

Chapter 8

Configuring the Router with the CLI

You configure the JUNOS software by entering configuration mode and creating a hierarchy of configuration mode statements. In configuration mode, the command-line interface (CLI) provides commands to configure the router, load a text (ASCII) file that contains the router configuration, activate a configuration, and save the configuration to a text file.

When you first log on to the router, you enter the CLI operational mode. Operational mode is indicated by the presence of the > prompt, which is preceded by string that defaults to the name of the user and the name of the router. For example:

```
user@host>
```

You enter configuration mode by issuing the `configure` command or the `edit` command from the CLI operation mode. When you do this, the CLI prompt changes from `user@host>` to `user@host#`. Configuration mode is indicated by the presence of the # prompt, which is preceded by string that defaults to the name of the user and the name of the router. For example:

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

To view a list of configuration mode statements, see “Entering Configuration Mode” on page 202. For information about CLI enhancements for the TX Matrix platform and its connected T640 routing nodes, see “Routing Matrix CLI Enhancements” on page 183.

This chapter discusses the following topics:

Configuration Statement Hierarchy on page 199

How the Configuration Is Stored on page 201

Entering Configuration Mode on page 202

Configuration Mode Prompt on page 208

Configuration Mode Banner on page 208

Configuration Statements and Identifiers on page 209

Getting Help About Configuration Mode Commands, Statements, and Identifiers on page 211

Creating and Modifying the Configuration on page 215

Moving Among Levels of the Hierarchy on page 218

Exiting Configuration Mode on page 220

Displaying the Current Configuration on page 221

Displaying set Commands from the Configuration on page 223

Displaying Users Currently Editing the Configuration on page 225

Removing a Statement from the Configuration on page 226

Using Regular Expressions to Remove Related Configuration Items on page 228

Copying a Statement in the Configuration on page 229

Renaming an Identifier on page 231

Inserting a New Identifier on page 231

Running an Operational Mode CLI Command from Configuration Mode on page 234

Displaying Configuration Mode Command History on page 234

Verifying a Configuration on page 235

Committing a Configuration on page 235

Synchronizing Routing Engines on page 241

Saving a Configuration to a File on page 243

Loading a Configuration on page 243

Returning to a Previously Committed Configuration on page 247

Returning to the Rescue Configuration on page 248

Configuration Mode Error Messages on page 249

Deactivating and Reactivating Statements and Identifiers in a Configuration on page 250

Adding Comments in a Configuration on page 251

Having Multiple Users Configure the Software on page 254

Example: Using the CLI to Configure the Router on page 254

Additional Details About Specifying Statements and Identifiers on page 261

For information about the configuration statements to use to configure particular system functionality, see the chapter about that feature.

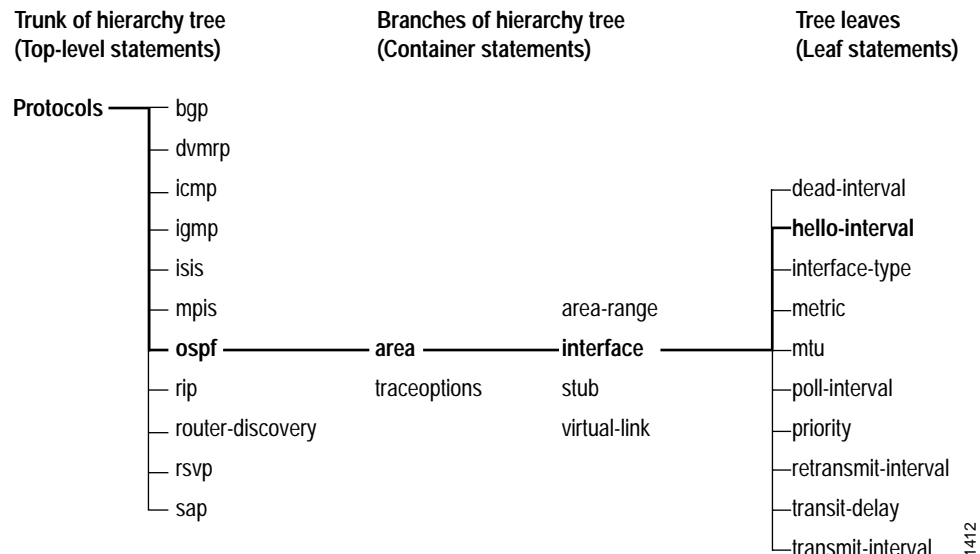
For general guidelines for using the CLI to configure a routing matrix, see “Configuring the Routing Matrix” on page 190.

Configuration Statement Hierarchy

The JUNOS software configuration consists of a hierarchy of *statements*. There are two types of statements: *container statements*, which are statements that contain other statements, and *leaf statements*, which do not contain other statements. All the container and leaf statements together form the *configuration hierarchy*.

Each statement at the top level of the configuration hierarchy resides at the trunk (or root level) of a hierarchy tree. The top-level statements are container statements, containing other statements that form the tree branches. The leaf statements are the leaves of the hierarchy tree. An individual hierarchy of statements, which starts at the trunk of the hierarchy tree, is called a *statement path*. Figure 4 illustrates the hierarchy tree, showing a statement path for the portion of the protocol configuration hierarchy that configures the hello interval on an interface in an Open Shortest Path First (OSPF) area. The protocols statement is a top-level statement at the trunk of the configuration tree. The ospf, area, and interface statements are all subordinate container statements of a higher statement (they are branches of the hierarchy tree), and the hello-interval statement is a leaf on the tree, which, in this case, contains a data value, the length of the hello interval, in seconds.

Figure 4: Configuration Mode Hierarchy of Statements



The CLI represents the statement path shown in Figure 4 as [protocols ospf area *area-number* interface *interface-name*], and displays the configuration as follows:

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

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

The CLI uses this indented representation when it displays the current system configuration, and you use this format when creating ASCII files that contain the software configuration. However, the format of ASCII configuration files is not as strict as the CLI output of the configuration. Although the braces and semicolons are required, the indentation and use of new lines, as shown above, are not required in ASCII configuration files.

How the Configuration Is Stored

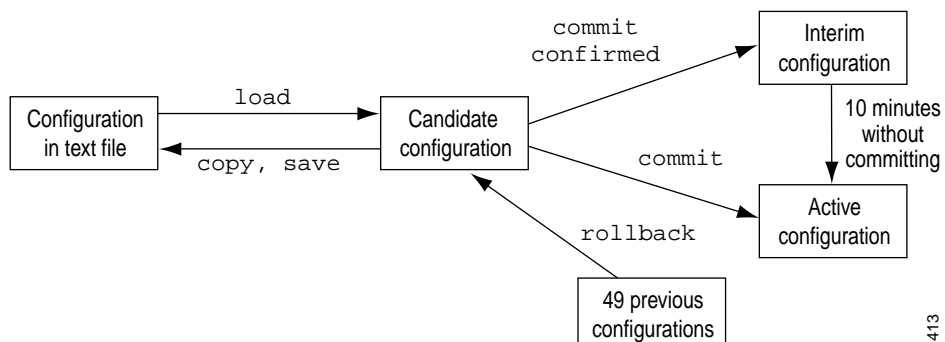
When you edit a configuration, you work in a copy of the current configuration to create a candidate configuration. The changes you make to the candidate configuration are visible in the CLI immediately, so if multiple users are editing the configuration at the same time, all users can see all changes.

To have a candidate configuration take effect, you *commit* the changes. At this point, the candidate file is checked for proper syntax, activated, and marked as the current, operational software configuration file. If multiple users are editing the configuration, when you commit the candidate configuration, all changes made by all the users take effect.

In addition to saving the current configuration, the CLI saves the current operational version and the previous 49 versions of committed configurations. The most recently committed configuration is version 0 (the current operational version, which is the default configuration that the system returns to if you roll back to a previous configuration), and the oldest saved configuration is version 49. The currently operational JUNOS software configuration is stored in the file `juniper.conf`, and the last three committed configurations are stored in the files `juniper.conf.1`, `juniper.conf.2`, and `juniper.conf.3`. These four files are located in the directory `/config`, which is on the router's flash drive. The remaining 46 previous 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.

