Introduction to Junos and the Command Line Interface (CLI)

Section 2: A Little Background on Junos

2.1. Junos Platforms

Before we get started on how you use Junos, let's take a few minutes to give you some background on how it works. While this information is helpful to working with Junos, none of this information is critical. So if you want to get right to the heart of the course, feel free to skip ahead.

First, not every Juniper Networks device runs Junos. Let's take a look at those devices that do.

Junos platforms come in many shapes and sizes, span switching, routing, and security, and are well suited for a variety of network environments.

Click the links onscreen to find out more about each Junos platform.

If you'd like to view a comparison of the throughput range or the application areas of all the Junos platforms, click the links under the ‘learn more’ heading.

The M-series multiservice routers range from over 7 Gbps up to over 320 Gbps of throughput, and are deployed in both high-end enterprise and service-provider environments. Large enterprises typically deploy M-series routers in a number of different roles, including Internet gateway router, WAN connectivity router, campus core router, and as regional backbone and data center routers. In service-provider environments, the M-series router is deployed predominantly as a multiservice edge router, but it can also be deployed in small and medium cores, peering, route reflector, multicast, mobile and data-center applications.

For additional information on the M-series multiservice routers, click the URL onscreen.

The T-series core routers range from 320 Gbps to over 2.5 Tbps of throughput. The T-series family is ideal for service provider environments and is deployed within the core of those networks.

For additional information on the T-series core routers, click the URL onscreen.

The J-series services routers range from 600 Mbps up to over 2 Gbps of throughput, and are deployed at branch and remote locations in the network to provide all-in-one secure WAN connectivity, IP telephony, and connection to local PCs and servers through integrated Ethernet switching.

For additional information on the J series services routers, click the URL onscreen.

The MX-series Ethernet services routers range from 240 Gbps up to over 960 Gbps, and are targeted for dense dedicated access aggregation and provider edge services in medium and large Points of Presence (or POPs). Large enterprise environments and service providers can leverage MX-series Ethernet services routers for a variety of network functions including Ethernet transport, aggregation, and offering new Ethernet-based services.

For additional information on the MX-series Ethernet services routers, click the URL onscreen.
The EX-series Ethernet switches range from 88 Gbps up to over 3.2 Tbps of throughput, are designed for access, aggregation, and core deployments, and are well suited for low to high-density enterprise and data center environments.

For additional information on the EX-series Ethernet switches, click the URL onscreen.

The SRX-series services gateways provide up to 120 Gbps of throughput, and are designed to meet the network and security requirements for consolidated data centers, managed services deployments, and aggregation of security services in both enterprise and service provider environments.

For additional information on the SRX-series services gateways, click the URL onscreen.

2.2 The Junos Architecture

What separates Junos from other network operating systems? Well, there are important differences. We'll take a quick look at all of them, beginning with the architecture.

First, the Junos kernel is based on the FreeBSD UNIX operating system, which is an open source software system.

Plus, Junos functionality is compartmentalized into multiple software processes. Each process runs in its own protected memory space, ensuring that one process cannot directly interfere with another. When a single process fails, the entire system does not necessarily fail. This modularity also ensures that new features can be added with less likelihood of breaking current functionality.

2.3 Single Software Train

The process for developing Junos is also important to its success. All Junos platforms use the same source code base. This design ensures that core features work in a consistent manner across all Junos-based platforms. Because many features and services are configured and managed the same way, the setup tasks and ongoing maintenance and operation within your network are simplified.

2.4 Separation of Control and Forwarding Planes

Unlike other network operating systems, Junos has separate control and forwarding planes. All routing protocols run on a control plane. The control plane maintains routing tables, which it uses to build forwarding tables. The forwarding plane receives the forwarding tables from the control plane and uses those to forward traffic correctly. The advantage of this architecture is that it's not possible for one plane to interfere with the other.

And the separation of the control and forwarding planes is one of the key reasons why Junos can support many different platforms from a common code base.

Here you can see the Junos architecture. The control plane, shown above the dashed line, runs on the Routing Engine (or RE). The RE is the brains of the platform and is responsible for performing protocol updates and system management. The RE runs various protocol and management software processes that reside inside a protected memory environment. The RE maintains the routing tables, bridging table, and primary forwarding table.
The forwarding plane, shown below the dashed line, runs on the Packet Forwarding Engine (or PFE), which is usually a separate device. The PFE is responsible for forwarding transit traffic through the device. The PFE, in many Junos platforms, uses application specific integrated circuits (or ASICs) for increased performance.

Because this architecture separates control operations—such as protocol updates and system management—from forwarding operations, Junos platforms can deliver superior performance and highly reliable deterministic operation.

The PFE receives the forwarding table from the RE by means of an internal link. Forwarding Table updates are a high priority for Junos and are performed incrementally.

Because the RE provides the intelligent side of the equation, the PFE can simply do what it is told to do—that is, it forwards traffic with a high degree of stability and performance.

Click the link onscreen to learn more about the separation of control and forwarding planes.

If you’d like to move to the next topic in this section, click Next.

**The Routing Engine and Packet Forwarding Engine**

Let’s take a closer look at the Routing and Packet Forwarding Engines.

The RE has three primary responsibilities, which are to maintain routing intelligence, to control and monitor the chassis, and to manage the PFE.

The RE handles all protocol processes as well as other software processes that control the device’s interfaces, the chassis components, system management, and user access to the device. These software processes run on top of the Junos kernel, which interacts with the PFE. The software directs all protocol traffic from the network to the RE for the required processing.

The RE provides access to the user interfaces, such as the command line interface (or CLI). These user interfaces run on top of the Junos kernel and provide user access and control of the device.

The RE controls the PFE by providing accurate, up-to-date forwarding tables and by downloading microcode and managing software processes that reside in the PFE’s microcode. The RE receives hardware and environmental status messages from the PFE and acts upon them as appropriate.

The PFE has two primary responsibilities, which are to forward traffic and to implement services.

The PFE is the central processing component of the forwarding plane. The PFE systematically forwards traffic based on its local copy of the forwarding table. The PFE’s forwarding table is a synchronized copy of the information created on and provided by the RE. Storing and using a local copy of the forwarding table allows the PFE to forward traffic more efficiently and eliminates the need to consult the RE each time a packet needs to be processed. Using this local copy of the forwarding table also allows Junos platforms to continue forwarding traffic during periods of control plane instability.

In addition to forwarding traffic, the PFE also implements a number of advanced services. Some examples of these advanced services include stateless firewall filters and class of service (or CoS).
2.5 Traffic Processing

Let’s wrap up this section on the differences between Junos and other network operating systems with a discussion of traffic processing.

Traffic that enters a Junos device is classified as either transit traffic or exception traffic.

Transit traffic enters an ingress network port, and is compared against the forwarding table entries and then forwarded out an egress network port toward its destination based on the matching forwarding table entry.

It’s worth noting that transit traffic only passes through the forwarding plane and is never sent to or processed by the control plane on a Junos platform. By processing transit traffic through the forwarding plane, Junos platforms are able to achieve predictably high performance rates.

Unlike transit traffic, exception traffic does not pass through the device but rather requires some form of special handling by the device. Examples of exception traffic include packets addressed to the local device, such as routing protocol updates, Telnet sessions, pings, traceroutes, and replies to traffic sourced from the RE.
Section 3: Understanding the Junos User Interface

3.1 Accessing the Junos Device

So you’ve got your Junos device installed and powered up. Now what? How do you access it?

The answer to that depends on which interface you plan to use to configure the device. Junos provides two methods of access: through the Junos CLI, which is a text-based command shell, and through the J-Web GUI. Note that while we do not cover the J-Web GUI in detail in this course, you can find out more about it at the link onscreen.

J-Web is accessible via either HTTP or HTTPS. J-Web simplifies the most common configuration tasks with quick configuration wizards. For more complicated configurations, the J-Web GUI allows you to directly edit the system’s text configuration file. The J-Web GUI is installed and enabled by default on most Junos platforms.

Note that the J-Web is not included in the install image for the M, T, or MX-series platforms. In order to use J-Web on these platforms, you'll need to install the J-Web image separately. You'll also need to enable the web management service once the image is installed.

To configure the device with the Junos CLI, you first need to physically connect a management device to the Junos device.

The recommended method for connecting to a new Junos device is through the out-of-band (or OoB) serial console connection.

The console port settings, shown here, are pre-defined and are not user configurable.

To learn how to connect to a Junos device through the console port, click the link onscreen.

To connect to a Junos device through the console port, locate the RJ-45/DB-9 console cable provided with your Junos device.

Plug the male RJ-45 end into the CONSOLE port on the Junos device.

Plug the female DB-9 end into the management device’s serial communication port.

Next, launch a serial terminal-emulation program on your workstation. The required settings to launch a console session are:

- Bits per second rate: 9600
- Data bits: 8
- Parity: None
- Stop bits: 1, and
- Flow Control: None
If nothing appears in the terminal window, press Enter to display the boot messages. You should see the initial login prompt.

Once you’ve completed initial configuration, another option for connecting to Junos devices is by using access protocols such as Telnet or SSH. Unlike the console connection, this access option requires configuration for a network port and the access protocol.

Many Junos platforms also offer a dedicated management Ethernet port. This management port provides OoB access, so transit traffic cannot be forwarded through this management port. Note that the name of the dedicated management Ethernet port varies between platforms.

3.2 CLI Basics

Once you’ve established the physical connection with the device, the first thing you’re going to want to do, of course, is log in. To do that, you’ll need to enter your username and password. Your network administrator will create a user account for you and assign permissions.

By default, all platforms come with the root user configured without a password. When you boot a Junos device running the factory-default configuration, you will encounter the word “Amnesiac” at the login prompt. This simply means that the device doesn’t yet have a configured hostname.

