@host# show | display translation-scripts

```

```

...
interfaces {
    ge-0/0/0 {
        description test;
        gigether-options {
            no-flow-control;
        }
    }
}
...
myint:intconfig {
    interfaces {
        interface ge-0/0/0 {
            config {
                description test;
            }
        }
    }
}
...

```

## Example of Displaying the RPC Tags for a Command

To display the remote procedure call (RPC) XML tags for an *operational mode command*, you 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>
```

### Example of 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
```

### Example of Displaying Output from the First Match of a Regular Expression

To display output starting with the first occurrence of text matching a regular expression, you 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
```

## Example of Retaining Output After the Last Screen

You can retain output and scroll or search through it by holding rather than returning immediately to the CLI prompt after viewing the last screen of output. To retain output, you 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
```

## Example of Displaying Output Beginning with the Last Entries

You can view log files in which the end of the file contains the most recent entries. To display text starting from the end of the output, you 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
```



**NOTE:** When the number of lines requested is less than the number of lines that the screen length setting permits you to display, the system returns a subset. The system

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, the system returns the last 19 lines instead of the last 10 lines.

## Example of Displaying Output That Matches a Regular Expression

To display output that matches a regular expression, you 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 {
```

## Example of Preventing Output from Being Paginated

By default, if output is longer than the length of the terminal screen, you receive a `---(more)---` message to display the remaining output. To display the remaining output, you press Space.

To prevent the output from being paginated, you 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 if you want to copy the entire output and paste it into an email message.

## Example of 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, you enter `request message (all | user account@terminal)` after the pipe symbol (|).

If you are troubleshooting your router and talking with a customer service representative on the phone, you can share the command output. You 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 that you enter on your terminal to the terminal of the user `root@tty1`:

```
user@host> show interfaces | request message user root@tty1
```

The user `root@tty1` sees the following output appear on the terminal screen:

```
Message from user@host on /dev/tty0 at 10:32 PST...
Physical interface: dsc, Enabled, Physical link is Up
  Interface index: 5, SNMP ifIndex: 5
  Type: Software-Pseudo, MTU: Unlimited...
```

## Example of 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, you 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
```

## Example of 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 message 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, you 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'
```

## Example of 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, you 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'
```

## Example of 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, you 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>
```

## Example of 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), you 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.

## Example of 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 run the following command:

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

## Filter 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 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 operational mode command. For example:

```
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 operational mode command and specify the list of Node group names in brackets. For example:

```
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

## Use Suppress-Zero Filter with the Pipe ( | ) Symbol to Filter Zero Values in Command Output

Junos OS supports suppress-zero filter to exclude lines with '0' values for any of the fields in given line. This feature is available for all the operational show commands. For show commands CLI output, lines with '0' values output can be masked. The output lines with non-zero values are displayed. You can use this to mask zero counters' values. If a line has non-zero values along with '0' values in a line, it is not masked. If '0' appears in description or string field, it is not masked and is emitted in the CLI output.



**NOTE:** This feature is applicable for CLI operational commands only. The suppress-zeros filter is not applicable for commands that don't use rendering. For example, show configuration, traceroute, ping and other monitor commands.

Here are a few examples for the show commands with suppress-zeroes filter:

- For example, if a command produces the following output:

```
user@host> show interfaces vtep
Physical interface: vtep, Enabled, Physical link is Up
  Interface index: 133, SNMP ifIndex: 517
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited, Speed:
Unlimited
  Device flags   : Present Running
  Interface flags: SNMP-Traps
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
  Input packets  : 0
  Output packets: 0
```

A pipe filter of | suppress-zeros displays the following output:

```
user@host> show interfaces vtep | suppress-zeros
Physical interface: vtep, Enabled, Physical link is Up
  Interface index: 133, SNMP ifIndex: 517
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited, Speed:
Unlimited
  Device flags   : Present Running
  Interface flags: SNMP-Traps
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
```

In this example, the Input packets and Output packets fields are masked as these fields contain '0' integer values.

- If a command produces the following output:

```
user@host> show interfaces fxp0
Physical interface: fxp0, Enabled, Physical link is Up
  Interface index: 8, SNMP ifIndex: 1
  Description: 0
  Type: Ethernet, Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps
  Device flags   : Present Running
  Interface Specific flags: Internal: 0x100000
  Interface flags: SNMP-Traps
  Link type      : Full-Duplex
  Link flags     : 0x4
  Current address: 54:04:0a:dd:85:8d, Hardware address: 54:04:0a:dd:85:8d
  Last flapped   : 2023-11-15 19:02:00 IST (21:40:35 ago)
    Input packets : 1530766
    Output packets: 13469
Logical interface fxp0.0 (Index 5) (SNMP ifIndex 13)
  Flags: Up SNMP-Traps 0x4000000 Encapsulation: ENET2
  Input packets : 1528251
  Output packets: 13481
  Protocol inet, MTU: 1500
    Max nh cache: 75000, New hold nh limit: 75000, Curr nh cnt: 186, Curr new hold cnt: 0, NH
drop cnt: 0
    Flags: Sendbcast-pkt-to-re, Is-Primary
```

Addresses, Flags: Is-Preferred Is-Primary  
 Destination: 10.221.128/18, Local: 10.221.133.141, Broadcast: 10.221.191.255

A pipe filter of | suppress-zeros displays the following output:

```

user@host> show interfaces fxp0 | suppress-zeros
Physical interface: fxp0, Enabled, Physical link is Up
  Interface index: 8, SNMP ifIndex: 1
  Description: 0
  Type: Ethernet, Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps
  Device flags   : Present Running
  Interface Specific flags: Internal: 0x100000
  Interface flags: SNMP-Traps
  Link type      : Full-Duplex
  Link flags     : 0x4
  Current address: 54:04:0a:dd:85:8d, Hardware address: 54:04:0a:dd:85:8d
  Last flapped   : 2023-11-15 19:02:00 IST (21:40:35 ago)
    Input packets : 1530766
    Output packets: 13469
Logical interface fxp0.0 (Index 5) (SNMP ifIndex 13)
  Flags: Up SNMP-Traps 0x4000000 Encapsulation: ENET2
  Input packets : 1528251
  Output packets: 13481
  Protocol inet, MTU: 1500
  Max nh cache: 75000, New hold nh limit: 75000, Curr nh cnt: 186, Curr new hold cnt: 0, NH drop cnt: 0
  Flags: Sendbroadcast-pkt-to-re, Is-Primary
  Addresses, Flags: Is-Preferred Is-Primary
    Destination: 10.221.128/18, Local: 10.221.133.141, Broadcast: 10.221.191.255
  
```

In this example, the Description field is not masked in the output as it is a string type and is a non-integer. The Max nh cache: 75000, New hold nh limit: 75000, Curr nh cnt: 186, Curr new hold cnt: 0, NH drop cnt: 0 line is not masked as it contains non-zero integer counter values in it.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
24.4R1	Starting in Junos OS Release 24.4R1, we've deprecated the compact statement at the [edit system export-format state-data json] hierarchy level.

18.2R1	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 <code>show ephemeral-configuration   display merge</code> command. Starting in Junos OS Release 18.2R1, the <code>display merge</code> option is deprecated.
17.3R1	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 <code>set system export-format state-data json compact</code> . This command converts XML format to compact JSON format. Else, it emits the JSON in non-compact format.
16.2R2	Starting in Junos OS Release 16.2R2, the <code>show   compare   display xml</code> command omits the <code>&lt;configuration&gt;</code> 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.
16.2R2	Starting in Junos OS Release 16.2R2, the <code>show   compare   display xml</code> command omits the <code>&lt;configuration&gt;</code> 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.
16.1	Starting in Junos OS Release 16.1, devices running Junos OS emit JSON-formatted configuration data using a new default implementation for serialization.
16.1	Starting in Junos OS Release 16.1, you can load YANG modules onto devices running Junos OS to augment the configuration hierarchy with data models that are not natively supported by Junos OS but can be supported by translation. 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.
14.2	Starting in Junos OS Release 14.2, you can display the configuration or command output in JavaScript Object Notation (JSON) format by entering <code>display json</code> after the pipe symbol ( <code> </code> ).
8.3	Starting with Junos OS Release 8.3, output from the <code>show   compare</code> command has been enhanced to more accurately reflect configuration changes. This includes more intelligent handling of order changes in lists.



# Configuration Statements and Operational Commands

---

## IN THIS CHAPTER

- [Junos CLI Reference Overview | 325](#)
-

# Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Read this guide to learn about the syntax and options that make up the statements and commands. Also understand the contexts in which you'll use these CLI elements in your network configurations and operations.

- [Junos CLI Reference](#)

Click the links to access Junos OS and Junos OS Evolved configuration statement and command summary topics.

- [Configuration Statements](#)
- [Operational Commands](#)