Figure 5 illustrates the various router configuration states and the configuration mode commands you use to load, commit, copy, save, or roll back the configuration.

Figure 5: Commands for Storing and Modifying the Router Configuration



1413

Entering Configuration Mode

You enter configuration mode by entering the `configure` operational mode command.

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 an ASCII file
quit          Quit from this level
rename        Rename a statement
rollback      Roll back database to last committed version
run           Run an operational-mode command
save          Save configuration to an ASCII file
set           Set a parameter
show          Show a parameter
status        Display database user status
top           Exit to top level of configuration
up            Exit one level of configuration

```

The access privilege level required to enter configuration mode is controlled by the `configure` permission bit. Users for whom this permission bit is not set do not see the `configure` command as a possible completion when they enter a `?` in operational mode, and they cannot enter configuration mode. Users for whom this bit is set do see this command and can enter configuration mode. When in configuration mode, a user can view and modify only those statements for which they have access privileges set. For more information, see “Configuring Access Privilege Levels” on page 376.

This section discusses the following topics:

Using the `Configure` Command on page 203

Using the `Configure Exclusive` Command on page 204

Using the `Configure Private` Command on page 205

Using the Configure Command

If you and other users enter configuration mode with the configure command, everyone can make configuration changes and commit all changes made to the configuration. This means that if you and another user have made configuration changes and the other user commits, the changes you made are committed as well. That is, no one has a lockout on the configuration file.

If, when you enter configuration mode, another user is also in configuration mode, a message shows who the user is and what part of the configuration that user is viewing or editing:

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

If, when you enter configuration mode, the configuration contains changes that have not been committed, a message appears:

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

If, while in configuration mode, you try to make a change while the configuration is locked by another user, a message indicates that the configuration database is locked, who the user is, and what portion of the configuration the user is viewing or editing:

```
user@host# set system host-name ipswitch
error: configuration database locked by:
  user2 terminal d0 (pid 1828) on since 19:47:58 EDT, idle 00:02:11
  exclusive [edit protocols]
```

Using the Configure Exclusive Command

If you enter configuration mode with the `configure exclusive` command, you lock the candidate configuration for as long as you remain in configuration mode, allowing you to make changes without interference from other users. Other users can enter and exit configuration mode, but they cannot change the configuration. If another user has locked the configuration, and you need to forcibly log him or her out, enter the operational mode command `request system logout pid pid_number`. When a user exits from configure exclusive mode when another user is in configure private mode, the JUNOS software will roll back any uncommitted changes.

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

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



NOTE: If you are using the `configure exclusive` command, you cannot exit configuration mode with uncommitted changes while another user is in a configure private session. For more information about the `configure private` command, see “Using the Configure Private Command” on page 205.

Users in configure exclusive mode cannot exit configuration mode with uncommitted changes. A warning message appears notifying the user in configure exclusive mode that any changes will be discarded if the user exits from the configuration:

```
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, the JUNOS software discards your uncommitted changes and rolls back your configuration. The `no` option allows you to continue editing or to commit your changes in configure exclusive mode. These options enforce the restriction that the global configuration must be unmodified for configure private users to commit changes.

Using the Configure Private Command

The configure private command allows multiple users to edit different parts of the configuration at the same time and to commit only their own changes, or to roll back without interfering with one another's changes. When you issue the configure private command, you work in a private candidate configuration, which is a copy of the most recently committed configuration.

When you commit a private candidate configuration, the JUNOS software temporarily locks the global configuration, enforces the restriction that the global configuration must be unmodified to commit private changes, and validates the private candidate configuration. If a merge conflict occurs, the commit fails and the configuration lock is released. You can then modify your private candidate configuration and commit it again. If there are no errors, the changes made in the private candidate configuration are merged into the most recently committed global configuration, are activated, and begin running on the router, and the configuration lock is released.



NOTE: You cannot commit changes in configure private mode when another user is in configure exclusive mode.

If the global configuration has changed, users in configure private mode can issue the rollback or update command to obtain the most recently committed global configuration. For more information about the update command, see “Updating the Configure Private Configuration” on page 207.

You must issue the commit command from the top of the configuration.

You cannot save a configure private session; uncommitted changes are discarded.

You cannot issue the commit confirm command when you are in configure private mode.

Users in configure exclusive mode cannot exit configuration mode with uncommitted changes while another user is in configure private mode. A warning message appears notifying the user in configure exclusive mode that any changes will be discarded if the user exits from the configuration:

```
[edit]
user@host# set system host-name fu

[edit]
user@host# quit
The configuration has been changed but not committed
warning: private edits in use. Auto rollback on exiting 'configure exclusive'
Discard uncommitted changes? [yes,no] (yes)

load complete
Exiting configuration mode
user@host
```

When you use the **yes** option to exit configure exclusive mode, the JUNOS software discards your uncommitted changes and rolls back your configuration. The **no** option allows you to continue editing or to commit your changes in configure exclusive mode. These options enforce the restriction that the global configuration must be unmodified for users to commit configure private changes.



NOTE: You cannot enter configure private mode when the global configuration has been modified.

If a configure private edit is in session, users who issue the configure command can only view the global configuration; a message appears indicating that these users must use the configure exclusive or configure private commands to modify the configuration:

```
[edit]
user@host set system host-name ipswitch
error: private edits in use. Try 'configure private' or 'configure
exclusive'.
[edit]
user@host
```

If the global configuration has been modified, users cannot enter configure private mode because they cannot commit changes when the global configuration has been modified. For example:

```
user@host configure private
error: shared configuration database modified
Users currently editing the configuration:
root terminal d0 (pid 7951) on since 2002-02-21 14:18:46 PST
[edit]
user@host>
```



NOTE: Users in configure or configure exclusive mode cannot exit the global configuration with uncommitted changes.

If another user commits a change to the same section of the configuration that the private user has modified, a merge conflict may result. The JUNOS software then updates the private user's configuration with the most recently committed global configuration and the private user can commit the changes. For example:

```
[edit]
user@host# set system host-name foo

[edit]
user@host# show | compare
[edit system]
- host-name host;
+ host-name foo;
```

```
[edit]
user@host# commit
[edit system host-name]
  'host-name bar'
    statement does not match patch; 'bar' != 'host'
load complete (1 errors)
```

```
[edit]
user@host# show | compare
[edit system]
- host-name bar;
+ host-name foo;
```

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

In this example, after the JUNOS software detects the merge conflict and fixes it, the user in configure private mode issues the `show | compare` command. This command displays the private user's database changes against the most recently committed global configuration.

Updating the Configure Private Configuration

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

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

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



NOTE: You can get merge conflicts when you issue the `update` command.

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

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

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

Configuration Mode Prompt

In configuration mode, the prompt changes from a > to a #. For example:

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

Configuration Mode Banner

The portion of the prompt in braces, [edit], is a *banner*. The banner indicates that you are in configuration mode and shows your location in the statement hierarchy. When you first enter configuration mode, you always are at the top level of the hierarchy, which is indicated by the [edit] banner. For example:

```
user@host> configure
enter configuration mode
[edit] ← Top-level banner
user@host# edit protocols bgp
[edit protocols bgp] ← Banner at the "protocols bgp" hierarchy level
user@host#
```

1463



NOTE: When the word ORPHANED appears to the right of the [edit] banner, it indicates that part of the configuration has been deleted by another user, and you are not working with the most recent global candidate configuration; for example:

```
[edit protocols bgp] ORPHANED
user@host#
```

To refresh your view of the global configuration, move to the top level of the hierarchy.

Configuration Statements and Identifiers

You configure all router properties by including *statements* in the configuration. A statement consists of a keyword, which is fixed text, and, optionally, an *identifier*. An identifier is an identifying name that you define, such as the name of an interface or a username, and that allows you and the CLI to discriminate among a collection of statements.

The following list shows the statements available at the top level of configuration mode (that is, the trunk of the hierarchy tree). Table 8 on page 210 describes each statement.

```

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

```

An angle bracket (>) before the statement name indicates that it is a container statement and that you can define other statements at levels below it.

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

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

```

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

```

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

You must enclose in quotation marks (double quotes) identifiers and any strings that include the following characters: space tab () [] { } ! @ # \$ % ^ & | ' = ?