When you log in as the root user, you are placed at the UNIX shell. You must start the CLI by typing the `cli` command.

When you exit the CLI, you return to the UNIX shell. For security reasons, make sure you log out of the shell by using the `exit` command.

The prompt you see will vary depending on the device’s assigned name and the name of the user currently logged in to the device. In our example here, the user’s name is “root” and the device name is “host.”

3.3 Operational Mode and Configuration Mode

Before we learn more about the CLI commands, we need to understand how the two CLI modes function.

Junos offers two modes to choose from. You use “operational mode” to monitor and troubleshoot the device. The `monitor`, `ping`, `show`, `test`, and `traceroute` commands let you display information and test network connectivity for a Junos device. We’ll cover operational mode commands in more detail later in the course.

You use “configuration mode” to configure all properties of Junos, including interfaces, protocols, and user access, as well as several system hardware properties.

You can tell which CLI mode you’re in by looking at the prompt. The greater than sign tells you you’re in operational mode. The pound sign indicates you’re in configuration mode.
If you're just starting out with Junos, you probably can't remember all the commands. The CLI provides context-sensitive help at any point in a command line. Help tells you which options are acceptable at the current point in the command and provides a brief description of each command or command option.

To receive help at any time while in the Junos CLI, type a question mark (?). You don't need to press Enter. The CLI lists the available commands and options.

If you type the question mark after entering the complete name of a command or an option, the CLI lists the available commands and options and then redisplays the command name and options that you typed. So, here you can see that typing `clear ?` returns the possible commands.

If you type the question mark in the middle of a command name, the CLI lists possible command completions that match the letters you have entered so far, then redisplays the letters that you typed. Here, we typed `cle ?`, which generated the possible commands on screen.

In addition to the question mark command for context-sensitive help, Junos also offers help through the `help` command.

There are several ways to use the `help` command. Entering `help topic interfaces address` displays usage guidelines for the topic in the statement.

In this example, we get information on configuring an interface address.

You can use the question mark to determine the topics for which help is available.

In the example on the screen we entered the `help topic interfaces` command followed by the question mark to see which options are available.

A different help command, the `help reference` command, displays summary information for the referenced configuration statement.

Here, once again, we are seeking help for information about interface addressing.

The Junos CLI provides another cool little helpful feature called the completion function. You can use the spacebar to complete a command so it’s not necessary to type the full command or the command option name for the CLI to recognize it.

To use the complete feature, simply press the spacebar after a command or option that you’ve partially typed. If the partially typed letters begin a string that uniquely identifies a command, the CLI displays the complete command name. Otherwise, the CLI beeps to indicate that you’ve entered an ambiguous command, and it displays the possible completions. Here, we entered `sh spacebar i spacebar`. Since `i` is the first character in six different commands, the CLI lists those commands.
A related command is the tab complete command. You use this command to auto-complete assigned variables, including policy names, AS paths, community names, and IP addresses. Here, you can see, we entered t tab. The CLI auto-completes the rest of the variable. Just like spacebar completion, tab completion offers a list of possible completions, should multiple, ambiguous options exist.
Section 4: Using the Junos CLI

4.1 Editing Command Lines

At this point, you should know how to access the Junos device, how to log in, how to get help, and you should know a few completion shortcuts for saving some time.

Now, let’s take a look at how you configure your device. We’ll start with editing the command line. The Junos CLI supports EMACS-style keyboard sequences that allow you to move around on a command line and delete specific characters or words.

Here, you can see the following sequences that are supported. To see a command, select a command from the list onscreen.

You can also use the Arrow keys to move around on the command line. Note that to use the Arrow keys, the user session must be set to the VT100 terminal type, which is the default terminal type for Junos.

4.2 The Pipe Command

Some outputs can be quite lengthy. Because you may not want to see ALL the output for a specific command, you can filter the output using the pipe command.

To filter the output of either an operational-mode or a configuration-mode command, just add a pipe and an option to the end of the command. Let’s take a look at a few pipe options.

The pipe command can be used to compare the files with the rollback number for viewing configuration changes, if any. Here, we’re comparing the current configuration with the previous configuration.

Note that, it can be used in configuration mode using only the show command.

You can count the number of lines in the output using the count command.

You can also use the pipe command to show the set commands used to create the current configuration. This command is available in both configuration and operational modes.

To filter the output by ignoring text matching a regular expression, the ‘except regular-expression’ command is used with the pipe command.

Note that if the regular expression contains spaces, operators, or wildcard characters, you must enclose it in quotation marks.

To display output starting at the first occurrence of text matching a regular expression, the ‘find regular-expression’ command is used with the pipe command. Here, we use the pipe command with the show interfaces command to show all of the output starting with the term “traffic.”

And again, note that if the regular expression contains spaces, operators, or wildcard characters, you must enclose it in quotation marks.
To display the last screen of information, use the last command along with the pipe command. Here we use the pipe command with the show log ospf-trace command to show the last 5 log entries.

To filter the output for text matching a regular expression the 'match regular-expression' command can be used along with the pipe command. Here, we show output for all interfaces matching the term “down.”

Note that if the regular expression contains spaces, operators, or wildcard characters, you must enclose it in quotation marks.

Using the pipe command with the no-more command displays all output at once instead of one screen at a time.

You can also combine a pipe command with a help command, as we've done here. In this case, adding help to the show route pipe command displays all the possible filters available.

Finally, you can combine multiple pipe commands on a single line. Here, we've combined two instances of pipe match to search the messages log file for entries that contain the ge-0/0/0 interface as well as the snmp reference.

### 4.3 Operational Mode

Earlier, we mentioned there are two modes you use to manage a Junos device: Operational mode and Configuration mode. Let's take a closer look at the two modes.

First is operational-mode. Operational-mode CLI commands monitor and control the operation of the device.

The operational-mode commands are hierarchically structured, as you can see here.

For example, the show command displays various types of information about the system and its environment. One of the possible options for the show command is `ospf`, which displays information about the Open Shortest Path First (or OSPF) protocol.

Specifying the interface option, as in the show ospf interface command, outputs information on OSPF interfaces.

Some key operational-mode capabilities include entering configuration mode, controlling the CLI environment, and exiting the CLI. You also perform monitoring and troubleshooting from operational mode, using the `clear`, `monitor`, `mtrace`, `ping`, `show`, `test`, and `traceroute` commands.

Other operational mode functions include connecting to other network systems, copying files, restarting software processes, and performing system-level operations.

Finally, another handy command is the run command, which you use to execute an operational-mode command from configuration-mode.

### 4.4 Configuration Mode

The second Junos mode is configuration mode. Before we can discuss configuration mode, though, we need to take a look at how configuration files are created in Junos.
Junos only has a single operational configuration at any point in time. This configuration is referred to as the Active configuration, and its the same configuration the system loads during the boot sequence. This concept is analogous to both the running configuration and startup configuration in other vendor's software.

Unlike some vendor's software, configuration changes made in Junos do not take effect immediately. The batch configuration model allows you to group together and apply multiple configuration changes to the running configuration as a single unit, so you can review all the changes to a configuration before actually committing them.

When you enter configuration mode a working copy of the active configuration is generated. When you make changes to the configuration file, you're actually modifying the candidate configuration. Once you have made your changes to the candidate configuration, you commit them, and they become part of the active configuration. This allows you to make multiple changes to the candidate configuration and review them prior to committing them.

4.5 Entering Configuration Mode

Now that you know the different configuration terminology, we can begin to talk about configuration mode. You enter configuration mode by issuing the `configure` command from the CLI's operational mode.

You can always tell you're in configuration mode, because the prompt changes from the angle bracket (>) of operational mode to the pound sign (#), proceeded by the name of the user and the name of the device, as you can see here.

The portion of the prompt in brackets, such as [edit] here, is a banner indicating that you are in configuration mode and specifying your location within the configuration hierarchy.

4.6 Exclusive and Private Configuration

There are a few rules about configuration mode that exist to manage versioning and prevent one user from over-writing another user's configuration.

For instance, if you enter configuration mode, while another user is also in configuration mode, a message appears indicating who the user is and what portion of the configuration the user is viewing or editing.

To allow only a single user to edit the configuration, use the `configure exclusive` command. This will ensure that no other users can make changes at the same time.

Exiting exclusive configuration without committing changes results in the loss of any modifications made to the candidate configuration.

Alternatively, if several users want the ability to make simultaneous changes to the configuration without sharing a candidate configuration, you can choose to give all users their own copy of the candidate configuration using the `configure private` command.

When you commit changes in a private copy of the configuration, only the changes you made are integrated into the active configuration. Those elements of the configuration that you did not change remain untouched. This means that several users can use configure private to make nonconflicting changes to the active configuration at the same time. If multiple users try to commit conflicting changes, the CLI
displays an error message to the second (and subsequent) users who attempt to make conflicting changes. Users can choose to make the changes anyway by committing their changes a second time.

### 4.7 Statement Hierarchy

Configurations are organized hierarchically. In configuration mode, you enter commands that affect the statement hierarchy. The statement hierarchy stores configuration information and is independent of the CLI operational-mode command hierarchy. The commands available in configuration mode are also independent of the commands available in operational mode.

For example, in the CLI’s operational mode a `show interfaces ge-0/0/0` command displays specific operational information. While in CLI’s configuration mode a `show interfaces ge-0/0/0` command displays the statement’s hierarchy. The two commands are independent of each other.

The statement hierarchy is organized in a tree structure similar to Windows folders or UNIX directories, grouping related information into a particular branch of the tree.

Here, you can see that we’ve entered the `edit protocols ospf area 51 stub` command.

