

Chapter 6

Using Commands and Statements to Configure the Router

This chapter describes how to use the CLI to configure the router.

Topics include:

- Understanding CLI Configuration Mode on page 75
- Entering and Exiting Configuration Mode on page 80
- Modifying the Configuration on page 82
- Verifying a Configuration on page 98
- Committing a Configuration on page 98
- When Multiple Users Configure the Software on page 105
- Displaying set Commands from the Configuration on page 111
- Displaying Additional Information About the Configuration on page 114

Understanding CLI Configuration Mode

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

As described in “Understanding CLI Command Modes” on page 4, a router configuration is stored as a hierarchy of statements. In configuration mode, you create the specific hierarchy of configuration statements that you want to use. When you have finished entering the configuration statements, you commit them, which activates the configuration on the router.

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

Topics in this section include:

- Configuration Mode Commands on page 76
- Configuration Statements and Identifiers on page 77
- Configuration Statement Hierarchy on page 79

Configuration Mode Commands

Table 11 summarizes each CLI configuration mode command. The commands are organized alphabetically.

Table 11: Summary of Configuration Mode Commands (1 of 2)

Command	Description
activate	Remove the <code>inactive: tag</code> from a statement, effectively reading the statement or identifier to the configuration. Statements or identifiers that have been activated take effect when you next issue the <code>commit</code> command.
annotate	Add comments to a configuration. You can add comments only at the current hierarchy level.
commit	Commit the set of changes to the database and cause the changes to take operational effect.
copy	Make a copy of an existing statement in the configuration.
deactivate	Add the <code>inactive: tag</code> to a statement, effectively commenting out the statement or identifier from the configuration. Statements or identifiers marked as inactive do not take effect when you issue the <code>commit</code> command.
delete	Delete a statement or identifier. All subordinate statements and identifiers contained within the specified statement path are deleted with it.
edit	Move inside the specified statement hierarchy. If the statement does not exist, it is created.
exit	Exit the current level of the statement hierarchy, returning to the level prior to the last edit command, or exit from configuration mode. The <code>quit</code> and <code>exit</code> commands are synonyms.
help	Display help about available configuration statements.
insert	Insert an identifier into an existing hierarchy.
load	Load a configuration from an ASCII configuration file or from terminal input. Your current location in the configuration hierarchy is ignored when the load operation occurs.
quit	Exit the current level of the statement hierarchy, returning to the level prior to the last edit command, or exit from configuration mode. The <code>quit</code> and <code>exit</code> commands are synonyms.
rename	Rename an existing configuration statement or identifier.
replace	Replace identifiers or values in a configuration.
rollback	Return to a previously committed configuration. The software saves the last 10 committed configurations, including the rollback number, date, time, and name of the user who issued the <code>commit</code> configuration command.

Table 11: Summary of Configuration Mode Commands (2 of 2)

Command	Description
run	Run a top-level CLI command without exiting from configuration mode.
save	Save the configuration to an ASCII file. The contents of the current level of the statement hierarchy (and below) are saved, along with the statement hierarchy containing it. This allows a section of the configuration to be saved, while fully specifying the statement hierarchy.
set	Create a statement hierarchy and set identifier values. This is similar to edit except that your current level in the hierarchy does not change.
show	Display the current configuration.
status	Display the users currently editing the configuration.
top	Return to the top level of configuration command mode, which is indicated by the [edit] banner.
up	Move up one level in the statement hierarchy.
update	Update a private database.
wildcard	Delete a statement or identifier.

For more information about configuration mode commands, see “Summary of CLI Configuration Mode Commands” on page 203.

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, which allows you and the CLI to discriminate among a collection of statements.

Table 12 describes top-level CLI configuration mode statements.