Table 8: Configuration Mode Top-Level Statements

Statement	Description
access	Configure the Challenge Handshake Authentication Protocol (CHAP). For information about the statements in this hierarchy, see “Access” on page 565.
accounting-options	Configure accounting statistics data collection for interfaces and firewall filters. For information about the statements in this hierarchy, see the <i>JUNOS Network Management Configuration Guide</i> .
chassis	Configure properties of the router chassis, including the clock source, conditions that activate alarms, and SONET/SDH framing and concatenation properties. For information about the statements in this hierarchy, see “Router Chassis” on page 693.
class-of-service	Configure class-of-service parameters. For information about the statements in this hierarchy, see the <i>JUNOS Network Interfaces and Class of Service Configuration Guide</i> .
firewall	Define filters that select packets based on their contents. For information about the statements in this hierarchy, see the <i>JUNOS Policy Framework Configuration Guide</i> .
forwarding-options	Define forwarding options, including traffic sampling options. For information about the statements in this hierarchy, see the <i>JUNOS Network Interfaces and Class of Service Configuration Guide</i> .
groups	Configure configuration groups. For information about statements in this hierarchy, see “Configuration Groups” on page 535.
interfaces	Configure interface information, such as encapsulation, interfaces, virtual channel identifiers (VCIs), and data-link connection identifiers (DLCIs). For information about the statements in this hierarchy, see the <i>JUNOS Network Interfaces and Class of Service Configuration Guide</i> .
policy-options	Define routing policies, which allow you to filter and set properties in incoming and outgoing routes. For information about the statements in this hierarchy, see the <i>JUNOS Routing Protocols Configuration Guide</i> .
protocols	Configure routing protocols, including Border Gateway Protocol (BGP), Intermediate System-to-Intermediate System (IS-IS), Label Distribution Protocol (LDP), Multiprotocol Label Switching (MPLS), OSPF, Routing Information Protocol (RIP), and Resource Reservation Protocol (RSVP). For information about the statements in this hierarchy, see the chapters that discuss how to configure the individual routing protocols in the <i>JUNOS Routing Protocols Configuration Guide</i> and the <i>JUNOS MPLS Applications Configuration Guide</i> .
routing-instances	Configure multiple routing instances. For information about the statements in this hierarchy, see the <i>JUNOS Routing Protocols Configuration Guide</i> .
routing-options	Configure protocol-independent routing options, such as static routes, autonomous system numbers, confederation members, and global tracing (debugging) operations to log. For information about the statements in this hierarchy, see the <i>JUNOS Routing Protocols Configuration Guide</i> .

Statement	Description
security	Configure IP Security (IPSec) services. For information about the statements in this hierarchy see “Security Services” on page 617.
snmp	Configure Simple Network Management Protocol (SNMP) community strings, interfaces, traps, and notifications. For information about the statements in this hierarchy, see the <i>JUNOS Network Management Configuration Guide</i> .
system	Configure systemwide properties, including the hostname, domain name, Domain Name System (DNS) server, user logins and permissions, mappings between hostnames and addresses, and software processes. For information about the statements in this hierarchy, see “System Management Configuration Statements” on page 341.

Getting Help About Configuration Mode Commands, Statements, and Identifiers

Configuration mode provides two different types of help:

Using Command Completion in Configuration Mode on page 211

Getting Help Based on a String in a Statement Name on page 213

Using Command Completion in Configuration Mode

The CLI command completion functions, described in “Using CLI Complete Commands” on page 165, for operational mode commands, 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 the tab key or spacebar.

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

Examples: Using Command Completion in Configuration Mode

List the configuration mode commands:

```

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 an ASCII file
quit        Quit from this level

```

rename	Rename a statement
rollback	Roll back database to last committed version
run	Run an operational-mode command
save	Save configuration to an ASCII file
set	Set a parameter
show	Show a parameter
status	Display database user status
top	Exit to top level of configuration
up	Exit one level of configuration

List all the statements available at a particular hierarchy level:

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

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

List all commands that start with a particular letter or string:

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

```
[edit]
user@host# edit routing-options a
```

List all configured Asynchronous Transfer Mode (ATM) interfaces:

```
user@host# edit interfaces at?
Possible completions:
  <interface_name>  Interface name
  at-2/1/1
  at-2/2/0
  at-5/1/0
[edit]
user@host# edit interfaces at
```

Display a list of all configured policy statements:

```
[edit]
user@host# show policy-options policy-statement ?
Possible completions:
<policy_name>      Name to identify a policy filter
[edit]
user@host# edit policy-options policy-statement
```

Getting Help Based on a String in a Statement Name

In configuration mode, you can use the help command to display help based on a text string contained in a statement name. This command displays help for statements at the current hierarchy level and below.

help apropos *string*

string is a text string about which you want to get help. This string is used to match statement names as well as the help strings that are displayed for the statements. If the string contains spaces, enclose it in quotation marks (" "). You also can specify a regular expression for the string, using standard UNIX-style regular expression syntax.

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

help topic *string*
help reference *string*

The help topic command displays usage guidelines for the statement, while the help reference command displays summary information about the statement.

For introductory information on using the help command, type help and press Enter:

help

Example: Getting Help Based on a String in a Statement Name

Get help about statements that contain the string “traps”:

```
[edit]
user@host# help apropos traps
set interfaces <interface_name>
    Enable SNMP notifications on state changes
set interfaces <interface_name> unit <interface_unit_number>
    Enable SNMP notifications on state changes
set snmp trap-group
    Configure traps and notifications
set snmp trap-group <group_name> version <version> all
    Send SNMPv1 and SNMPv2 traps
set snmp trap-group <group_name> version <version> v1
    Send SNMPv1 traps
set snmp trap-group <group_name> version <version> v2
    Send SNMPv2 traps
set protocols mpls log-updown
    Send SNMP traps
set firewall filter <filter-name> term <rule-name> from source-port snmptrap
    SNMP traps
set firewall filter <filter-name> term <rule-name> from source-port-except snmptrap
    SNMP traps
set firewall filter <filter-name> term <rule-name> from destination-port snmptrap
    SNMP traps
set firewall filter <filter-name> term <rule-name> from destination-port-except
snmptrap
    SNMP traps
set firewall filter <filter-name> term <rule-name> from port snmptrap
    SNMP traps
set firewall filter <filter-name> term <rule-name> from port-except snmptrap
    SNMP traps
[edit]
user@host# edit interfaces at-5/3/0
[edit interfaces at-5/3/0]
user@host# help apropos traps
set <interface_name>
    Enable SNMP notifications on state changes
set <interface_name> unit <interface_unit_number>
    Enable SNMP notifications on state changes
```

Creating and Modifying the Configuration

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

For example, to configure an interface in OSPF area 0, you must configure the following hierarchy of statements:

```

protocols
  ospf
    area 0
      interface interface-name
  
```

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

set—Creates a statement hierarchy 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:

```

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

statement-path is the hierarchy to the configuration statement and the statement itself. If you have already moved to the statement's hierarchy level, you omit this.

statement is the configuration statement itself.

identifier is a string that identifies an instance of a statement. Not all statements require identifiers. In the example shown at the beginning of this section, the area name and the interface names are identifiers. In many cases, the identifier can contain a space. When you type these identifiers in the configuration, you must enclose them in quotation marks. When the CLI displays these identifiers in the output of a show or other command, it encloses them in quotation marks.

The set command is analogous to an operating system command in which you specify the full pathname of the statement you are performing an action on, for example, `mkdir /usr/home/boojum/files` or `mkdir f:\home\boojum\files`.

For statements that can have more than one identifier, when you issue a set command to set an identifier, only that identifier is set. The other identifiers that are specified in the statement remain.

edit—Moves to a particular hierarchy level. If that hierarchy level does not exist, the edit command creates it and then moves to it. After you issue an edit command, the banner changes to indicate your current level in the hierarchy.

The edit command has the following general syntax:

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

The edit command is analogous to the combination of operating system commands that you would use to first change to a directory and then perform an action; for example, `cd /usr/home/boojum;mkdir files`.

Examples: Creating and Modifying the Configuration

To configure an interface to run OSPF, you could issue a single set command from the top level of the configuration hierarchy. The initial [edit] banner indicates that you are at the top level. Notice that after you issue the set command, you remain at the top level of the statement hierarchy, as indicated by the second [edit] banner.

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

You also can use the edit command to create and move to the [edit protocols ospf area 0.0.0.0 interface so-0/0/0] hierarchy level and then issue a set command to set the value of the hello-interval statement. After you issue the edit command, you move down in the hierarchy, as indicated by the [edit protocols ospf area 0.0.0.0 interface so-0/0/0] banner.

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

Because hello-interval is an identifier and not a statement, you cannot use the edit command to set the hello interval value. You must use the set command. You can determine that hello-interval is an identifier by listing the available commands at the [edit protocols ospf area 0.0.0.0 interface so-0/0/0] banner. All the statements *not* preceded by a > are identifiers.

