

# Chapter 7

## Configure the Router with the CLI

To configure the router, including the routing protocols, the router interfaces, network management, and user access, you must enter a separate mode, called configuration mode, by issuing the configure operational mode command.

In configuration mode, the 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.

This chapter discusses the following topics:

Configuration Statement Hierarchy on page 78

How the Configuration Is Stored on page 80

Enter Configuration Mode on page 81

Configuration Mode Prompt on page 82

Configuration Mode Banner on page 82

Configuration Statements and Identifiers on page 82

Get Help about Configuration Mode Commands, Statements, and Identifiers on page 84

Create and Modify the Configuration on page 87

Move among Levels of the Hierarchy on page 90

Exit Configuration Mode on page 92

Display the Current Configuration on page 93

Remove a Statement from the Configuration on page 95

Copy a Statement in the Configuration on page 96

Rename an Identifier on page 97

Insert a New Identifier on page 98

Run an Operational Mode CLI Command from Configuration Mode on page 100

Display Configuration Mode Command History on page 100

- Verify a Configuration on page 101
- Commit a Configuration on page 101
- Save a Configuration to a File on page 103
- Load a Configuration on page 103
- Return to a Previously Committed Configuration on page 106
- Configuration Mode Error Messages on page 107
- Deactivate and Reactivate Statements and Identifiers in a Configuration on page 108
- Add Comments in a Configuration on page 109
- Have Multiple Users Configure the Software on page 111
- Walk-through Example: Using the CLI to Configure the Router on page 112
- Additional Details about Specifying Statements and Identifiers on page 117

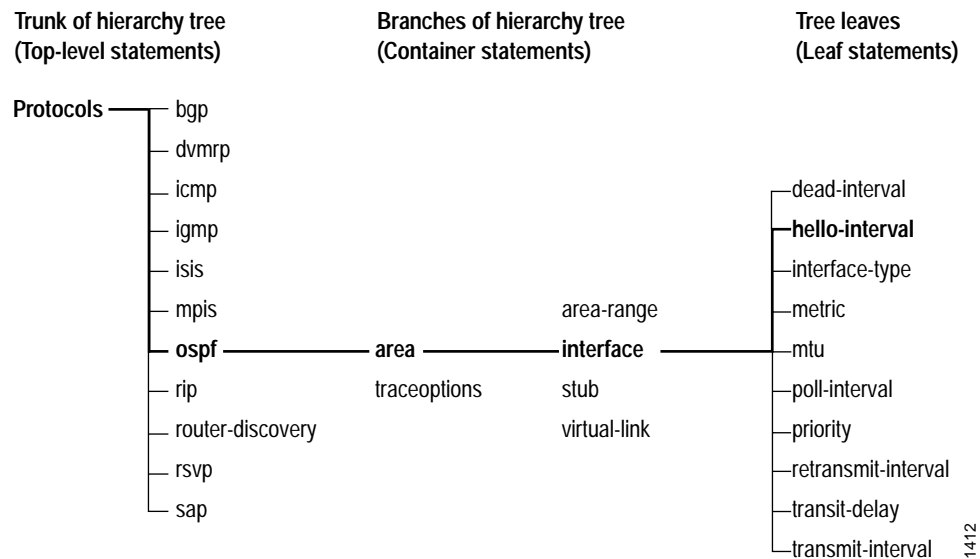
For information about the configuration statements to use to configure particular system functionality, see the chapter about that feature.

## 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 metaphorical 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 3 illustrates the hierarchy tree, showing a statement path for the portion of the protocol configuration hierarchy that configures the hello interval on an interface in an OSPF area. The protocols statement is a top-level statement at the trunk of the configuration tree. The ospf, area, and interface statements are all subordinate container statements of a higher statement (they are branches of the hierarchy tree), and the hello-interval statement is a leaf on the tree, which, in this case, contains a data value, the length of the hello interval in seconds.