Table 12: Configuration Mode Top-Level Statements (1 of 2)

Statement	Description
access	Configure the Challenge Handshake Authentication Protocol (CHAP). For information about the statements in this hierarchy, see the <i>JUNOS System Basics Configuration Guide</i> .
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 conditions that activate alarms and SONET/SDH framing and concatenation properties. For information about the statements in this hierarchy, see the <i>JUNOS System Basics Configuration Guide</i> .
class-of-service	Configure class-of-service parameters. For information about the statements in this hierarchy, see the <i>JUNOS 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> .

Table 12: Configuration Mode Top-Level Statements (2 of 2)

Statement	Description
forwarding-options	Define forwarding options, including traffic sampling options. For information about the statements in this hierarchy, see the <i>JUNOS Network Interfaces Configuration Guide</i> .
groups	Configure configuration groups. For information about statements in this hierarchy, see the <i>JUNOS System Basics Configuration Guide</i> .
interfaces	Configure interface information, such as encapsulation, interfaces, virtual channel identifiers (VCIs), and data-link connection identifiers (DLCIs). For information about the statements in this hierarchy, see the <i>JUNOS Network Interfaces 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 Policy Framework 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> .
security	Configure IP Security (IPSec) services. For information about the statements in this hierarchy see the <i>JUNOS System Basics Configuration Guide</i> .
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 the <i>JUNOS System Basics Configuration Guide</i> .

For specific information on configuration statements, see the JUNOS configuration guides.

The CLI represents the statement path shown in Figure 14 on page 79 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.

Entering and Exiting Configuration Mode

You configure the JUNOS software by entering configuration mode and creating a hierarchy of configuration mode statements.

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

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

```

user@host> configure
entering configuration mode
[edit]
user@host# ?
Possible completions:
<[Enter]>      Execute this command
activate      Remove the inactive tag from a statement
annotate      Annotate the statement with a comment
commit        Commit current set of changes
copy          Copy a statement
deactivate    Add the inactive tag to a statement
delete        Delete a data element
edit          Edit a sub-element
exit          Exit from this level
help          Provide help information
insert        Insert a new ordered data element
load          Load configuration from 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

Users must have **configure** permission to view and use the **configure** command. When in configuration mode, a user can view and modify only those statements for which they have access privileges set. For more information, see the *JUNOS System Basics Configuration Guide*.

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

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

Up to 32 users can be in configuration mode simultaneously, and they all can make changes to the configuration at the same time. For more information, see “When Multiple Users Configure the Software” on page 105.

- 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>
```

Modifying the Configuration

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

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

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

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

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

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

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

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

Topics in this section include:

- Displaying the Current Configuration on page 83
- Adding Configuration Statements and Identifiers on page 85
- Deleting a Statement from the Configuration on page 86
- Copying a Statement in the Configuration on page 88
- Issuing Relative Configuration Commands on page 89
- Renaming an Identifier on page 89
- Inserting a New Identifier on page 90

- Deactivating and Reactivating Statements and Identifiers on page 92
- Adding Comments in a Configuration on page 94
- Omitting Portions of the Hierarchy When Displaying a Configuration on page 96

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>
```

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
```

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

```
## Last commit: 2006-07-18 11:21:58 PDT by echen
version 8.28.2
```

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
## Last commit: 2006-08-10 11:21:58 PDT by user
version 8.2
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/0 {
        hello-interval 5;
      }
    }
  }
}

```

Adding 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 which you define, such as the name of an interface or a username, and which allows you and the CLI to discriminate among a collection of statements.

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

```

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

```

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

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

```

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

```

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

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

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>.
```

Deleting 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, you are prompted to confirm the deletion:

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

Examples: Deleting 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 about how to use regular expressions to delete related configuration items, see “Example 3: Using Global Replace in a Configuration” on page 159.

Copying a Statement in the Configuration

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

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

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

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

Example: Copying a Statement in the 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;
    }
}

```

Issuing Relative Configuration Commands

You can quickly move to the top of the hierarchy or to a level above the area you are configuring. 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 up in the hierarchy, use the **up** configuration mode command; specify the number of levels you want to move up the hierarchy and then specify a configuration command. For example:

```

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

```

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. Also note that you must create the term, as shown in this example, before you can place it with the `insert` command.