```
[edit]
user@host# edit protocols ospf area 0.0.0.0 interface so-0/0/0
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host# set ?
Possible completions:
+ apply-groups      Groups from which to inherit configuration data
> authentication-key Authentication key
  dead-interval     Dead interval (seconds)
  disable           Disable OSPF on this interface
  hello-interval    Hello interval (seconds)
  interface-type     Type of interface
  metric            Interface metric (1..65535)
> neighbor          NBMA neighbor
```

```

passive          Do not run OSPF, but advertise it
poll-interval    Poll interval for NBMA interfaces
priority         Designated router priority
retransmit-interval Retransmission interval (seconds)
transit-delay    Transit delay (seconds)
transmit-interval OSPF packet transmit interval (milliseconds)
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host# set

```

In both examples above, using either just the set command or a combination of the set and edit commands, you create the same configuration hierarchy:

```

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

```

Notice that the CLI uses indentation to visually represent the hierarchy levels, and it also places braces at the beginning and end of each hierarchy level to set them off. The CLI also places a semicolon at the end of the line that configures the hello-interval statement.

You also use the set command to modify the value of an existing identifier. The following example changes the hello interval in the configuration shown above:

```

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

```

When a statement can have more than one identifier, use the set command to add additional identifiers. Any identifiers that you have already set remain set.

Moving Among Levels of the Hierarchy

When you first enter configuration mode, you are at the top level of the configuration command hierarchy, which is indicated by the [edit] banner:

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

This section discusses the following topics:

Moving Down to a Specific Level on page 218

Moving Back Up to Your Previous Level on page 218

Moving Up One Level on page 219

Moving Directly to the Top of the Hierarchy on page 219

Warning Messages When Moving Up on page 219

Issuing Relative Configuration Commands on page 220

Moving Down to a Specific Level

To move down through an existing configuration command hierarchy, or to create a hierarchy and move down to that level, use the edit configuration mode command, specifying the hierarchy level at which you want to be. After you issue an edit command, the banner changes to indicate your current level in the hierarchy.

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

For example:

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

Moving Back Up to Your Previous Level

To move up the hierarchy, use the exit configuration mode command. This command is, in effect, the opposite of the edit command. That is, the exit command moves you back to your previous level. For example:

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

Moving Up One Level

To move up the hierarchy one level at a time, use the up configuration mode command. For example:

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

Moving Directly to the Top of the Hierarchy

To move directly to the top level, use the top configuration mode command. For example:

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

Warning Messages When Moving Up

If you have omitted a required statement at a particular level, when you issue a show command that displays that hierarchy level, a warning message indicates which statement is missing. For example:

```
[edit protocols mpls]
user@host# set statistics file
[edit protocols mpls]
user@host# show
statistics {
    file; # Warning: missing mandatory statement(s): <filename>
}
interface all;
interface so-3/0/0 {
    disable;
}
```

Issuing Relative Configuration Commands

You can issue configuration mode commands from the top of the hierarchy, or from a level above the area you are configuring. This enables you to perform configurations without having to move from your current location in the hierarchy. To do this, use the `top` or `up` commands followed by another configuration command, including `edit`, `insert`, `delete`, `deactivate`, `annotate`, or `show`.

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

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

To issue configuration mode commands from a location higher in the hierarchy, use the `up` configuration mode command; then specify a configuration command. For example:

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

Exiting Configuration Mode

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

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

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

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

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

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

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

Displaying the Current Configuration

To display the current configuration, use the `show configuration mode` command. This command displays the configuration at the current hierarchy level or at the specified level.

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

When displaying the configuration, the CLI indents each subordinate hierarchy level, inserts braces to indicate the beginning and end of each hierarchy level, and places semicolons at the end of statements that are at the lowest level of the hierarchy. You use the same format when creating an ASCII configuration file, and the CLI uses the same format when saving a configuration to an ASCII file.

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

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

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

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

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

Examples: Displaying the Current Configuration

Display the entire configuration:

```
[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;
      }
    }
  }
}
```

Display a particular hierarchy in the configuration:

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

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

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

Display all of the last committed configuration:

```
[edit]
user@host# set protocols ospf area 0.0.0.0 interface so-0/0/0 hello-interval 5
[edit]
user@host# commit
commit complete
[edit]
user@host# quit
exiting configuration mode
user@host> show configuration
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/0 {
        hello-interval 5;
      }
    }
  }
}
```

Displaying set Commands from the Configuration

In configuration mode, you can display the configuration as a series of configuration mode commands required to recreate 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. For information about the set command, see “Creating and Modifying the Configuration” on page 215.

To display the configuration as a series of configuration mode commands required to recreate 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
```

Example: Displaying set Commands from the Configuration

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

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

To display the configuration as a series of configuration mode commands required to recreate the configuration from the current hierarchy level, issue the `show` configuration mode command with the `| display set relative` option:

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

Example: Displaying Required Set Commands at the Current Hierarchy Level

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

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

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

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

For more information about how to use the `match` option, see “Searching for a String in the Output” on page 171.

Example: Displaying set Commands with the Match Option

Display IP addresses associated with an interface:

```
ge-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 ge-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

```

Displaying Users Currently Editing the Configuration

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

```

user@host# status
Current configuration users:
user terminal p0 (pid 518) on since 2000-03-12 18:24:27 PST
[edit protocols]

```

The system displays who is editing the configuration (user), from where the user is logged in (terminal p0), the date and time the user logged in (2000-03-12 18:24:27 PST), and what level of the hierarchy the user is editing ([edit protocols]).

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

```

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

```

For information about how to schedule a commit, see “Scheduling a Commit” on page 238.

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

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

For more information about configure exclusive, see “Using the Configure Exclusive Command” on page 204.

Removing a Statement from the Configuration

To delete a statement or identifier, use the delete configuration mode command. Deleting a statement or an identifier effectively “unconfigures” the functionality associated with that statement or identifier, returning that functionality to its default condition.

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

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

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

To delete the entire hierarchy starting at the current hierarchy level, do not specify a statement or an identifier in the delete command. When you omit the statement or identifier, a prompt appears asking you 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)
```

Examples: Removing a Statement from the Configuration

Delete the ospf statement, effectively unconfiguring OSPF on the router:

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

Delete all statements from the current level down:

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

Unconfigure a particular property:

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

For information how to use regular expressions to remove related configuration items, see, “Using Regular Expressions to Remove Related Configuration Items” on page 228.

Using Regular Expressions to Remove Related Configuration Items

You can delete related configuration items simultaneously, such as channelized interfaces or static routes, by using a single command and regular expressions. Deleting a statement or an identifier effectively “unconfigures” the functionality associated with that statement or identifier, returning that functionality to its default condition.

You can only delete several parts of the configuration where you normally put multiple items; for example, interfaces. However, you cannot delete “groups” of different items; for example:

```

user@host# show system services
ftp;
rlogin;
rsh;
ssh {
  root-login allow;
}
telnet;

[edit]
user@host# wildcard delete system services *
                                     ^
syntax error.

```

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

To delete related configuration items, issue the wildcard configuration mode command with the delete option and specify the statement path, the items to be summarized with a regular expression, and then the regular expression.