The statement hierarchy shows the command broken down into its constituent commands from the most general at the top to the most specific at the bottom.

### 4.8 Hierarchical Configuration

Now that you know how to enter configuration mode and how configuration hierarchy statements work, it’s time to actually start configuring the device.

To modify the candidate configuration, you use `set` commands. (Remember, you have to be in CLI configuration mode.)

To display the candidate configuration, you use the `show` command. Both commands are relative to the current configuration hierarchy, shown by the `[edit]` prompt. Here, we’re in the `edit system` hierarchy.

Configuration files use curly brackets (```{}```) and indentation to visually display the hierarchical structure of the configuration. So, here, since two distinct hierarchy levels are displayed, you see two sets of curly brackets.

The first set is for web-management hierarchy and the second set is for http. Notice that the curly brackets line up directly beneath the statement they relate to.

Terminating—or leaf—statements in the configuration hierarchy are displayed with a trailing semicolon (`;`). But note that you enter neither the curly brackets nor the semicolons as part of the set command.

### 4.9 Moving Between Levels – Moving Down

Because configuration statements are hierarchical, you’ll want to be able to move between levels in order to edit them.

Moving between levels is like changing directories.
To move down through an existing configuration statement hierarchy or to create a hierarchy and move down to that level, use the **edit** command, specifying your desired hierarchy level. After you issue an **edit** command, the configuration mode banner changes to indicate your current level in the hierarchy.

Here, you can see that we move from the edit level, as indicated inside the brackets, down four levels to the **edit protocols ospf area 51 stub** level.

4.10 Moving Between Levels – Moving Up

To move up one level from the current position in the hierarchy, use the **up** command.

To move up more than one level from the current position in the hierarchy, supply an optional count to the **up** command. You will be moved up the number of levels specified or to the top of the hierarchy if there are fewer levels than specified.

If you want to move quickly back up to the top of the configuration hierarchy, you can use the **top** command.

4.11 Moving Between Levels – Moving Around

There are other handy commands for moving around within or between hierarchies. You can combine different commands to move to a different hierarchy. Here we combine the **top** and **edit** commands to quickly move to a **system login** hierarchy and then with the **show** command to display the configuration details for that hierarchy.

The **exit** command returns the user to the most recent higher level of the hierarchy. Here, entering the exit command at the level **edit protocols ospf area 0.0.0.51 stub** takes us back up to the level **edit protocols ospf**.

Entering **exit** at the top level of the hierarchy, however, exits configuration mode.

Likewise, entering **exit configuration-mode** from any level of the hierarchy also allows you to exit configuration mode.

4.12 Adding Configuration Statements

Now that you know how to edit configuration statements and move around within and between levels, let's take a closer look at how you actually change a configuration.

As we mentioned before, you use **set** commands in the CLI configuration mode to modify the candidate configuration.

Here, you can see we entered **set ftp** in the **edit system services** level to add the **ftp** service. When we enter the **show** command, the **ftp** service appears.

4.13 Removing Configuration Statements

Similarly, you use the configuration-mode **delete** command to remove statements that were added to the configuration with a **set** command. This command deletes the statement and all its subordinate statements and identifiers. Here, you can see we deleted the telnet service.
Deleting a statement or an identifier effectively *unconfigures* the functionality associated with that statement or identifier, returning that functionality to its default condition.

You might consider using the **wildcard delete** function when deleting individual statements is too arduous but you don’t want to delete an entire configuration subhierarchy. Here’s sample syntax for a **wildcard delete** where we delete all interfaces that begin with **gigabit ethernet one**.

### 4.14 Other Handy Junos Commands

We’ve covered a lot of information so far. But before we move on to how to commit a configuration, there are a few more useful commands you should know.

Say you want to move a connection to another port for testing purposes. With the **rename** command in Junos, this is simple. Here, we’ve renamed interface ge-0/0/10 to ge-0/0/11.

Similarly, you can use the **replace** command to change a pattern of configurations statements.

What if you want to replicate an existing configuration and only change a few components? You can use the **copy** command to replicate a statement and then make the needed changes. For example, here we replicate the ge-0/0/10 configuration on interface ge-0/0/11.

Other handy commands include the **deactivate** command, which here we use to ignore the interfaces in the configuration statement, and the **insert** command, which we’ve used to insert a statement in another location, in the case inserting term three before term two for a policy named test.

### 4.15 Viewing the Candidate Configuration

Once you’ve created or edited a configuration statement, you’ll want to commit it, of course. (Remember, in Junos configurations don’t take effect automatically; you have to commit them first.)

Before you commit the configuration, though, you’ll want to view it. To display the candidate configuration, use the configuration-mode **show** command. This command displays the configuration at the current hierarchy level or at the specified level below the current location. Here we use the **show system services** command to display just the services. Alternatively, we can use the **edit** command to move to the system services sub hierarchy, and then enter the **show** command.

The **show** command has the following syntax: **show statement-path**. When displaying the configuration, the CLI indents each subordinate hierarchy level, inserts curly brackets to indicate the beginning and end of each hierarchy level, and places a semicolon at the end of statements that are at the lowest level of the hierarchy. The display format is the same format you use when creating an ASCII configuration file, and it is also the same format that the CLI uses when saving a configuration to an ASCII file. In cases, where an empty statement leads to an invalid configuration because it is incomplete or meaningless, the **show** command does not display any of the statement paths.

### 4.16 Committing a Configuration- Part 1

Next, to activate your candidate configuration, you use the **commit** command.

The commit operation can typically be performed from any hierarchy level.
The exception is when users enter configuration mode using the `configure private` option, which requires the `commit` command to be issued at the top hierarchy level.

When you commit a candidate configuration, the entire configuration is activated in its current form. So you might consider first entering the `commit check` command to validate the syntax of a candidate configuration without actually placing it into effect.

Here, you can see that after entering the `commit check` command, our configuration failed because of an incompatible setting at the referenced hierarchy level. The commit failure notifications typically provide a very clear explanation of why the commit operation has failed, as illustrated in this example.

Of course, `commit check` cannot catch logical errors in your configuration. One unfortunate scenario occurs when you configure a device remotely and make a mistake that leaves that device inaccessible to remote connection.

You can avoid this by using the `commit confirmed` command.

When you issue a `commit confirmed time-out` command, the system starts a timer, during which time it expects to see another `commit`. If you don’t issue a second `commit` within the time-out value specified (a range of 1 to 65,535 minutes is supported, with 10 minutes being the default), the system performs a rollback 1, `commit` sequence on your behalf.

Here you see the time out value is set to one minute. Since a second commit was not issued within the timeout specified, an automated rollback operation is performed.

After the automatic rollback you can load the rollback 1 file to look for your mistake.

Note that when you attempt to commit a configuration, Junos checks the candidate configuration for proper syntax. If the syntax is not correct, an error message indicates the location of the error, and no part of the configuration is activated. You must correct the errors before the commit operation will succeed.

4.17 Committing a Configuration, Part 2

There are a number of variations of the commit command that can make your life easier.

For example, you can also schedule a commit that occurs at a specific time using the `commit at time` command.

To view and clear pending commits, use the `show system commit` and `clear system commit` commands.

You can also add a log entry to your commit using the `commit comment “comment-string”` option. As you can see here, these logs are visible in the output of the `show system commit` command.

Finally, you can add the `and-quit` option to the `commit` command to activate your changes and exit configuration mode in a single step.

4.18 Backing Out of Configuration Changes

Say for some reason there’s a problem with the current configuration and you need to restore a previous configuration.
Junos saves the last 50 committed versions of the configuration, so you can restore any of them. To overwrite the candidate configuration with one of these previously committed versions, you use the CLI configuration `rollback` command. By default, the system returns to the most recently committed configuration (the active configuration).

To return to a version prior to the configuration most recently committed, you would include the version number in the `rollback` command.

Note that the `version` argument can be any one between 0 and 49. The most recently saved configuration is version 0, so restoring version 0 will reset the candidate configuration to the current active configuration. The oldest committed configuration that is automatically saved is version 49.

Of course, the `rollback` command modifies only the candidate configuration. To activate the changes loaded through the rollback operation, you must issue the `commit` command.

### 4.19 Comparing Configuration File Differences

Now, suppose you want to compare the differences between configurations. To do this, you have a few options to choose from.

First, you can add the `show` and `pipe` commands to the CLI `compare` function to display the differences between the candidate configuration and active configuration. This is also known as `rollback 0`. Configuration comparison is `patch`-like. Thus, instead of showing the entire configuration and where changes were made, only the actual changes are shown.

You can compare active and rollback configurations using the operational-mode `show configuration pipe compare rollback number` command, as shown here. This allows you to view differences between the active configuration and any of the 49 rollback configurations.

Similarly, the `show configuration pipe compare filename` command allows you to compare the active configuration to an arbitrary file. The `show pipe compare rollback number` and `show pipe compare filename` commands in configuration mode compare the candidate configuration with rollback configurations and arbitrary files respectively.

Finally, you can view differences between any two text files, including log files, by using the operational-mode `file compare files` command. The output of this command is in the same patch-like format as the `show configuration pipe compare` command.

### 4.20 Saving Configuration Files

There may be times when you want to save a candidate configuration with or without committing it.

You can save the candidate configuration from your current configuration session to an ASCII file using the `save` command. Saving a candidate configuration saves the configuration in its current form, including any uncommitted changes.

Note that configuration statements only at the current hierarchy level and below are saved. To save the entire candidate configuration, you must be at the top level of the configuration hierarchy.

If a path is not specified, Junos saves the configuration to the user’s working directory.
As an example, if user *nancy* saved a configuration file without specifying a path name, the configuration file would be saved in the /var/home/nancy directory by default.