```
[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 {
    from protocol direct;
    then reject;
}
term term3 {
    from protocol static;
    then accept;
}
term term4 {
    from protocol local;
    then reject;
}
term term5 {
    from protocol aggregate;
    then reject;
}
term term6 {
    then accept;
}

```

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;
```

Deactivating and Reactivating Statements and Identifiers

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

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 comments 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 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 */
}

```

Omitting Portions of the Hierarchy When Displaying a Configuration

You can use the `apply-flags omit` configuration statement to control which parts of the configuration hierarchy appear when you use the `show` command to display a configuration. This feature is useful for hiding portions of the configuration hierarchy that include lengthy or redundant configuration statements.

To omit a portion of the hierarchy from `show` command output, include the `apply-flags omit` configuration statement at any hierarchy level:

```
apply-flags omit;
```

The portion of the hierarchy that contains the statement is omitted from `show` command output. Although the hierarchy is omitted from `show` command output, it is not omitted from the actual configuration.

Example: Omitting Portions of the Hierarchy When Displaying a Configuration

In the following configuration,

```
[edit system services]
apply-flags omit;
ftp;
rlogin;
rsh;
telnet;
```

output from the `show` command appears as follows:

```
[edit]
user@host# show system
domain-search [ domain1.net spglab.domain2.net ];
time-zone America/Los_Angeles;
authentication-order [ password radius ];
```

```

name-server {
    192.168.5.68;
    172.17.28.101;
}
radius-server {
    192.168.170.241 secret
"$9$615r/u1SyKvWXB1EyreLX-VwgJGjH.zn9s2PTz6tp"; # SECRET-DATA
    192.168.64.10 secret
"$9$rR7I87ws4oJUx7dsYgGUHqmT36Ct0hyeP5BRhrMW"; # SECRET-DATA
}
services { /* OMITTED */};

[edit]
user@host#

```

To override the `apply-flags omit` configuration statement and display an omitted configuration, use one of the following methods:

- Use the `show <statement-path>` command where `<statement-path>` specifies the omitted configuration. For example:

```

[edit]
user@host# show system services
apply-flags omit;
ftp;
rlogin;
rsh;
telnet;

```

- Include the `|` (pipe) display omit option with the `show` command. For example:

```

[edit system]
user@host# show | display omit
domain-search [ domain1.net spglab.domain2.net ];
time-zone America/Los_Angeles;
authentication-order [ password radius ];
name-server {
    192.168.5.68;
    172.17.28.101;
}
radius-server {
    192.168.170.241 secret
"$9$615r/u1SyKvWXB1EyreLX-VwgJGjH.zn9s2PTz6tp"; ## SECRET-DATA
    192.168.64.10 secret
"$9$rR7I87ws4oJUx7dsYgGUHqmT36Ct0hyeP5BRhrMW"; ## SECRET-DATA
}
services {
    apply-rules omit;
    ftp;
    rlogin;
    rsh;
    telnet;
}

```

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 `commit` configuration mode command:

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

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

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

These sections discuss how to commit configurations:

- [Committing a Configuration and Exiting Configuration Mode on page 99](#)
- [Activating a Configuration but Requiring Confirmation on page 100](#)
- [Scheduling a Commit Operation on page 101](#)
- [Monitoring the Commit Process on page 102](#)
- [Adding a Comment to Describe the Committed Configuration on page 103](#)
- [Updating the Alternate Boot Drive on page 104](#)

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.



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