```

user@host# wildcard delete <statement-path> <identifier> <regular-expression>

```



NOTE: When you use the wildcard command to remove related configuration items, the regular expression must be the final statement.

If the JUNOS software matches more than eight related items, the CLI displays only the first eight items.

Example: Deleting Interfaces from the Configuration

Delete multiple T1 interfaces in the range from t1-0/0/0:0 through t1-0/0/0:23:

```
user@host# wildcard delete interfaces t1-1/3/0:.*
matched: t1-1/3/0:0
matched: t1-1/3/0:1
matched: t1-1/3/0:2
Delete 3 objects? [yes,no] (no) no
```

Example: Deleting Routes from the Configuration

Delete static routes in the range from 172.0.0.0 to 172.255.0.0:

```
user@host# wildcard delete routing-options static route 172.*
matched: 172.16.0.0/12
matched: 172.16.14.0/24
matched: 172.16.100.0/24
matched: 172.16.128.0/19
matched: 172.16.160.0/24
matched: 172.17.12.0/23
matched: 172.17.24.0/23
matched: 172.17.28.0/23
...
Delete 13 objects? [yes,no] (no)
```

Copying a Statement in the Configuration

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

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

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

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

Example: Copying a Statement in the Configuration

After you have created one virtual connection (VC) on an interface, copy its configuration to create a second VC:

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

Renaming an Identifier

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

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

Example: Renaming an Identifier

Change the Network Time Protocol (NTP) server address to 10.0.0.6:

```
[edit]  
user@host# rename system network-time server 10.0.0.7 to server 10.0.0.6
```

Inserting a New Identifier

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

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

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

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

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

Examples: Inserting a New Identifier

Insert policy terms in a routing policy configuration. Note that if you do not use the insert command, but rather just configure another term, the added term is placed at the end of the existing list of terms.

```
[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 reject;
    }
    term term4 {
      then accept;
    }
  }
}
[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
```

```

[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 { # reject direct routes
  from protocol direct;
  then reject;
}
term term3 { # reject static routes
  from protocol static;
  then accept;
}
term term4 { #reject local routes
  from protocol local;
  then reject;
}
term term5 { #reject aggregate routes
  from protocol aggregate;
  then reject;
}
term term6 { #accept all other routes
  then accept;
}

```

Insert a transit router in a dynamic MPLS path:

```

[edit protocols mpls path ny-sf]
user@host# show
1.1.1.1;
2.2.2.2;
3.3.3.3 loose;
4.4.4.4 strict;
6.6.6.6;
[edit protocols mpls path ny-sf]
user@host# insert 5.5.5.5 before 6.6.6.6
[edit protocols mpls path ny-sf]
user@host# set 5.5.5.5 strict
[edit protocols mpls path ny-sf]
user@host# show
1.1.1.1;
2.2.2.2;
3.3.3.3 loose;
4.4.4.4 strict;
5.5.5.5 strict;
6.6.6.6;

```

Running an Operational Mode CLI Command from Configuration Mode

At times, you might need to display the output of an operational mode show or other command while configuring the software. While in configuration mode, you can execute a single operational mode command by issuing the configuration mode run command and specifying the operational mode command:

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

Example: Running an Operational Mode CLI Command from Configuration Mode

Display the priority value of the Virtual Router Redundancy Protocol (VRRP) master router while you are modifying the VRRP configuration for a backup router:

```
[edit interfaces ge-4/2/0 unit 0 family inet vrrp-group 27]
user@host# show
virtual-address [ 192.168.1.15 ];
[edit interfaces ge-4/2/0 unit 0 family inet vrrp-group 27]
user@host# run show vrrp detail
Physical interface: ge-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 ge-4/2/0 unit 0 family inet vrrp-group 27]
user@host# set priority ...
```

Displaying Configuration Mode Command History

In configuration mode, you can display a list of the recent commands you issued while in configuration mode. To do this, use the run show cli history command:

```
user@host> configure
...
[edit]
user@host# run show cli history
12:40:08 -- show
12:40:17 -- edit protocols
12:40:27 -- set isis
12:40:29 -- edit isis
12:40:40 -- run show cli history
[edit protocols isis]
user@host#
```

By default, this command displays the last 100 commands issued in the CLI. If you specify a number with the command, it displays that number of recent commands. For example:

```
user@host# run show cli history 3
12:40:08 -- show
12:40:17 -- edit protocols
12:40:27 -- set isis
```

Verifying a Configuration

To verify that the syntax of a 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.

Committing a Configuration

To save software configuration changes to the configuration database and activate the configuration on the router, use the configuration mode `commit` command:

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

The configuration is checked for syntax errors. If the syntax is correct, the configuration is activated and becomes the current, operational router configuration.

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

These sections discuss how to commit configurations:

Committing a Configuration and Exit Configuration Mode on page 236

Activating a Configuration but Requiring Confirmation on page 237

Scheduling a Commit on page 238

Monitoring the Commit Process on page 239

Adding a Comment to Describe the Committed Configuration on page 240

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.

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

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

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



NOTE: After you issue this command, you cannot return to the previous version of the software, because the running and backup copies of the software are identical.

Committing a Configuration and Exit Configuration Mode

To save software configuration changes, activate the configuration on the router, and exit configuration mode, use the commit and-quit configuration mode command. This command succeeds only if the configuration contains no errors.

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

Activating a Configuration but Requiring Confirmation

You can commit the current candidate configuration but require an explicit confirmation for the commit to become permanent. This is useful for verifying that a configuration change works correctly and does not prevent management access to the router. If the change prevents access or causes other errors, the automatic rollback to the previous configuration restores access after the rollback confirmation timeout passes.

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
[edit]
user@host#
```

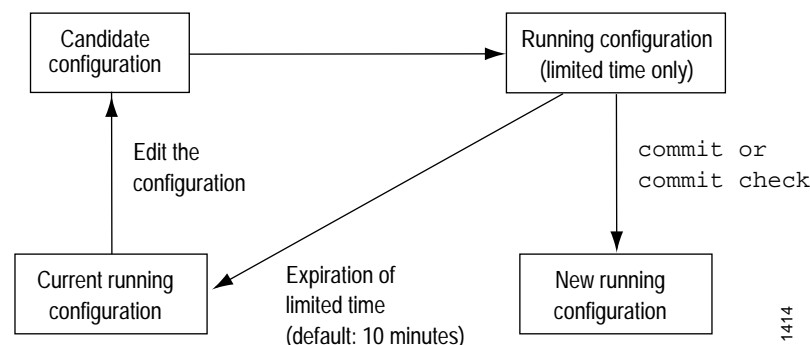
To keep the new configuration active, enter a `commit` or `commit check` command within 10 minutes of the `commit confirmed` command. If the commit is not confirmed within a certain amount of time (10 minutes by default), the JUNOS software automatically rolls back to the previous configuration. For example,

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

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

Figure 6 illustrates how the `commit confirmed` command works.

Figure 6: Confirm a Configuration



1414

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

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

Scheduling a Commit

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

```
[edit]
user@host # commit at <string>
```

`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 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 `yyy-mm-dd hh:mm[:ss]` (year, month, date, hours, minutes, and, optionally, seconds)—Commit the configuration at the specified day and time, which must be after the `commit at` command is issued. Use 24-hour time for the `hh` value. For example, 2003-08-21 12:30:00 is 12:30 PM on August 21, 2003. The time is interpreted with respect to the clock and time zone settings on the router.

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



NOTE: If the JUNOS software fails before the configuration changes become active, all configuration changes are lost.

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

You cannot issue 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 *JUNOS Protocols, Class of Service, and System Basics Command Reference*.