Alternatively, you can specify a path and filename to save the configuration to, such as we've done here. Or, you can specify a URL as we've done here. This puts the file in the location explicitly described by this URL using the FTP protocol.

Substituting the word “prompt” for the password causes the FTP server to prompt you for the user's password.

Finally, you can specify a URL using the SCP protocol to put the file on a remote system using the SSH protocol. You will be prompted for the user's password.

### 4.21 Loading Configuration Files

If you've saved a configuration file, you'll want to be able to load it, of course.

To do that, you use the configuration-mode load command. This command loads a complete or partial configuration from a local file, from a file on a remote machine, or from a terminal emulation program's capture buffer. The load command supports several arguments that determine the specifics of the operation.

Let's cover in detail some of the arguments to the load command.

- **factory-default** replaces the full current configuration with the factory-default configuration.
- **merge** combines the current configuration with the configuration being loaded.
- **override** completely overwrites the current configuration with the configuration being loaded. You must perform override operations at the root of the configuration hierarchy.
- **patch** adds or deletes variables from the configuration based on the contents of a specified patch file. The patch file used in this operation uses the contextual diff format. The file generated from a `show | compare | save` operation creates such a file.
- **replace** looks for a replace tag in the configuration being loaded. Existing statements of the same name are replaced with those in the loaded configuration for stanzas marked with the replace tag.
- **set** allows users to load set commands from the terminal or from a saved file which consists of set configuration statements.
- **update** updates existing configuration with the configuration being loaded. When the update option is used, Junos attempts to notify only those processes affected by the configuration changes.

When the override option is used no such attempt is made by Junos. The update option can be used from any hierarchy while the override option can only be used from the top level hierarchy.

- **terminal** uses the text you type at the terminal as input to the configuration. Type Ctrl+d to end terminal input. This option is usually used in conjunction with a terminal emulation program's copy/paste functionality to copy and paste configuration data from one system to another.
Here the load function combines the text from the terminal to the configuration file using the `merge` option. Note the other options that can be used are `replace` and `override`.

**Relative** negates the normal requirement that the data being loaded in a `load merge` or `load replace` operation contain a full path to the related configuration hierarchy. The `relative` option negates this need by telling the device to assume that the data being loaded should be added `relative` to the current configuration hierarchy.

Here the load function combines the text from the terminal to the configuration file using the `merge` option in the `edit interfaces` hierarchy. Again, the other option available is `replace`.

Finally, of course, after the `load` operation is complete, you must issue a `commit` to activate the changes made to the configuration.

### 4.22 Using the Run Command

The final command we’re going to cover in this section is the `run` command.

The `run` command allows you to execute operational-mode commands while in configuration mode. It is similar to the `do` command on other vendors’ equipment, although much more flexible. This extremely handy time-saver works for all operational-mode commands and is supported at all configuration hierarchies. In this example, we are editing the configuration for the router’s ge-0/0/12 interface. After assigning what we hope to be the correct IP address, we commit the change and invoke the `run` command to execute a quick ping test.
Section 5: Initial Configuration

5.1 Powering on and off Junos Devices

By now, you've probably connected power and turned on the device, but in case you haven't, go ahead and do that now. Always refer to the safety guidelines shipped with your device to avoid accidents and injury.

If power is disrupted, Junos devices automatically power back up when the power is restored. In other words, no manual intervention is required when power is temporarily interrupted.

If you need to remove power from a Junos device, you should shut the system down gracefully. Since Junos software is a multitasking operating system, an abrupt removal of power could potentially corrupt the file system and prevent the system from booting properly.

To gracefully shut down Junos, use the request system halt CLI command.

As shown on the screen, the request system halt command provides a number of options which allow you to schedule the time of a system shut down, specify the media from which the next boot up operation should use, and log a user-defined message to the console and the messages log file.

5.2 The Factory-Default Configuration

When a new Junos device is shipped, it comes pre-configured with a factory-default configuration. These configurations are platform-specific and are designed to ease the initial implementation.

All factory-default configurations, regardless of the Junos platform, allow initial access through the root account. You may remember, we mentioned earlier there is no configured password for the root account.

In addition to the root account, factory-default configurations include system logging (or syslog) parameters, which tracks system events and writes those events to pre-defined log files.

We discuss system logging in greater detail in the secondary system configuration section.

All other configuration parameters found within a factory-default configuration are platform or model-specific.

There might be instances when you want to return a Junos device to its factory-default configuration.

Click the link on screen to learn how the restore a device to it’s factory default configuration.

You can overwrite the candidate configuration while in configuration mode using the load factory-default command. Junos does not allow you to save the configuration until you configure root authentication. So even though you can restore the factory default configuration, a password is required. Only the first time a device is accessed does it not require a password.

Also, don’t forget to issue the commit command to activate your changes.

Note that the prompt continues to display the previously defined hostname until the system is rebooted.
5.3 Initial Configuration Checklist

For security purposes, Junos requires that a root password be configured before any other modifications to the factory-default configuration are made. The root password, as well as the password for any other user, must be six characters or longer and must include a change of case, digits, or punctuation.

In addition to the root password, Juniper also recommends that you configure the device’s hostname, the system’s time, system services to allow remote access, and the management network parameters. We’ll walk you through these tasks typically associated with creating an initial configuration.

5.4 Logging in as Root

Before any configuration takes place you have to log into the system. The presence of Amnesiac, at the login prompt, indicates there’s no current hostname assigned to the device which is to be expected since you’re logging in to a device with a factory-default configuration.

At the log in prompt, type root and press enter. Notice that no password is required.

5.5 Starting the CLI

When you log in as the root user, you’re placed at the UNIX shell. You must start the CLI by typing the cli command and pressing enter.

When you exit the CLI, you return to the UNIX shell. For security reasons, make sure you also log out of the shell using the exit command.

5.6 Entering Configuration Mode

Notice how the prompt has changed. You’re now in the CLI operational mode. As we mentioned earlier, you can only make changes to the configuration file in configuration mode. You enter configuration mode by typing configure at the operational-mode prompt.

5.7 Defining the Root Password

Now that you’re in configuration mode, let’s make our first change to the candidate configuration file by defining a root password.

Navigate to the [edit system] hierarchy level by typing edit system.

Next, enter set system root-authentication plain-text-password and press enter.

Here, you’re presented with a prompt where you can enter the new password.

Once again, you’re presented with a prompt where you confirm the new password.

In the event that the passwords entered do not match, you’ll receive an error message and will need to try again.

Junos enforces the following password restrictions:
the password must be at least 6 characters, you can include most character classes in a password (alphabetic, numeric, and special characters), except control characters and a valid password must contain at least one change of case or character class.

5.8 Defining the System’s Host Name

To help identify individual devices within your network environment, you should assign unique host names to each system.

To configure the system’s host name, type set host-name, then the desired host name and press enter. Here, we’ve used the hostname host.

5.9 Setting the System’s Time

To maintain consistent time on devices, you should configure your system’s time parameters.

You can configure the current date and time information along with the proper time zone for the device. Or even better, you can use the Network Time Protocol (or NTP) on all of your network devices so the time is synchronized. This can come in handy if a network problem occurs and you need to compare system logs from multiple devices.

To find out how to set your device to the local time, click the first link onscreen. To find out how to configure an NTP server, click the second link onscreen.

To set the device to the local time, first set the time zone with set time-zone and the desired time zone. Here, we’ve set the time zone to Los Angeles but this is just an example. You can select the option for your specific location.

Then you set the time by entering the year, month, date, time format. So for instance, the command run set date 200911070900.00 sets the time to 9:00 AM on November 7, 2009. Note that the system time is defined in operational mode rather than configuration mode.

To configure an NTP server, enter the command set ntp server and the server address, then press enter. Here, we’ve used ntp server 10.210.14.130.

To ensure your system synchronizes its time with the NTP server when booting, you need to define an NTP boot server. This is typically the same device functioning as the primary NTP server.

To configure an NTP boot server, enter the command set ntp boot-server and the IP address of the NTP server and press enter. Here, we’ve used ntp boot server 10.210.14.130.

5.10 Enabling Access Protocols

You may want to access your Junos device in the future through Telnet or SSH. In order to do that you must enable those services.

To enable both Telnet and SSH, type set services telnet and press enter.

Then, type set services ssh and press enter.
5.11 Configuring the Management Ethernet Interface

Before remote access is possible, you must assign the system an IP address. You may remember that most Junos devices have a dedicated management Ethernet port, which is specifically designed for remote access through an access protocol such as Telnet or SSH.

Let’s assign a management IP address to the system’s management Ethernet interface now.

First, we need to navigate to the [edit interfaces] hierarchy level. To do this, you would enter the top and edit interfaces CLI commands separately or you can combine the two commands and simply enter the top edit interfaces command as shown here.

Then, you enter the command, set interface-name, unit number, family inet address. Here, we’ve used ge-0/0/0 as the interface name, unit 0, and family inet address 10.210.14.131/27 as an example. Then press enter to assign the management interface.

Notice in this case the management Ethernet interface is ge-0/0/0. The management Ethernet interface can vary between Junos devices.

Don’t worry if you’re still unclear on interface configuration. We cover interface configuration in greater detail in the next section.

5.12 Defining a Static Route for Management Traffic

When accessing a Junos device from a remote network through the management network, you’ll need to define a static route to the remote network with a next hop address assigned to a device on the management network.

In our example scenario, the management network is 10.210.14.128/27. Let’s assume that the remote network from which you’ll be connecting to this device is 10.210.0.0/16 and the device used as a next hop to reach that remote network is 10.210.14.129

Let’s add a static route that will accommodate management traffic from that remote network.