```
load merge
load replace
load override
load update
```

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

Committing a Configuration and Exiting Configuration Mode

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

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

Activating a Configuration but Requiring Confirmation

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

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

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

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

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

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

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

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

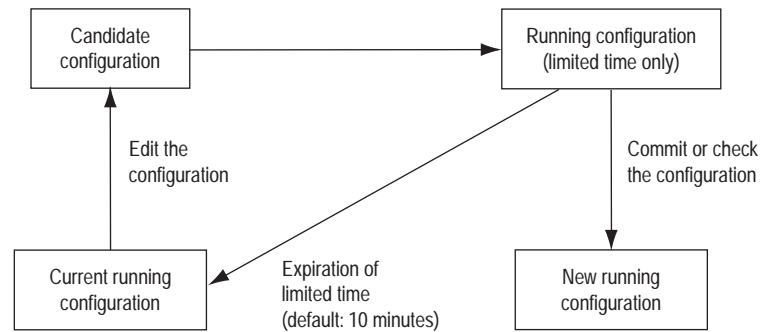
To show when a rollback is scheduled after a `commit confirmed` command, enter the `show system commit` command. For example:

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

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

Figure 15 illustrates how the `commit confirmed` command works.

Figure 15: 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 Operation

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

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

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



NOTE: If 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 System Basics and Services 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 do so, include the `commit comment` statement. The comment can be as long as 512 bytes and you must type it on a single line.

```

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

```

comment-string is the text of the comment.



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

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

```

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

```

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

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

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

Updating the Alternate Boot Drive

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

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



NOTE: To back up the file system on a J-series Services Router, you must specify a media type (primary compact flash drive, removable compact flash drive, or USB storage device) for backup. For more information, see the *J-series Services Router Administration Guide*.

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

When Multiple Users Configure the Software

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

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

Topics in this section include:

- Forms of the Configure Command on page 105
- Example: Using the configure Command on page 106
- Displaying Users Currently Editing the Configuration on page 107
- Using the configure exclusive Command on page 108
- Using the configure private Command on page 109

Forms of the configure Command

The JUNOS software supports three forms of the `configure` command: `configure`, `configure private`, and `configure exclusive`. These forms control how users edit and commit configurations and can be useful when multiple users configure the software. See Table 13.

Table 13: Forms of the configure Command (1 of 2)

Command	Edit Access	Commit Access
<code>configure</code>	<ul style="list-style-type: none"> ■ No one can lock the configuration. All users can make configuration changes. ■ When you enter configuration mode, the CLI displays the following information: <ul style="list-style-type: none"> ■ A list of other users editing the configuration. ■ Hierarchy levels the users are viewing or editing. ■ Whether the configuration has been changed, but not committed. 	<ul style="list-style-type: none"> ■ No one can lock the configuration. All users can commit all changes to the configuration. ■ If you and another user make changes and the other user commits changes, your changes are committed as well.

Table 13: Forms of the configure Command (2 of 2)

Command	Edit Access	Commit Access
configure exclusive	<ul style="list-style-type: none"> ■ One user locks the configuration and makes changes without interference from other users. ■ Other users can enter and exit configuration mode, but they cannot change the configuration. ■ If you enter configuration mode while another user has locked the configuration (with the configure exclusive command), the CLI displays the user and the hierarchy level the user is viewing or editing. ■ If you enter configuration mode while another user has locked the configuration, you can forcibly log out that user with the request system lockout operational mode command. For details, see the <i>JUNOS System Basics and Services Command Reference</i>. 	
configure private	<ul style="list-style-type: none"> ■ Multiple users can edit the configuration at the same time. ■ Each user has a private candidate configuration to edit independently of other users. 	<ul style="list-style-type: none"> ■ When you commit the configuration, the router verifies that the operational (running) configuration has not been modified by another user before accepting your private candidate configuration as the new operational configuration. ■ If the configuration has been modified by another user, you can merge the modifications into your private candidate configuration and attempt to commit again.

Example: Using the configure Command

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>
    
```

Displaying Users Currently Editing the Configuration

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

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

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

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

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

For information about how to schedule a commit, see “Scheduling a Commit Operation” on page 101.

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 the configure exclusive mode, see “Using the configure exclusive Command” on page 108.