Monitoring the Commit Process

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

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

For example:

```
[edit]
user@host# commit | display detail
2003-09-22 15:39:39 PDT: exporting juniper.conf
2003-09-22 15:39:39 PDT: setup foreign files
2003-09-22 15:39:39 PDT: propagating foreign files
2003-09-22 15:39:39 PDT: complete foreign files
2003-09-22 15:39:40 PDT: copying configuration to juniper.data+
2003-09-22 15:39:40 PDT: dropping unchanged foreign files
2003-09-22 15:39:40 PDT: daemons checking new configuration
2003-09-22 15:39:41 PDT: commit wrapup...
2003-09-22 15:39:42 PDT: activating '/var/etc/ntp.conf'
2003-09-22 15:39:42 PDT: activating '/var/etc/kmd.conf'
2003-09-22 15:39:42 PDT: activating '/var/db/juniper.data'
2003-09-22 15:39:42 PDT: notifying daemons of new configuration
2003-09-22 15:39:42 PDT: signaling 'Firewall daemon', pid 24567, signal 1,
status 0
2003-09-22 15:39:42 PDT: signaling 'Interface daemon', pid 24568, signal 1,
status 0
2003-09-22 15:39:43 PDT: signaling 'Routing protocol daemon', pid 25679,
signal 1, status 0
2003-09-22 15:39:43 PDT: signaling 'MIB2 daemon', pid 24549, signal 1, status
0
2003-09-22 15:39:43 PDT: signaling 'NTP daemon', pid 37863, signal 1, status 0
2003-09-22 15:39:43 PDT: signaling 'Sonet APS daemon', pid 24551, signal 1,
status 0
2003-09-22 15:39:43 PDT: signaling 'VRRP daemon', pid 24552, signal 1, status
0
2003-09-22 15:39:43 PDT: signaling 'PFE daemon', pid 2316, signal 1, status 0
2003-09-22 15:39:43 PDT: signaling 'Traffic sampling control daemon', pid
24553, signal 1, status 0
2003-09-22 15:39:43 PDT: signaling 'IPSec Key Management daemon', pid
24556, signal 1, status 0
2003-09-22 15:39:43 PDT: signaling 'Forwarding UDP daemon', pid 2320, signal
1, status 0
commit complete
```

Adding a Comment to Describe the Committed Configuration

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

```
[edit]
user@host # commit comment <comment-string>
```

comment-string is the text of the comment.



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

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

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

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

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

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

Synchronizing Routing Engines

If your router has two Routing Engines, you can manually direct one Routing Engine to synchronize its configuration with the other by issuing the `commit synchronize` command. The Routing Engine on which you execute this command (requesting Routing Engine) copies and loads its candidate configuration to the other (responding Routing Engine). Both Routing Engines then perform a syntax check on the candidate configuration file being committed. If no errors are found, the configuration is activated and becomes the current operational configuration on both Routing Engines.

For example, if you are logged in to `re1` (requesting Routing Engine) and you want `re0` (responding Routing Engine) to have the same configuration as `re1`, issue the `commit synchronize` command on `re1`. `re1` copies and loads its candidate configuration to `re0`. Both Routing Engines then perform a syntax check on the candidate configuration file being committed. If no errors are found, `re1`'s candidate configuration is activated and becomes the current operational configuration on both Routing Engines.



NOTE: When you issue the `commit synchronize` command, you must use the `group re0` and `re1`. For information about how to use the `apply groups` statement, see “Applying a Configuration Group” on page 539.

The responding Routing Engine must be running JUNOS release 5.0 or higher.

For information about issuing the `commit synchronize` command on a routing matrix, see the “TX Matrix Platform and T640 Routing Node Configuration Guidelines” on page 742.

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

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

Example: Using Apply Groups re0 and re1

The following example shows apply groups re0 and re1 with some configuration data that might be different on re0 and re1:

```

re0 {
  system {
    host-name my_router_RE0;
  }
  interfaces {
    fxp0 {
      unit 0 {
        family inet {
          address 192.168.15.49/24;
        }
        family iso;
      }
    }
  }
}
re1 {
  system {
    host-name my_router_RE1;
  }
  interfaces {
    fxp0 {
      unit 0 {
        family inet {
          address 192.168.15.50/24;
        }
        family iso;
      }
    }
  }
}

```

Example: Setting Apply Groups Re0 and Re1

The following example sets the apply groups re0 and re1:

```

[edit]
user@host# set apply-groups [re0 re1]
[edit]
user@host#

```

Saving a Configuration to a File

You might want to save the configuration to a file so that you can edit it with a text editor of your choice. You can save your current configuration to an ASCII file, which saves the configuration in its current form, including any uncommitted changes. If more than one user is modifying the configuration, all changes made by all users are saved.

To save software configuration changes to an ASCII file, use the save configuration mode command:

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

By default, the configuration is saved to a file in your home directory, which is on the flash drive. For information about specifying the filename, see “Specifying Filenames and URLs” on page 336.

Loading a Configuration

You can create a file, copy the file to the local router, and then load the file in to the CLI. After you have loaded the file, you can commit it to activate the configuration on the router, or you can edit the configuration interactively using the CLI and commit it at a later time.

You can also create a configuration while typing at the terminal and then load it. Loading a configuration from the terminal is generally useful when you are cutting existing portions of the configuration and pasting them elsewhere in the configuration.

To load an existing configuration file that is located on the router, use the load configuration mode command:

```
[edit]
user@host# load (merge | override | patch | replace | update) filename relative)
```

To load a configuration from the terminal, use the following version of the load configuration mode command:

```
[edit]
user@host# load (merge | override | patch | replace | update) terminal relative)
[Type ^D to end input]
```

To replace an entire configuration, specify the override option at any level of the hierarchy.

An override operation discards the current candidate configuration and loads the configuration in *filename* or the one that you type at the terminal. When you use the override option and commit the configuration, all system processes reparse the configuration.

To replace only the configuration that has changed, specify the update option at any level of the hierarchy. An update operation compares the current configuration and the current candidate configuration, and loads only the changes between these configurations in *filename* or the one that you type at the terminal. When you use the update operation and commit the configuration, the JUNOS software attempts to notify the smallest set of system processes that are affected by the configuration change.

To combine the current configuration and the configuration in *filename* or the one that you type at the terminal, specify the merge option. A merge operation is useful when you are adding a new section to an existing 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 portions of a configuration, specify the replace option. For this operation to work, you must include `replace: tags` in the file or configuration you type at the terminal. The software searches for the `replace: tags`, deletes the existing statements of the same name, if any, and replaces them with the incoming configuration. If there is no existing statement of the same name, the replace operation adds to the configuration the statements marked with the `replace: tag`.

To use the merge, replace, or update option without specifying the full hierarchy level, specify the relative option. For example:

```
[edit system]
user@host# show static-host-mapping
bob sysid 987.654.321ab

[edit system]
user@host# load replace terminal relative
{Type ^D at a new line to end input}
replace: static-host-mapping {
    bob sysid 0123.456.789bc;
}
load complete

[edit system]
user@host# show static-host-mapping
bob sysid 0123.456.789bc;
```

To change part of the configuration with a patch file and mark only those parts as changed, specify the patch option.

If, in an override or merge operation, you specify a file or type text that contains `replace: tags`, the `replace: tags` are ignored, and the override or merge operation is performed.

If you are performing a replace operation and the file you specify or text you type does not contain any `replace: tags`, the replace operation is effectively equivalent to a merge operation. This might be useful if you are running automated scripts and cannot know in advance whether the scripts need to perform a replace or a merge operation. The scripts can use the replace operation to cover either case.

For information about specifying the filename, see “Specifying Filenames and URLs” on page 336.

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 *JUNOS Protocols, Class of Service, and System Basics Command Reference*.