Figure 3: Configuration Mode Hierarchy of Statements



The CLI represents the statement path shown in Figure 3 as [protocols ospf area *area-number* interface *interface-name*], and it 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 close 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. While 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

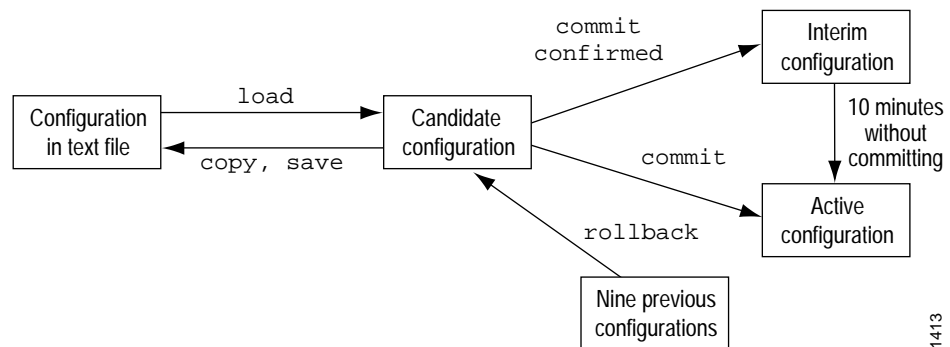
When you edit a configuration, you work in a copy of the current configuration to create a candidate configuration. The changes you make to the candidate configuration are visible in the CLI immediately, so if multiple users are editing the configuration at the same time, all users can see all changes.

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 nine 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 9. 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 six previous versions of committed configurations, the files `juniper.conf.4` through `juniper.conf.9`, are stored in the directory `/var/db/config` on the hard disk.

Figure 4 illustrates the various states of router configuration and the configuration mode commands that you use to act upon the configuration.

Figure 4: Commands for Storing and Modifying the Router Configuration



1413

## Enter 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

```

For a complete list of the configuration commands, see “Complete Configuration Statement Hierarchy” on page 26.

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 “Configure Access Privilege Levels” on page 206.

If, when you enter configuration mode, another user is also in configuration mode, a message indicates who the user is and what portion of the configuration the 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>
```

## 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
entering configuration mode
[edit]
user@host# edit protocols bgp
[edit protocols bgp]
user@host#
```

← Top-level banner

← Banner at the "protocols bgp" hierarchy level

## 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 user name, 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 4 on page 83 describes each statement.

```

user@host# set ?
Possible completions:
+ 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-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 an SDH/SONET interface that is on the FPC in slot 0, in the first PIC location, and in the first port on the PIC. For other identifiers, such as interface descriptive text and policy and firewall term names, you can specify a name of your choice. These identifiers can include any characters.

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

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

Statement	Description
chassis	Configure properties of the router chassis, including the clock source, conditions that activate alarms, and SDH/SONET framing and concatenation properties. For information about the statements in this hierarchy, see the <i>JUNOS Internet Software Configuration Guide: Interfaces and Chassis</i> .
class-of-service	Configure class-of-service parameters. For information about the statements in this hierarchy, see the <i>JUNOS Internet Software Configuration Guide: Interfaces and Chassis</i> .
firewall	Define filters that select packets based on their contents. For information about the statements in this hierarchy, see the <i>JUNOS Internet Software Configuration Guide: Interfaces and Chassis</i> .
forwarding-options	Define forwarding options, including traffic sampling options. For information about the statements in this hierarchy, see the <i>JUNOS Internet Software Configuration Guide: Interfaces and Chassis</i> .