Using the `configure exclusive` Command

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

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

If you enter configuration mode and another user is also in configuration mode and has locked the configuration, a message 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]
```

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

```
user@host> configure exclusive
warning: uncommitted changes will be discarded on exit
Entering configuration mode

[edit]
user@host# set system host-name cool

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

warning: discarding uncommitted changes
load complete
Exiting configuration mode
```

When you use the `yes` option to exit `configure exclusive` mode, 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.

When a user exits from `configure exclusive` mode while another user is in `configure private` mode, the JUNOS software will roll back any uncommitted 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.



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

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 shared configuration. For more information about the `update` command, see “Updating the Configure Private Configuration” on page 111.

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 confirmed` command when you are in `configure private` mode.

Following an upgrade or downgrade of JUNOS software, you must use normal configuration mode or `configure exclusive` mode to commit a configuration before you can use the `configure private` command.

If a `configure private` edit is in session, other 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 private` 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. In this case, the JUNOS software updates the private user's configuration with the most recently committed global configuration and then allows the private user to 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 shared configuration.

Updating the Configure Private Configuration

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

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

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



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

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

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

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

Displaying set Commands from the Configuration

In configuration mode, you can display the configuration as a series of configuration mode commands required to re-create the configuration. This is useful if you are not familiar with how to use configuration mode commands or if you want to cut, paste, and edit the displayed configuration. For information about the **set** command, see “Displaying the Current Configuration” on page 83.

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

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

Example: Displaying Required set Commands at the Current Hierarchy Level

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

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

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

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

Example: Displaying set Commands with the Match Option

Display IP addresses associated with an interface:

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

Displaying Additional Information About the Configuration

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

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

For example:

```
[edit]
user@host# show | display detail
##
## version: Software version information
## require: system
##
version "3.4R1 [tlim]";
system {
##
## host-name: Host name for this router
## match: ^[:alnum:]._]+$
## require: system
##
host-name router-name;
##
## domain-name: Domain name for this router
## match: ^[:alnum:]._]+$
## require: system
##
domain-name isp.net;
##
## backup-router: Address of router to use while booting
##
backup-router 192.168.100.1;
root-authentication {
##
## encrypted-password: Encrypted password string
##
encrypted-password "$1$BYJQE$/ocQof8pmcm7MSGK0"; # SECRET-DATA
}
##
## name-server: DNS name servers
## require: system
##
name-server {
##
## name-server: DNS name server address
##
208.197.1.0;
}
login {
##
## class: User name (login)
## match: ^[:alnum:]._]+$
##
}
```

```

class super-user {
  ##
  ## permissions: Set of permitted operation categories
  ##
  permissions all;
}
...
##
## services: System services
## require: system
##
services {
  ## services: Service name
  ##
  ftp;
  ##
  ## services: Service name
  ##
  telnet;
  ##
}
syslog {
  ##
  ## file-name: File to record logging data
  ##
  file messages {
    ##
    ## Facility type
    ## Level name
    ##
    any notice;
    ##
    ## Facility type
    ## Level name
    ##
    authorization info;
  }
}
}
chassis {
  alarm {
    sonet {
      ##
      ## lol: Loss of light
      ## alias: loss-of-light
      ##
      lol red;
    }
  }
}
}
}

```

```
interfaces {
  ##
  ## Interface name
  ##
  at-2/1/1 {
    atm-options {
      ##
      ## vpi: Virtual path index
      ## range: 0 .. 255
      ## maximum-vcs: Maximum number of virtual circuits on this VP
      ##
      vpi 0 maximum-vcs 512;
    }
    ##
    ## unit: Logical unit number
    ## range: 0 .. 16384
    ##
    unit 0 {
      ##
      ## vci: ATM point-to-point virtual circuit identifier ([vpi.]vci)
      ## match: ^([[:digit:]]+){0,1}[[:digit:]]+$
      ##
      vci 0.128;
    }
  }
}
...
```