Examples: Load a Configuration from a File

Figure 7: Example 1: Load a Configuration from a File

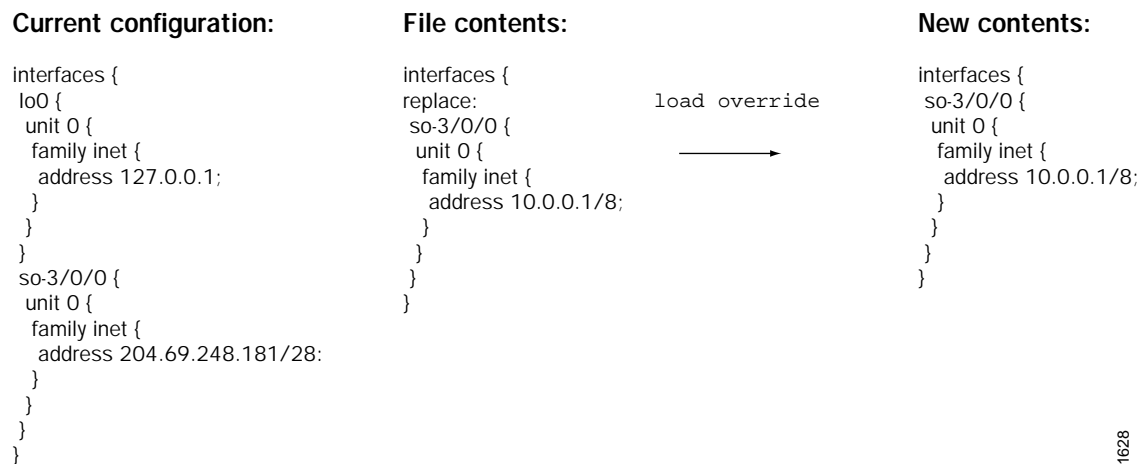


Figure 8: Example 2: Load a Configuration from a File

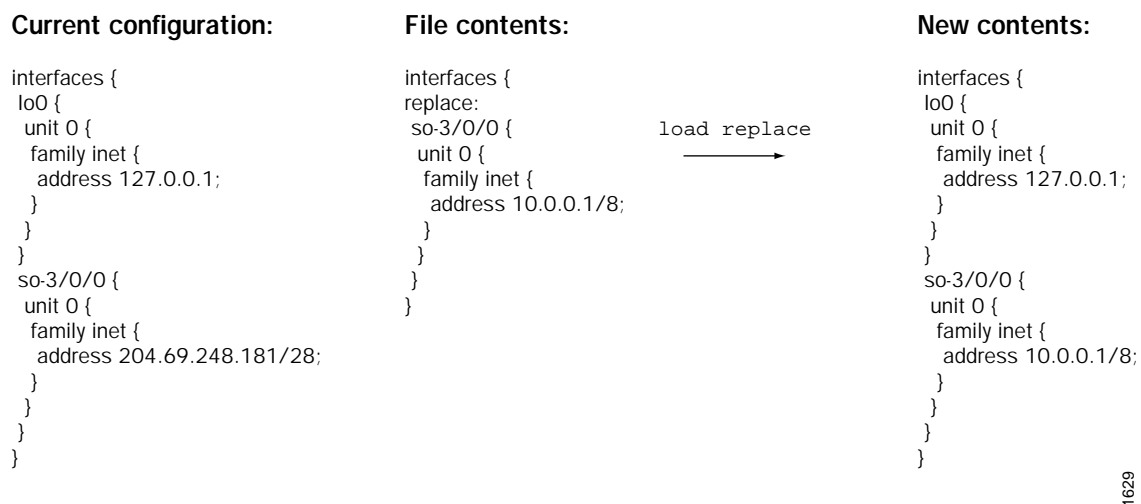
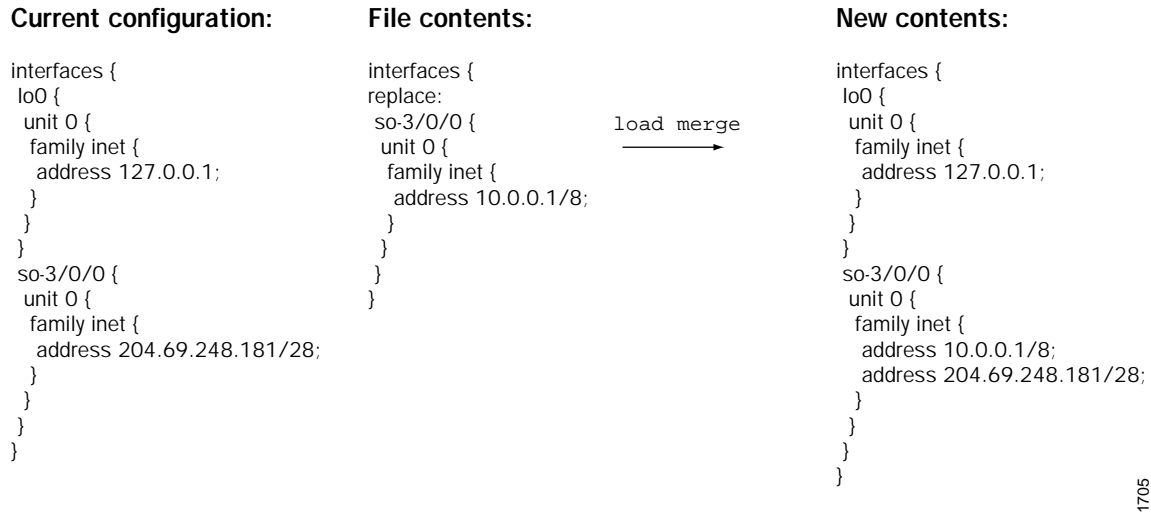


Figure 9: Example 3: Load a Configuration from a File



1705

Figure 10: Example 4: Load a Configuration from a File



1969

Returning to a Previously 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 that you loaded, use the commit command:

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

To return to a configuration prior to the most recently committed one, include the number in the rollback command:

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

number can be a number in the range from 0 through 49. 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.

To display previous configurations, including rollback number, date, time, the name of the user who committed changes, and the method of commit, use the rollback ? command:

```
[edit]
user@host# rollback ?
Possible completions:
<[Enter]> Execute this command
<number> Numeric argument
0 2001-02-27 12:52:10 PST by abc via cli
1 2001-02-26 14:47:42 PST by cde via cli
2 2001-02-14 21:55:45 PST by fgh via cli
3 2001-02-10 16:11:30 PST by hij via cli
4 2001-02-10 16:02:35 PST by klm via cli
| Pipe through a command
[edit]
```

For more information about configuration versions, see “How the Configuration Is Stored” on page 201.

The access privilege level for using the rollback command is controlled by the rollback permission bit. Users for whom this permission bit is not set can return only to the most recently committed configuration. Users for whom this bit is set can return to any prior committed configuration. For more information, see “Configuring Access Privilege Levels” on page 376.

Example: Returning to a Previously Committed Version of the Configuration

Return to and activate the version stored in the file juniper.conf.3:

```
[edit]
user@host# rollback 3
load complete
[edit]
user@host# commit
```

Returning to the Rescue Configuration

You can return to the previously saved rescue configuration at any time.

To return to the rescue configuration, use the rollback rescue configuration mode command:

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

To activate the rescue configuration that you have loaded, use the commit command:

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

To save the most recently committed configuration as the rescue configuration so that you can return to it at any time using either the rollback command, issue the request system configuration rescue save command:

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

To delete an existing rescue configuration, issue the request system configuration rescue delete command:

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

For more information about the request system configuration rescue delete and request system configuration rescue save commands, see the *JUNOS Protocols, Class of Service, and System Basics Command Reference*.

Configuration Mode Error Messages

If you do not type an option for a statement that requires one, a message indicates the type of information expected.

In this example, you need to type an area number to complete the command:

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

In this example, you need to type a value for the hello interval to complete the command:

```
[edit]
user@host# set protocols ospf area 45 interface so-0/0/0
             hello-interval<Enter>
                        ^
syntax error, expecting <data>
```

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

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

Deactivating and Reactivating Statements and Identifiers in a Configuration

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

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

```
deactivate (statement | identifier)
```

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

```
activate (statement | identifier)
```

In both commands, the `statement` or `identifier` you specify must be at the current hierarchy level.

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 deactivate a statement, that specific object or property is completely ignored and is not applied at all when you issue a commit command. When you disable a functionality, it is activated when you issue a commit command but is treated as though it is down or administratively disabled.