Statement	Description
groups	Configure configuration groups. For information about statements in this hierarchy, see "Configuration Groups" on page 121.
interfaces	Configure interface information, such as encapsulation, interfaces, VCs, and DLCIs. For information about the statements in this hierarchy, see the <i>JUNOS Internet Software Configuration Guide: Interfaces and Chassis</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 Internet Software Configuration Guide: Routing and Routing Protocols</i> .
protocols	Configure routing protocols, including BGP, IS-IS, OSPF, RIP, MPLS, LDP, and RSVP. For information about the statements in this hierarchy, see the chapters that discuss how to configure the individual routing protocols in the <i>JUNOS Internet Software Configuration Guide: Routing and Routing Protocols</i> and the <i>JUNOS Internet Software Configuration Guide: Traffic Engineering</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 Internet Software Configuration Guide: Routing and Routing Protocols</i> .
snmp	Configure SNMP community strings, interfaces, traps, and notifications. For information about the statements in this hierarchy, see "Configure SNMP" on page 283.
system	Configure systemwide properties, including the host name, domain name, DNS name server, user logins and permissions, mappings between host names and addresses, and software processes. For information about the statements in this hierarchy, see "System Management Configuration Statements" on page 187.

## Get Help about Configuration Mode Commands, Statements, and Identifiers

Configuration mode provides two different types of help:

Use Command Completion in Configuration Mode on page 84

Get Help Based on a String in a Statement Name on page 86

### **Use Command Completion in Configuration Mode**

When you are in configuration mode, the CLI command completion functions described in the section "Have the CLI Complete Commands" on page 58 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, and to complete a command or statement that you have partially typed, press a tab or space.

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. However, to complete an identifier, you must press tab. This scheme allows you to enter identifiers with similar names and then press the space key when you are done typing the identifier name.

**Examples: Use 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:
> 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-options  Protocol-independent routing option configuration
> snmp         Simple Network Management Protocol
> system       System parameters
[edit]
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 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 t3

```

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# show policy-options policy-statement

```

## Get Help Based on a String in a Statement Name

In configuration mode, you can display help based on a text string contained in a statement name using the help command. 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.

**Example: Get Help Based on a String Contained 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
```

**Create and Modify 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 area-name
      interface interface-name
```

To create the hierarchy and hence configure the router, 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 general syntax:

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

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

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

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

## Examples: Create and Modify 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.

## Move 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] prompt:

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

### ***Move 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.

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

For example:

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

## ***Move 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#
```

## ***Move 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#
```

## ***Move 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;
}
```

## Exit Configuration Mode

To exit from configuration mode from any level, 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, when you try to exit from configuration mode, 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>
```

## Display 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.

```
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. This is the same format that you use when creating an ASCII configuration file, and it is the same format that the CLI uses 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: Display 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;
            }
        }
    }
}
```

## Display 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]).

## Remove 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.

```
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: Remove 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;
}
```

## Copy 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:

```
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: Copy 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;
    }
  }
}
```

## Rename 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:

```
rename <statement-path> identifier1 to identifier2
```

## Example: Rename an Identifier

Change the network time (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
```

## Insert 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 Multiprotocol Label Switching (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:

```
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: Insert 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# 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# edit policy-options policy-statement statics term4
```

```

[edit policy-options policy-statement statics term4]
user@host# set from protocol local
[edit policy-options policy-statement statics term4]
user@host# set then reject
[edit policy-options policy-statement statics term4]
user@host# edit policy-options policy-statement statics term5
[edit policy-options policy-statement statics term5]
user@host# set from protocol aggregate
[edit policy-options policy-statement statics term5]
user@host# set then reject
[edit policy-options policy-statement statics term5]
user@host# top
[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; # reject the prefixes in the route list
    }
    term term2 { # reject direct routes
      from protocol direct;
      then reject;
    }
    term term3 { # reject static routes
      from protocol static;
      then reject;
    }
    term term4 { # reject local routes
      from protocol local;
      then reject;
    }
    term term5 { # reject aggregate routes
      from protocol aggregate;
      then reject;
    }
    term term6 {
      then accept; # accept all other routes
    }
  }
}

```

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;
```

## Run 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: Run an Operational Mode CLI Command from Configuration Mode**

Display the priority value of the 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 ...
```

## Display 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
```

## Verify 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.

## Commit a Configuration

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

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 have committed the configuration and are satisfied that the new configuration is successfully running, 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 causes the root file system to be backed up to /altroot, and /config to be backed up 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 drive..