Enter top edit routing-options at the CLI command prompt and press enter to navigate to the [edit routing-options] hierarchy level.


Typically, static routes used for management traffic are only used by the local device and should not be readvertised to other devices connected through inband networks. We’ve included the no-readvertise option to exclude the route from being mistakenly readvertised in to a dynamic routing protocol, such as OSPF, through routing policy.

5.13 Activating the Initial Configuration

Now that we’ve made the initial modifications to the configuration, we’ll activate the candidate configuration and return to operational mode. To do that, you issue the commit command in conjunction with the exit command. Or you can simply enter the commit and-quit command to save time.

Notice that the configured host name now appears. This is evidence that your changes have taken effect.
5.14 **Viewing the Resulting Configuration**

Remember that the resulting configuration consists of all default parameters, included in the factory-default configuration, as well as all new parameters you've just added.

Let's take a look at the resulting configuration now.

Typing `show configuration` and pressing enter will display the current configuration.

Note that the configuration displayed here consists of the parameters included in the factor-default configuration as well as the parameters we just added.

5.15 **The Rescue Configuration**

Now that we've defined the initial configuration and ensured basic connectivity to the device through the management network is available, you should consider saving the configuration as the rescue configuration file. A rescue configuration is a user-defined, known-good configuration designed to restore connectivity in the event of configuration problems. We recommend that the rescue configuration contain the minimum elements necessary to restore network connectivity. For added security, the rescue configuration must include a root password.

You can save the active configuration as the rescue configuration using the CLI's operational-mode `request system configuration rescue save` command. If a rescue configuration already exists, the `request system configuration rescue save` command replaces the rescue configuration file with the contents from the active configuration. To manually delete the current rescue configuration, issue the `request system configuration rescue delete` command.

Once saved, you can load the rescue configuration by entering the `rollback rescue` configuration-mode command. Since the rollback operation only replaces the contents of the candidate configuration, you must issue `commit` to activate the configuration.

5.16 **Logging out**

Since you're logged in as the root user you'll not only need to log out of the CLI but you'll also need to log out of the shell. This is done using either the `exit` or `quit` command.

Entering `exit` returns back to the Unix shell prompt.

Entering `exit` once again logs out of the system completely.
Section 6: Interface Configuration

6.1 Interface Overview

Now that you’ve completed the initial configuration, you’ll probably want to put your device to work passing traffic. To do this, you’ll first need to configure the interfaces. But in order to do that, you need to understand how interfaces work on Junos devices.

Interfaces are primarily used to connect a device to a network so that traffic can be sent to and from the device.

You can install different types of physical interface cards (or PICs) in Junos devices, depending on the type of network the device is attached to. Common examples of PICs include Fast Ethernet, Gigabit Ethernet, DS3, ATM, and SONET. We discuss network interfaces in more detail later in this section.

Junos devices also have another type of PIC called service PICs. Service PICs are used for such things as monitoring traffic, performing Network Address Translation (or NAT), and tunneling traffic between two endpoints. Services interfaces can be provided through a physical services interface card or through software, depending on the Junos platform.

Finally, special interfaces within the device don’t correspond to any hardware card. These interfaces are used for internal communication within the device itself, for router management interfaces, and for the device’s loopback address.

Click the links onscreen to find out more about internal, management, and loopback interfaces.

As we mentioned in section 2, the control and forwarding planes in Junos devices are separate. But they still need to be connected, and internal interfaces are how they are connected. The actual designation for this interface is platform-specific, but examples include fxp1 and em0. To avoid any communication issues, you should not attempt to make any changes to internal interfaces.

You use the management Ethernet interface for out-of-band access to your Junos device. Unlike normal network interfaces, which receive and transmit traffic flowing between different network interfaces on the device (which is called transit traffic), the out-of-band management interface accepts traffic only to and from the device itself.

The actual designation for this interface is platform-specific but examples include fxp0 and me0.

Loopback interfaces are used to provide Junos devices a constant and dependable interface. Loopback interfaces are hardware independent, which means they are always reachable, while at least one physical path to that device exists.

The loopback interface uses the lo0 designation on all Junos platforms.

6.2 Interface Designations

If you simply install the interface cards into a device, the cards won’t work. You need to configure them, at a minimum assigning an IP address to the interface.
To configure an interface, you need to know the type of PIC it is and which slot in the device the card is installed, because you configure the interface based on its type and location in the device. Each interface name has two parts: A text string that identifies the interface type and digits that correspond to the interface's location in the device.

Some common interface types and corresponding designations include:

- fe for Fast Ethernet Interfaces
- ge for Gigabit Ethernet Interfaces
- so for SONET interfaces, and
- t3 for DS3 Interfaces

To see other common interface types and their designations, click the link on screen.

On screen you can see the most common interface types and their associated designations.

The location portion of the name varies among devices, depending on how many slots the device has for network cards. The digits identify the Flexible PIC Concentrator (or FPC), PIC, and port numbers. Note that FPCs house PICs, and PICs used for network connectivity contain network ports. PICs and Port numbers are usually counted from right to left.

To illustrate interface naming, suppose that you're configuring a Juniper router with eight FPC slots. The FPCs in the router are numbered from 0 through 7, with FPC 0 being in the first slot and FPC 7 being in the eighth one. The first FPC slot in all routers is slot 0.

Each FPC has fixed positions on it for PICs, generally for four PICs. The PIC positions on an FPC are numbered starting again at 0, from 0 through 3. (Be sure to check your specific device for PIC slot orientation.)

To illustrate how interfaces are named, suppose a Gigabit Ethernet PIC is installed in the third PIC position in an FPC installed in the router's second slot. You’d use the following name in the configuration for a port on that PIC:

The ge indicates a Gigabit Ethernet PIC; the 1 identifies the second FPC slot; and the 2 indicates the third position on the FPC. The last number, 0, corresponds to the port number on the PIC. Ports are numbered sequentially, beginning with 0. Therefore, a 0 here indicates the first port on the PIC.

Here's another example. fe-2/1/0 identifies a Fast Ethernet port that is located on the FPC in slot 2, on the PIC in slot 1 of the FPC, and is the first Fast Ethernet port on that PIC. Slot, PIC, and port numbering always start with 0. For another example, so-0/1/0 refers to a SONET port. That port is on FPC 0, PIC slot 1, and is port 0, the first port on the PIC.

### 6.3 Logical Units

In Junos, all physical interfaces have one or more logical interfaces called logical units or just units. At least one logical unit is required on every physical interface, which allows you to map one or more logical interfaces to a single physical interface. Some networks, such as in ATM and Frame Relay, may require multiple logical interfaces.
Layer 3 configuration always occurs at the logical interface—or unit—level. Because you always configure Layer 3 parameters at the unit level, the configuration of all interfaces is consistent. In addition, if you later decide to reconfigure an interface to use multiple logical interfaces, you simply add additional units. You don’t need to remove the physical interface configuration first. In these cases, you are free to choose whatever unit number you like.

Some interfaces (mostly point-to-point, such as interfaces with HDLC and PPP encapsulation) only support a single unit. In these cases, the unit must be Unit 0.

When you configure multiple units for a single physical interface, each unit is treated as a completely separate logical interface. So, it is possible, for example, to route traffic between two logical interfaces on the same physical interface.

A circuit identifier is different from a unit number. The circuit identifier identifies the logical tunnel or circuit, while the unit is used to identify a logical partition of the physical interface.

Although not required, it is generally considered best practice to keep the unit number and circuit identifier the same. This practice can greatly aid in troubleshooting when you have many logical circuits.

6.4 Configuration Hierarchy

All interfaces have the same configuration hierarchy organization. Junos considers all properties defined directly under the interface name to be the physical properties of that interface. Junos considers all properties defined directly under the unit number to be the logical properties of each particular subinterface. So all physical and logical interface properties are configured at their respective levels.

Configurable physical properties include data link layer protocol, link speed and duplex, and physical MTU.

Examples of configurable logical properties include protocol family, addresses, and virtual circuits.

The following list provides details for some physical interface properties:

- Data link layer protocol and keepalives.
- Link mode
- Speed
- Maximum transmission unit (or MTU)
- Clocking
- Scrambling
- Frame check sequence (or FCS), and
- Diagnostic characteristics

Note that the physical interface properties are associated with specific interface types.

The following list provides details for some logical interface properties:

- Protocol family
- Addresses
- Virtual circuits, and
- Other characteristics
We’ll cover an example in the next section.

6.5 Defining Interface Properties

In order for an interface to pass IP traffic on a Layer 3 network, it must be assigned an IP address. Here, you can see the command for setting an IP address for a Gigabit Ethernet interface named ge-1/2/0. Ge-1/2/0 is the physical interface, of course. Unit 0, as we just saw, is the logical interface. The family identifies the protocol that the interface will process, in this case IPv4. We cover protocols more in a moment. The last component of the command is the IP address and subnet mask associated with the interface.

It’s worth seeing how the address is displayed in the configuration file because it visually shows the different sections of the interface configuration by the layers of indentation:

The output shows four levels of indentation under the interfaces statement:

The first level, ge-1/2/0, is the container for the Gigabit Ethernet physical interface. Any statements configured directly under this level apply to the entire physical interface.

The second level has the statement "unit 0." A unit is a smaller portion of the physical interface that is called a logical interface. A physical interface must have at least one logical interface, and the first one is numbered 0 (not 1).

The third level is the family portion, which identifies a protocol, such as IP, that will be processed on this logical interface.

Some of the more common protocols you can configure on interfaces are:

*Inet*, which is the protocol for IPv4. Specify a 32-bit IPv4 prefix, followed by a slash and the prefix length (as we’ve shown earlier)

*inet6* is the protocol for IPv6. Specify a 64-bit IPv6 prefix, followed by a slash and the prefix length. *iso* is the protocol for interfaces that need to support CLNS, which is the ISO network layer service protocol that is used by IS-IS. You also need to configure one or more addresses on the router’s loopback interface, which IS-IS uses for its interface addresses.