Examples: Deactivating and Reactivating Statements and Identifiers in a Configuration

Deactivate an interface in the configuration:

```
[edit interfaces]
user@host# show
at-5/2/0 {
  traceoptions {
    traceflag all;
  }
  atm-options {
    vpi 0 maximum-vcs 256;
  }
  unit 0 {
    ...
  }
}
[edit interfaces]
user@host# deactivate at-5/2/0
[edit interfaces]
user@host# show
inactive: at-5/2/0 {
  traceoptions {
    traceflag all;
  }
}
...
```

Reactivate the interface:

```
[edit interfaces]
user@host# activate at-5/2/0
[edit interfaces]
user@host# show
at-5/2/0 {
    traceoptions {
        traceflag all;
    }
}
...
```

Adding Comments in a Configuration

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

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 ""
```

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.

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).

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.

When configuring interfaces, you can add comments about the interface by including the `description` statement at the `[edit interfaces interface-name]` hierarchy level. Any comments you include appear in the output of the `show interfaces` commands. For more information about the `description` statement, see the *JUNOS Network Interfaces and Class of Service Configuration Guide*.

Examples: Including Comments in Configurations

Add comments to a configuration:

```
[edit]
user@host# show
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/0 {
        hello-interval 5;
      }
    }
  }
}
[edit]
user@host# edit protocols ospf
[edit protocols ospf]
user@host# set area 0.0.0.0
user@host# annotate area 0.0.0.0 "Backbone area configuration added June 15,
1998"
[edit protocols ospf]
user@host# edit area 0.0.0.0
[edit protocols ospf area 0.0.0.0]
user@host# annotate interface so0 "Interface from router sj1 to router sj2"
[edit protocols ospf area 0.0.0.0]
user@host# top
```

```

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

```

The following excerpt from a configuration example illustrates how to enter 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 */
}

```

Having Multiple Users Configure the Software

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

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

Example: Using the CLI to Configure the Router

This section walks through an example of creating a simple configuration, illustrating how to use the CLI to create, display, and modify the software configuration for your system. The example used in this section creates the following configuration:

```

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 this entire configuration with 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 Example

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

First, you enter configuration mode by issuing the configure top-level command:

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

The prompt in braces shows that you are in configuration edit mode, at the top of the hierarchy. If you want to create the above configuration, you start by editing the protocols ospf statements:

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

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#
```

Next, add the first interface:

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

You now have four nested statements. Next, set the hello and dead intervals. Note that command completion (enter a tab or space) and context-sensitive help (type a question mark) are always available.

```
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host# set ?
Possible completions:
+ apply-groups      Groups from which to inherit configuration data
> authentication-key Authentication key
  dead-interval     Dead interval (seconds)
  disable           Disable OSPF on this interface
  hello-interval    Hello interval (seconds)
  interface-type    Type of interface
  metric            Interface metric (1..65535)
> neighbor         NBMA neighbor
  passive           Do not run OSPF, but advertise it
  poll-interval     Poll interval for NBMA interfaces
  priority          Designated router priority
  retransmit-interval Retransmission interval (seconds)
  transit-delay     Transit delay (seconds)
  transmit-interval OSPF packet transmit interval (milliseconds)
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host# set hello-interval 5
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host# set dead-interval 20
[edit protocols ospf area 0.0.0.0 interface so-0/0/0]
user@host#
```

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

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

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

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

Now, 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#
```

Now, back up to the top level and see what you have:

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

This configuration now contains the statements you want. Before committing it, which activates the configuration, verify that the configuration is correct:

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

Now you can commit the configuration to activate it on the router:

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

Suppose you decide to use different dead and hello intervals on interface so-0/0/1. You can make changes to the configuration. You can go directly to the appropriate hierarchy level by typing the full hierarchy path to the statement you want to edit.

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

If you change your mind and decide not to run OSPF on the first interface, you can 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#
```

Note that everything inside of the statement you deleted was deleted with it. You could eliminate the entire OSPF configuration by simply entering `delete protocols ospf` while at the top level.

Suppose you decide to use the default values for the hello and dead intervals on your remaining interface, but you want OSPF to run on that interface:

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

You can set multiple statements at the same time as long as they are all part of the same hierarchy (the path of statements from the top inward, as well as one or more statements at the bottom of the hierarchy). Doing this can reduce considerably the number of commands you must enter. For example, if you want to go back to the original hello and dead interval timers on interface so-0/0/1, you can 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#
```

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

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

Additional Details About Specifying Statements and Identifiers

This section provides more detailed information about specifying statements and identifiers in configuration mode:

Specifying Statements on page 261

Performing CLI Type-Checking on page 263

Specifying Statements

This section provides more detailed information about CLI container and leaf statements so that you can better understand how the CLI displays them in a configuration and how you must specify them when creating ASCII configuration files.

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-name > < identifier > {
    statement;
    additional-statements;
}
```

Statement name, identifier, and a single identifier:

```
< statement-name > < identifier > identifier;
```

The *statement-name* is the name of the statement. In the configuration example shown in the previous section, *ospf* and *area* are statement names.

The *identifier* is a name or other string that uniquely identifies an instance of a statement. The identifier is used when a statement can be specified more than once in a configuration. In the configuration example shown in the previous section, the identifier for the *area* statement is 0 and the identifier for the *interface* statement is so-0/0/0.

When specifying a statement, you must specify either a statement name or an identifier, or both, depending on the statement hierarchy.

You specify identifiers in one of the following ways:

identifier—The *identifier* is a flag, which is a single keyword.

identifier value—The *identifier* is a keyword, and the *value* is a required option variable.

identifier [value1 value 2 value3 ...]—The *identifier* is a set that accepts multiple values. The brackets are required when you specify a set of identifiers; however, they are optional when you specify only one identifier.

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),
            interface so-0/0/0 {# which contains an interface named "so-0/0/0."
                hello-interval 25;# Identifier and value (identifier-name value).
                priority 2;      # Identifier and value (identifier-name value).
                disable;        # Flag identifier (identifier-name).
            }
            interface so-0/0/1;# Another instance of "interface," named so-0/0/1,
        }                    # this instance contains no data, so no braces
    }                        # are displayed.
}

policy-options {
    # Top-level statement (statement-name).
    term term1 {
        # Statement under "policy-options"
        # (statement-name value).
        from {
            # Statement under "term" (statement-name).
            route-filter 10.0.0.0/8 orlonger reject;# One identifier ("route-filter") with
            route-filter 127.0.0.0/8 orlonger reject;# multiple values.
            route-filter 128.0.0.0/16 orlonger reject;
            route-filter 149.20.64.0/24 orlonger reject;
            route-filter 172.16.0.0/12 orlonger reject;
            route-filter 191.255.0.0/16 orlonger reject;
        }
        then {
            # Statement under "term" (statement-name).
            next term;    # Identifier (identifier-name).
        }
    }
}

```

When you create an ASCII configuration file, you can specify statements and identifiers in one of the following ways. However, each statement has a preferred style, and the CLI uses that style when displaying the configuration in response to a configuration mode show command.

Statement followed by identifiers:

```
statement-name identifier-name [...] identifier-name value [...];
```

Statement followed by identifiers enclosed in braces:

```
statement-name {
    identifier-name;
    [...]
    identifier-name value;
    [...]
}
```

For some repeating identifiers, you can use one set of braces for all the statements:

```
statement-name {
    identifier-name value1;
    identifier-name value2;
}
```

Performing CLI Type-Checking

When you specify identifiers and values, the CLI expects to receive specific types of input and 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 checks that you entered an address in a valid format. If you have not, an error message indicates what you were expected to type. Table 9 lists the data types the CLI checks.

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>	Correct: so-0/0/1 Incorrect: so-0
Full interface name	<i>type-fpc/pic/port<:channel>.logical</i>	Correct: so-0/0/1.0 Incorrect: so-0/0/1
Full or abbreviated interface name (used in places other than the edit interfaces hierarchy)	<i>type-<fpc</pic/port>><<:channel>.logical></i>	Correct: so, so-1, so-1/2/3:4.5
IP address	<i>0xhex-bytes</i> <i>octet< .octet< .octet< .octet>>></i>	Correct: 1.2.3.4, 0x01020304, 128.8.1, 128.8 Sample translations: 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</length></i> <i>octet< .octet< .octet< .octet>>></length></i> <i>></i>	Correct: 10/8, 128.8/16, 1.2.3.4/32, 1.2.3.4 Sample translations: 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

Data Type	Format	Examples
International Organization for Standardization (ISO) address	<i>hex-nibble<hex-nibble ...></i>	<p>Correct: 47.1234.2345.3456.00, 47123423453456.00, 47.12.34.23.45.34.56.00</p> <p>Sample translations: 47123456 becomes 47.1234.56 47.12.34.56 becomes 47.1234.56 4712.3456 becomes 47.1234.56</p>
OSPF area identifier (ID)	<i>0xhex-bytes octet< .octet< .octet.<octet>>> decimal-number</i>	<p>Correct: 54, 0.0.0.54, 0x01020304, 1.2.3.4</p> <p>Sample translations: 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</p>