**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.

## ***Commit 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>
```

## ***Activate a Configuration for a Limited Time***

There might be times when you want to commit a candidate configuration, activating the configuration for a limited time to determine that it functions properly, reverting to the previous configuration afterwards. One case when you might want to do this is to ensure that a modified configuration does not lock you out of the command-line interface or the router.

To activate a configuration for a limited time, use the commit confirmed configuration mode command:

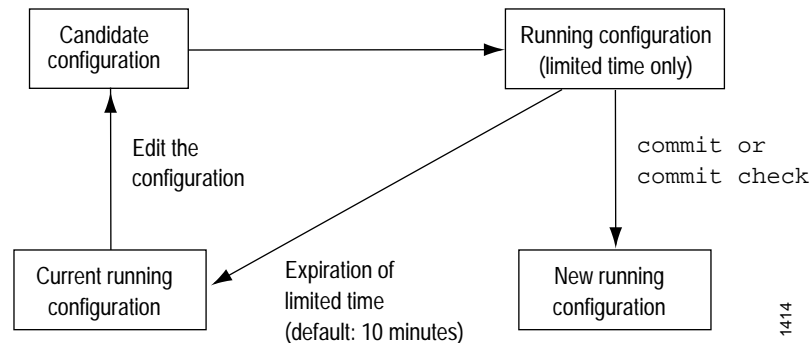
```
[edit]
user@host# commit confirmed
commit complete
[edit]
user@host#
```

As with 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. By default, the configuration runs for 10 minutes. Then, the router reverts to the previous configuration.

To keep the new configuration active, enter a commit or commit check command within 10 minutes of the commit confirmed command.

Figure 5 illustrates how the commit confirmed command works.

Figure 5: Confirm a Configuration



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

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

## Save a Configuration to a File

You can save the software configuration from your current configuration session to an ASCII file. Doing so saves the configuration in its current form, including any uncommitted changes. If more than one user is modifying the configuration, saving it captures all changes made by all users.

You might want to save the configuration to a file so that you can edit it with a text editor of your choice.

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 that file in your home directory, which is on the flash disk. For information about specifying the filename, see “How to Specify Filenames” on page 184.

## Load a Configuration

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

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

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

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

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

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

To replace an entire configuration, specify the override option. An override operation discards the current candidate configuration and loads the configuration in *filename* or the one that you type at the terminal.

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 `replace: tag`.

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

If, when you are performing a replace operation, 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 “How to Specify Filenames” on page 184.

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 Internet Software Command Reference*.

**Examples: Load a Configuration from a File**

Current configuration:

```

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

```

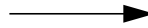
File contents:

```

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

```

load override



New contents:

```

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

```

Current configuration:

```

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

```

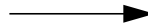
File contents:

```

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

```

load replace

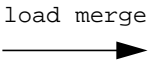


New contents:

```

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

```

Current configuration:	File contents:		New contents:
<pre> interfaces {   lo0 {     unit 0 {       family inet {         address 127.0.0.1;       }     }   }   so-3/0/0 {     unit 0 {       family inet {         address 204.69.248.181/28;       }     }   } } </pre>	<pre> interfaces {   replace:   so-3/0/0 {     unit 0 {       family inet {         address 10.0.0.1/8;       }     }   } } </pre>	<p>load merge</p> 	<pre> interfaces {   lo0 {     unit 0 {       family inet {         address 127.0.0.1;       }     }   }   fxp 0 {     unit 0 {       family inet {         address 10.0.0.1/8;         address 204.69.248.181/28;       }     }   } } </pre>

## Return 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 0 through 9. 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 9.

For more information about versions of the configuration, see “How the Configuration Is Stored” on page 80.

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 “Configure Access Privilege Levels” on page 206.

### **Example: Return 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
```

## 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'
    }
  }
}
```

## Deactivate and Reactivate 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 being down or administratively disabled.

### **Examples: Deactivate and Reactivate 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;
  }
}
...
```