*mpls* which is the protocol for interfaces that need to send and receive Multiprotocol Label Switching traffic. You don’t need to configure an address.

*Ethernet-switching* is the protocol for Ethernet interfaces that need to support Layer 2 switching.

While you can configure the *inet* and *inet6* protocol families on a single logical unit, you can’t configure a second protocol family in conjunction with the ethernet-switching protocol family.

Fortunately, Junos keeps track of these details for you and reminds you when you try to commit the configuration!

You almost always want to configure at least one family on each logical interface. Here the protocol is *inet*, which is the Junos way of saying IPv4. You must configure at least one protocol family on each logical interface to allow it to receive and transmit protocol traffic, and you often want to configure at least one address per protocol.
At the final level, most families have an address portion to associate an address with the interface.

Junos platforms can have more than one address on a single logical interface. Issuing a second `set` command does not overwrite the previous address but rather adds an additional address under the logical unit. Use of the CLI’s `rename` command is an excellent way to correct addressing mistakes. An example is shown here:

### 6.6 Tracking Interface State

Because each interface in the device has one or more addresses, you often need to find out which addresses are associated with the device, especially when you’re debugging a network problem or working on the device’s configuration. The `show interfaces terse` command shows the addresses along with the basic operational status of each of the device’s interfaces.

The `Local` column in the output lists the interface addresses. You can also see the `administrative` and `link` status, the `protocol` family details, and the address information for the specified `interface`. We cover interface monitoring in greater detail in a subsequent section.

### 6.7 Test your Knowledge-I

Here’s an opportunity to test what you’ve learned in this section. Let’s see how well you have understood the material in this section.

The M7i router has a single built-in FPC, and PIC slots are numbered right to left. See if you can identify the ge-0/2/0 port on the M7i router onscreen by clicking on the appropriate port.

Let’s try another one.

The M10i router has two built-in FPCs. FPCs are numbered top to bottom, and PIC slots are numbered right to left. Now, click on the fe-1/1/2 port on the M10i router onscreen. Or, click Get Answer to get the correct answer.

Finally, click on the ct3-0/2/3 port on the same M10i router onscreen. Or, click Get Answer to get the correct answer.
Section 7: Configuring User Accounts

7.1 Authentication Overview

Junos offers several options for setting up user accounts. You can set up individual accounts locally on Junos devices or on remote centralized authentication servers running RADIUS or TACACS+, which are authentication protocols.

For a small group of network devices, setting up individual accounts is probably simplest. You'll still need to manually configure the account information on each device, but as long as you don't have to modify the information very often, this is probably the way to go.

For larger networks, you'll probably want to use a centralized authentication server. That way, you can store all account information in one place, and you update it only once when changes occur. If an authentication server is used, the Junos device will require either a local user account or a template user account that corresponds with the user account defined on the authentication server. We look at a sample user account configuration in the next section.

7.2 Defining a User Account

When you create an account for an individual user, you assign a login name, password, and privileges, and provide information about the user.

Here, we've defined an account for John and given him super-user privileges, which allow him to perform all operations on the device.

The next command defines his full name.

And the third command creates a password for John. Even though the command says it's a plain-text command, Junos encrypts the password, as we see when we display the configuration:

Junos restrictions require that the passwords must be at least 6 characters. You can use alphabetic, numeric, and all special characters except control characters in a password, but it must contain at least one change of case or character class.

7.3 Using a Centralized Authentication Server

If you're setting up a larger network, you'll probably want to consider centralizing the authentication process, and set up a RADIUS or TACACS+ server on the network. RADIUS stands for Remote Authentication Dial-In User Service while TACACS+ stands for Terminal Access Controller Access-Control System Plus. We're going to focus on RADIUS here, but TACACS+ is similar.

A user's account information, including username, password, and privilege class, is stored on the server. When a user attempts to log in to the router, the router queries the RADIUS server to validate the user.

To set up centralized authentication:

First, enter configuration mode and configure the IP address and password (which RADIUS calls a “secret”) of the RADIUS server under the edit system hierarchy:
Again, when you display the RADIUS server configuration, the password is encrypted.

Then, add RADIUS as the primary authentication method within the authentication order.

With this configuration, when a user tries to log into the device, Junos first attempts to authenticate them through the RADIUS server. If the user’s credentials match those on the server, the user will be logged in.

If not, Junos checks for accounts configured locally on the device. If the user has a local account and the credentials match, they are logged in. Otherwise, access is denied.

As illustrated in our example, you can prioritize the order in which the software tries the different authentication methods through the authentication order statement.

For each login attempt, Junos tries the authentication methods in order, until the password is accepted. If no reply is received from any of the listed authentication methods, Junos always consults local authentication as a last resort (even if it’s not listed in the authentication order). We’ll look at some examples to help explain these concepts further in the next section.

7.4 Authentication Examples

Click the buttons onscreen to view examples of different authentication methods.

Example 1 - Authentication Order: radius>tacplus>password

In this example, a Junos device is configured with an authentication order of \texttt{[radius tacplus password]}.

When the user attempts to establish a connection to this Junos device, the user is prompted to enter the username and password. Once the user enters the username \texttt{lab} and the password \texttt{lab789}.

The Junos device attempts to authenticate the user through the defined authentication methods. The credentials are first passed to the RADIUS server. Since the RADIUS server cannot find matching credentials in its database, it rejects the authentication request.

Once the Junos device receives the reject message, it sends the user’s credentials on to the second authentication method defined in the authentication order statement, the TACACS-PLUS server in this case. In a similar manner the TACACS-PLUS server rejects the authentication request because the credentials do not match an entry in the server’s local database.

Finally, the Junos device attempts to authenticate the user through the local user database where a match is found and the user is authenticated.

Example 2 - Authentication Order: radius>tacplus

In this example, we’ve removed the password authentication method from the Junos device’s authentication order. As illustrated in the previous example, both authentication servers respond with reject messages because the user credentials received do not match the local entries stored on the authentication servers. In this case the local password database on the Junos device is not checked and the user is not authenticated.

Example 3 - Authentication Order: radius>tacplus>local database
This example uses the same authentication order as the last authentication example with the difference that the authentication servers are not responsive and the Junos device does not receive the reject messages. Since no reject messages are received by the Junos device, the local authentication database is consulted and the password is accepted. Junos always consults the local authentication database when a response is not received so that the user has some way of accessing the device.

7.5 Login Classes

Beyond user authentication, each command or configuration statement you enter must be authorized.

Junos applies authorization to all nonroot users, and you cannot disable this feature. Authorization applies to both the J-Web interface and the CLI.

A configured hierarchy of authorization components, as shown by the graphic here, defines whether or not a command is authorized.

At the highest level, how a user account is defined determines authorization parameters.

You assign user privileges to a user by assigning that user to a specific login class.

A user’s login class is what determines that user’s permissions, or what they are allowed to do on the device.

There are four predefined login classes that can be used for many common situations. The first login class is "super-user", which has all permissions. A super-user can perform any and all operations on a Junos device. You should only assign this login class to key individuals who monitor and maintain all aspects of your devices.

The next login class is "operator." An operator is allowed to work in operational mode to check the status of the device, clear statistics, and perform reset operations, including restarting processes and rebooting the system. Operators can look at the device’s configuration, but can’t modify it. This login class is ideal for a network operations team responsible for monitoring Junos devices.

A "read-only" user is exactly what it sounds like. This user has permissions to display and monitor a device’s status and its protocols. If further action is required, a "read-only" user must get an engineer or administrator with additional permissions.

The final class is "unauthorized." An unauthorized user has no privileges at all on the device. When a user in the unauthorized class logs in, Junos restricts the user from performing any action other than logging out of the system.

In addition to the predefined login classes we just mentioned, you can also configure custom classes, if needed. A custom login class allows you to associate specific permissions with the custom login class. Click the link onscreen to see the assignable user permissions.

Here is the list of permissions that can be assigned to a custom login class.

In addition to specific permission flags, you can also add or limit the user of specific operational and configuration mode commands through custom login classes. You use the deny-commands, allow-commands, deny-configuration, and allow-configuration statements to define regular expressions that
match operational commands or configuration statements. Matches are explicitly allowed or denied, regardless of whether you set the corresponding permission flags. You apply the deny-statements before the corresponding allow-statements, resulting in the authorization of commands that match both.

7.6 A Custom Login Class Example

Let’s take a look at a configuration example of a custom login class.

In this example, user nancy is a member of the noc-admin class. We can see the noc-admin class has the clear, network, reset, and view permissions.

In addition to the assigned permissions, we can also see that the noc-admin class can enter configuration mode using the configure private command and is allowed to alter configuration parameters at the [edit interfaces] and [edit firewall] hierarchy levels.

Notice, though, that the noc-admin class is denied the ability to manipulate files using the operational-mode’s file command and is specifically excluded from navigating to or viewing configuration details at the [edit groups] hierarchy level.
Section 8: Tracking System Events

8.1 Logging

Keeping records of what happens on your device tells you how the device operates under normal conditions and alerts you when things go wrong. The two common ways to track device operation are SNMP and logging.

Let’s take a look at logging first. You can save event history to files on the device or a remote server as well as watch events as they happen in real time.

Junos has two types of logging facilities. System logging (also called syslog) records system-wide events. Trace logging (also called tracing) records events related to specific operations (such as routing protocols).

Collecting syslog messages is straightforward: you configure the file in which to store them, the type of events you want to track, and the event severity.