## Add 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:

```
annotate statement "comment-string"
```

*statement* is the configuration statement to which you are attaching the comment and 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:

```
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 Internet Software Configuration Guide: Interfaces and Chassis*.

## Examples: Include 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 */
}

```

## Have 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.

## Walk-through 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 thus 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#
```

Note that the interface statement has come into view because you have popped up to inside 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 causes the configuration to be activated, 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. Note that you jump all the way down through the hierarchy 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#
```

Next, if you change your mind and decide not to run OSPF on the first interface at all, you can just 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.

Next, you decide to use the default values for the hello and dead intervals on our 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#
```

Note that you can set multiple statements at the same time so 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 that must be entered. For instance, 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 re-create the other interface, as you had it before, with only a single entry:

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

## Additional Details about Specifying Statements and Identifiers

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

How to Specify Statements on page 117

How the CLI Performs Type Checking on page 119

### ***How to Specify 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 statement name. In the configuration example shown in the previous section, ospf and area are statements.

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 variable option.

*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;
}
```

### How the CLI Performs 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 have typed is in the correct format. For example, for a statement in which you must specify an IP address, the CLI checks that you have typed an address in a valid format. If you have not, an error message indicates what you were expected to type. Table 5 lists the data types that the CLI checks.

Table 5: CLI Configuration Input Types

Data Type	Format	Examples
Physical interface name (used in the edit interfaces hierarchy)	<i>type-fpc/pic/port</i>	<b>Correct:</b> so-0/0/1 <b>Incorrect:</b> so-0
Full interface name	<i>type-fpc/pic/port&lt;:channel&gt;.logical</i>	<b>Correct:</b> so-0/0/1.0 <b>Incorrect:</b> so-0/0/1
Full or abbreviated interface name (used in places <i>other</i> than the edit interfaces hierarchy)	<i>type-&lt;fpc/&gt;pic/&gt;port&gt;&gt;&lt;&lt;:channel&gt;.logical&gt;</i>	<b>Correct:</b> so, so-1, so-1/2/3:4.5
IP address	<i>0xhex-bytes</i> <i>octet&lt;.octet&lt;.octet.&lt;octet&gt;&gt;&gt;</i>	<b>Correct:</b> 1.2.3.4, 0x01020304, 128.8.1, 128.8  <b>Sample translations:</b> 1.2.3 becomes 1.2.3.0 0x01020304 becomes 1.2.3.4 0x010203 becomes 0.1.2.3

Data Type	Format	Examples
IP address (destination prefix) and prefix length	<i>O</i> hex-bytes</length> <i>o</i> ctet<. <i>o</i> ctet<. <i>o</i> ctet.< <i>o</i> ctet>>></length>	<b>Correct:</b> 10/8, 128.8/16, 1.2.3.4/32, 1.2.3.4 <b>Sample translations:</b> 1.2.3 becomes 1.2.3.0/32 0x01020304 becomes 1.2.3.4/32 0x010203 becomes 0.1.2.3/32 default becomes 0.0.0.0/0
ISO address	<i>h</i> ex-nibble< <i>h</i> ex-nibble ...>	<b>Correct:</b> 47.1234.2345.3456.00, 47123423453456.00, 47.12.34.23.45.34.56.00 <b>Sample translations:</b> 47123456 becomes 47.1234.56 47.12.34.56 becomes 47.1234.56 4712.3456 becomes 47.1234.56
OSPF area identifier (ID)	<i>O</i> hex-bytes <i>o</i> ctet<. <i>o</i> ctet<. <i>o</i> ctet.< <i>o</i> ctet>>> <i>d</i> ecimal-number	<b>Correct:</b> 54, 0.0.0.54, 0x01020304, 1.2.3.4 <b>Sample translations:</b> 54 becomes 0.0.0.54 257 becomes 0.0.1.1 128.8 becomes 128.8.0.0 0x010203 becomes 0.1.2.3