Here, you can see we configured the device to store logs of all events that have a severity level of “warning” (or more severe) in a file named “messages”:

In this configuration, “any warning” indicates the event and severity level.

Onscreen, you can see the different types of logging events and severity levels you can use in collecting syslog messages, such as “authorization error” or “change log notice” to chart activity on the device that might be suspicious and damaging to the current device setup.

To look at the syslog messages on a Junos device, use the “show log” command followed by the name of the log file. In the example on the screen we use the messages file, which is the primary syslog file:

8.2 Trace options

In addition to the syslog feature, you can use the “traceoptions” feature to get more detailed information about a particular operation on the device. (This process provides information similar to that produced by the “debug” feature on some other companies’ devices.)

For example, you may want to keep an eye on routing protocol operation. When you do, you can turn tracing on for all routing protocols or for an individual routing protocol.

To get an idea of the general routing protocol operation on a Junos device, configure a file in which to store the operational events and a list of flags that define the types of events you want to record.

The configuration here collects information about all events (or flags) in the file trace-events.

Logging is an excellent way to monitor and keep track of your device, and the permutations are open enough to adapt to the size, complexity, and security of your network.

To save time when troubleshooting, it may be beneficial to pre configure tracing for protocols, interfaces, and global routing or switching operations and simply leave the tracing configuration inactive until it’s required.
8.3 Traceoptions Configuration Example

To trace the operations of a specific protocol, you include the `traceoptions` statement at the `[edit protocols protocol-name]` hierarchy.

Here, we’ve set a typical tracing configuration that provides details about the Open Shortest Path First (or OSPF) Protocol. You can see that the trace results are being written to the file "ospf-trace". You can also see that the “replace,” “size,” “files,” “no-stamp,” and “no-world-readable” options are also included. We’ll cover these options in a moment.

In many cases you’ll want to add the `detail` switch to a given protocol flag, like we’ve done here, for the added information often needed in troubleshooting scenarios.

In any event, you’ll want to be selective in what you trace because selecting the `all` keyword will likely numb your mind with trivial details.

Here are the configuration options for tracing files:

The `file filename` option specifies the name of the file in which to store information.

The `size` option specifies the maximum size of each trace file, in kilobytes, megabytes, or gigabytes.

When a trace file named `trace-file` reaches this size, it is renamed `trace-file.0`.

When the trace file again reaches its maximum size, `trace-file.0` is renamed `trace-file.1`, and `trace-file` is renamed `trace-file.0`.

This renaming scheme continues until the maximum number of trace files is reached. The software then overwrites the oldest trace file. If you specify a maximum file size, you also must specify a maximum number of trace files with the `files` option. The default size of the trace files depends on your specific Junos device.

The `files number` option specifies the maximum number of trace files.

The default is ten files.

The `no-stamp` option prevents timestamp information from being placed at the beginning of each line in the trace file. By default, if you omit this option, timestamp information is placed at the beginning of each line of the tracing output.

The `replace` option replaces an existing trace file if one exists. By default, if you omit this option, tracing output is appended to an existing trace file.

The `readable` option allows any user to view the file.

The `no-world-readable` option allows only the user who configured the file to view it. This is the default setting.

Traceoptions are also available at other configuration hierarchies. Including the `traceoptions` statement at the `[edit interfaces interface-name]` hierarchy level allows you to trace the operations of individual
interfaces. When you enable tracing for a specific interface, you cannot specify a filename. In this case the Junos kernel logs the tracing information for the specified interface in the messages file.

### 8.4 Analyzing Log and Trace Files

By default, Junos stores log and trace files in the/var/log directory.

To view stored log files, use the `show log` command.

Note that the CLI automatically pauses when there is more than one screen's worth of information, and that at the more prompt, you can enter a forward slash character to conduct a forward search.

Enter h when at a more prompt to view the context help screen of available commands, as shown here.

Being able to cascade multiple instances of the CLI's pipe functionality is a real benefit when searching a long file for specific information. Here you can see the required syntax to evoke logical AND and logical OR searches within extensive outputs and files.

### 8.5 Miscellaneous Log File Commands

Let's wrap up our discussion of using logging and tracing for event tracking by covering a few other logging and tracing commands.

You use the `monitor start` command to view real-time log information. You can monitor several log files at a time. The messages from each log are identified by `filename`, where `filename` is the name of the file from which entries are being displayed. Junos lists the filename before displaying a message.

To stop all real-time monitoring, use the `monitor stop` command.

You can monitor a file in real time while displaying only entries that match your search criteria. To use this functionality, use the `pipe match` command in conjunction with the search criterion like you see here.

Another command is the `Esc+q` command, which enables and disables syslog output to the screen.

If you don't delete or disable all trace flags, tracing continues in the background and the output continues to be written to the specified file. The file remains on the system's storage device until it is either deleted manually or overwritten according to the traceoptions file parameters we covered in the last section. To disable all tracing at a particular hierarchy, issue a `delete traceoptions` command at that hierarchy and commit the change.

To truncate files used for logging, use the `clear log filename` command.

To delete a file, use the `file delete` command. When a log file is deleted, logging to that file cannot continue since the file no longer exists even though the corresponding configuration for the file still exists. If a log file is deleted and you want to resume logging to that file, you can recreate that log file by deactivating and reactivating that portion of the configuration file.

### 8.6 Simple Network Management Protocol (SNMP) Operation
Another way to track the events that occur on network devices is to use Simple Network Management Protocol (or SNMP). SNMP is an Internet standard protocol for managing devices on an IP network. SNMP has centralized network management systems (or managers) that actively monitor network devices (or agents) by querying them and collecting status information and statistics from them. An SNMP manager is typically a dedicated computer while SNMP agents are devices such as routers and switches.

SNMP uses a set of terms all its own. Click here to find out what all the acronyms mean.

SMI stands for Structure of Management Information, which defines the way data is stored in a Management Information Base (or MIB).

MIB stands for Management Information Bases. They are hierarchical databases, like the directory structure on a PC or Mac, in which SNMP agents such as a router store their status information and statistics. The information in each MIB is arranged as a hierarchical tree structure, with branches that move down from the root node. Each branch eventually ends at a leaf. Branches are just like directories on a computer, and leaves are just like the files in directories. One difference between MIBs and computer directories is that each branch and leaf in the MIB is identified not only by a name, but also by a number.

SNMP defines standard MIBs, and individual network equipment vendors can also define proprietary MIBs. The standard MIB for use in TCP/IP networks is called MIB-II because it is the second version of this MIB. For your SNMP client to be able to retrieve information stored in these MIBs, it must know the structure of the MIB. You can download all Juniper router MIBs, both the standard and proprietary ones, from the juniper.net.

OID stands for Object Identifier, and is the number that uniquely identifies a branch or leaf in the MIB. The OID is actually a string of numbers, with one number for each branch in the hierarchy and one number for the final leaf in the hierarchy. The OID generally begins with a period to indicate the top of the tree (or the root node), and each subsequent number is separated by a period. An example of an OID from the standard MIB-II MIB is .1.3.6.1.2.1.1.1, which points to the device’s description.

If you want an SNMP manager to monitor your network device, you need to configure the device to be an SNMP agent:

To transform the device into an agent, you place the device into an SNMP community using the command `set snmp community public authorization read-only`. This command uses one of the common SNMP communities, `public`. The second part of the command defines how the agent (your device) will respond to requests from the manger (or NMS). An authorization of `read only` means that the device will send its information to the NMS, but the NMS won’t be able to modify any settings on the device (which it could do if you specified an authorization of `read write`). You can configure the device to respond to multiple communities, each with its own authorization level.

You can configure basic information about the device for the NMS to collect when it queries the device, such as the device’s location and description and who to contact about the device. This information corresponds to leaves in the `system` group in the standard MIB-II, and NMS systems on the network can collect it when querying the device.

In normal SNMP operation, the NMS periodically queries the device. If any unexpected events occur on the device, the NMS will find out only after sending a query.

The device can be configured to send notifications to the NMS when unexpected events occur.

This notification means that the NMS, and the people monitoring the NMS, can find out about device problems more quickly.
These notifications are called traps, and you can configure the types of events that trigger the device to send traps. Here you can see the various SNMP trap categories.

Onscreen, you can see we’ve configured the device to send traps when an NMS system uses the wrong community when trying to access the device.
Section 9: Operational Monitoring and Maintenance

9.1 Monitoring Tools

Now that we’ve covered some of the basic configuration tasks, let’s take a look at how you perform some basic monitoring and maintenance.

We’ll start with monitoring.

The primary monitoring tool for Junos is the Junos CLI itself. The CLI includes several show and monitor commands which facilitate system monitoring. We’ll highlight many of the CLI monitoring capabilities shortly.

In addition to the Junos CLI, a number of secondary monitoring tools exist such as the J-Web, SNMP, hardware LEDs, and front-panel displays or LCDs.

For specific details on a particular platform, check the technical publications at the link onscreen.

9.2 Monitoring System-Level Operation

You can use the CLI to obtain most system information by using the show system commands.

Let’s look at some of the most common commands.

The alarms argument displays current system alarms.

The boot-messages argument displays the messages seen during the last system boot.

The connections argument displays the status of local TCP and UDP connections.

The processes argument displays the system’s running processes.

The statistics argument provides options for viewing various protocol statistics.

The storage argument displays the status of the file system storage space.

9.3 Monitoring the Chassis

You can monitor the chassis and obtain chassis information using show chassis commands. The following arguments are some of the most common:

The alarms argument displays current chassis alarms.

The environment argument displays component and environmental status as well as the operational speeds of the cooling system.

The hardware argument displays an inventory of the installed hardware components along with each component’s serial number.

The routing-engine argument provides operational status and utilization details for the routing engine.
9.4 Verifying Interface Status

You can use the `show interfaces` command to verify various details and status information for interfaces. A number of command options exist that determine the generated output for the `show interfaces` command.

Here, you can see we used the `interface-name` option, which filters the generated output and displays details only for the specified interface.

If the `interface-name` option is excluded, the output displays interface details for all installed interfaces.

9.5 Terse Output

The example here illustrates the `show interfaces terse` command. In this example, we've omitted the `interface-name` option, so all installed interfaces and their accompanying details are displayed.

This command is ideal when you simply need to verify state information for physical and logical interfaces.

The output from this command displays all installed interfaces in the left column and provides state, protocol family, and addressing details to the right of each listed interface.

9.6 Extensive Output

An alternative to the `terse` command is the `show interface extensive` command, which you can use to view detailed information for a named interface (or all interfaces when a specific interface is not identified).

The example here shows a portion of the generated output when using the `extensive` option.

This command is ideal for investigating or troubleshooting interfaces because it shows extensive physical and logical interface properties. It's also a great command when determining default settings for interfaces.

9.7 Monitoring Interfaces

Here you can see typical output from the `monitor interface` command. Your terminal session must support VT100 emulation for the screen to display correctly.

This command provides real-time packet and byte counters as well as displaying error and alarm conditions.

To view real-time usage statistics for all interfaces, use the `monitor interface traffic` command. The sample output of this command shows how much each field has changed since you started the command or since you cleared the counters by using the `c` key.

9.8 Network Utilities: Ping and Traceroute

Now let’s take a look at several network utilities available to you.

Junos provides `ping` and `traceroute` utilities which you can use to determine general network reachability and the path that packets take to reach a destination.

The `ping` command is issued with a specific destination. The responses you see on-screen indicate a successful ping test and confirm reachability to the destination.
The **traceroute** command illustrates the path taken by packets to reach the specified network host.

You can use various arguments with the **ping** and **traceroute** commands, such as source IP address and packet size, to further assist in problem isolation.

By default, the ping utility sends a continuous flow of ICMP echo requests to the referenced destination. To stop the ping operation, you press the `Ctrl+c` keys, as we show here.

Alternatively, you can include the **count** option with a specified number of ICMP echo requests to send out as shown below:

**9.9 Network Utilities: Monitoring Traffic**

Junos provides a network utility, similar to tcpdump, through the **monitor traffic** command. This command allows you to monitor traffic originating from or terminating on the local Routing Engine (or RE).

This utility is often used as a troubleshooting tool and includes a number of arguments to tailor your troubleshooting activities. Some of the more common arguments are highlighted in the example onscreen:

The **interface** argument allows you to specify the interface through which the monitored traffic will enter or leave. If you do not specify an interface, the traffic on the management interface will be monitored.

The **layer2-headers** argument instructs the system to display Layer 2 header information. This option is ideal when monitoring or diagnosing Layer 2 problems on an interface.

The **matching** argument allows you to match on specific packet fields or protocols specified with the argument.

You can include the **detail** or **extensive** arguments to fully decode the protocol and provide additional information.

You can also include the **no-resolve** argument to avoid delays related to the reverse lookup process.

To stop traffic monitoring and return to the command prompt, enter the `Control-C` key sequence.

**9.10 Network Utilities: Telnet, SSH, and FTP Commands**

Junos supports Telnet, SSH and FTP clients. These clients allow you to originate and establish Telnet, SSH, and FTP sessions from your Junos device.

To transfer files to and from Junos devices, you use the **file copy** command. The example here shows the **file copy** command used in conjunction with the FTP client to transfer a file from a remote FTP server to the local Junos device.

To initiate Telnet or SSH sessions from your Junos device, you use the **telnet** or **ssh** commands.

**9.11 Junos Maintenance: Displaying Junos Version**

To determine the current Junos version running on your device, you use the **show version** command.
You can include the **detail** option to view additional details about the software packages and the processes included in the version.

Some common Junos packages include:

- **J kernel**, the kernel and network tools package, which contains the basic operating system files.
- **J route**, the Routing Engine package, which contains the Routing Engine software.
- **J p-f-e**, the Packet Forwarding Engine package, which contains the PFE software.
- **J docs**, the documentation package, which contains the documentation set for the software.
- **J crypto**, the encryption package, which contains the domestic version of the security software.

The actual packages may vary between the different Junos images. For a complete list of Junos packages for a specific image, use the **show version detail** command.

### 9.12 Junos Naming Convention

Junos follows a naming convention format of **package-release-edition**.

- **package** is the description of the software contents, and includes **jinstall**, which is used on M, T, and MX-series, **jinstall-ex**, which is used on EX-series, **junos-jsr**, which is used on J-series, and **junos-srx**, which is used on SRX-series. In our example here, the package is jinstall.

- **release** describes the Junos version and includes several subcomponents. The release includes two integers that represent the major and minor release numbers as well as a capital letter that indicates the type of software release. In most cases, the letter is an R to indicate that this is released software. If you are involved in testing prereleased software, this letter might be a B (for beta-level software) or I (for internal, test, or experimental versions of software). The release also includes a build and spin number for the Junos version.

- **edition** will typically be either **domestic** or **export**. Domestic versions support strong encryption, whereas export versions do not. A third, less common, edition called FIPS exists which provides advanced network security for customers who must comply with and operate in a Federal Information Processing Standards (or FIPS) 140-2 environment.

All Junos software is delivered in signed packages that contain digital signatures, Secure Hash Algorithm (SHA-1), and Message Digest 5 (MD5) checksums. A package is installed only if the checksum within it matches the hash recorded in its corresponding file. The actual checksum used depends on the software version.

### 9.13 Upgrading Junos Software

To upgrade Junos, you must first download the appropriate image for your Junos device from the Junos download site. You can download Junos using a web browser or through an FTP client (including the one on the Junos device itself). Regardless of the download method you choose, you must have a valid service contract and access account.
To download Junos through a web browser, point your browser to the http://www.juniper.net/support/ URL, and login using an access account. On the support page that opens, select your geographical location. Then select the desired image. In the dialog box that appears, accept the request to begin the download process.

To access Junos through an FTP client, initiate an FTP session from an FTP client to the FTP server using the command ftp ftp.juniper.net. Log in with your customer support-supplied username and password. Once validated, the FTP session opens. Navigate to the correct software directory. The server software download structure here is /volume/download/docroot/software/junos/junos version, build and spin no.. Set the file transfer mode to binary using the bin command. Next, the directory in which the file should be specified. Here we have saved the file to /var/tmp/ location. Finally, download the installation file using the get command along with the desired file name. In our example we download the junos-jsr-junos version, build and spin no. file. Once the download is complete, close the FTP session using the bye command.

Since individual Junos images are designed for specific Junos platforms, you must ensure the correct image is downloaded!

Once you've downloaded the image, enter the request system software add <path/image name> command, where <path/image name> is the local path and file name or the remote FTP or SCP URI that contains the required Junos image to download and install. To activate the new software, you must reboot the system. The system reboot can be performed as a separate step or can be initiated by adding the reboot option at the end of the request system software add command.

Once Junos is installed, you are notified that the system is rebooting to complete the installation. You can use a console connection to view details of the upgrade process. Watch for any error messages indicating a problem with the upgrade.

Junos devices run binaries supplied only by Juniper Networks. Each Junos image includes a digitally signed manifest of executables, which are registered with the system only if the signature can be validated. Junos does not execute any binary without a registered fingerprint. This feature is designed to protect the system against unauthorized software and activity that might compromise the integrity of your Junos device.

### 9.14 Software Upgrade Example

Let’s take a look at an example upgrade. When upgrading Junos, you need to reference the image name and a local path or a remote server within a Uniform Reference Identifier (or URI). Junos images copied to a Junos device in preparation for an upgrade should be stored in the /var/tmp directory.

These images can be deleted later using the request system storage cleanup command.

Although there is typically plenty of storage space, it is a good practice to check available storage capacity before downloading a new Junos image. You can view compact-flash device storage details with the show system storage command.

When an upgrade is performed, the system must be rebooted in order for the new version to take affect. To save time and keystrokes, you can use the reboot option when performing the upgrade. Once Junos is installed, you are notified that the system is rebooting to complete the installation. Use the console
connection to view details of the upgrade process. Watch for any error messages indicating a problem with the upgrade.

Once the system has rebooted, you can issue the `show version` command, to verify the Junos version.

### 9.15 Password Recovery

If you become locked out of a Junos device, and need to recover the root password, you need to be connected through the console connection. This is a security precaution, designed to prevent unauthorized access.

To recover the root password, follow these steps:

Please note that the on-screen CLI sequence is only an approximate simulation.

First, establish console access and reboot the system. Watch as the system boots, and press the Spacebar when prompted during the boot loader process. When the system presents a `loader>` prompt or an `OK` prompt, enter `boot -s` to boot into single-user mode as shown.

Note that the `OK` prompt is used for Junos platforms based on the Intel architecture while the `loader>` prompt is used on Junos platforms based on PowerPC architecture.

Next, the system performs a single-user boot-up process and prompts you to run the recovery script. Enter a shell pathname or press Enter for a default shell. Enter `recovery` at this point. When you enter the `recovery` command, the system initiates the recovery script which allows you to login without the currently configured password.

Then, after a series of messages, the CLI starts and you are presented with an operational mode command prompt. At this point, you can enter configuration mode and reset the root password. The required commands for these tasks are shown onscreen. Do not forget to commit your configuration.

To complete the recovery, exit configuration mode. You are then prompted to reboot the system. Choose yes to reboot the system. Once the reboot is complete, you can login with the new root password.