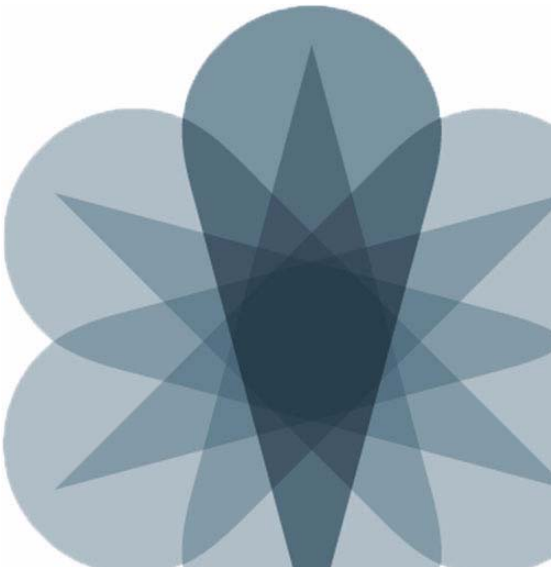
JNCIA-Junos Study Guide—Part 1

Worldwide Education Services

1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA
408-745-2000
www.juniper.net



JNCIA-Junos Study Guide—Part 1.

The information in this document is current as of the date listed above.

The information in this document has been carefully verified and is believed to be accurate for software Release 10.1R1.8. Juniper Networks assumes no responsibilities for any inaccuracies that may appear in this document. In no event will Juniper Networks be liable for direct, indirect, special, exemplary, incidental or consequential damages resulting from any defect or omission in this document, even if advised of the possibility of such damages.

Contents

Overview

Welcome to the *JNCIA-Junos Study Guide—Part 1*. The purpose of this guide is to help you prepare for your JN0-101 exam and achieve your JNCIA-Junos credential. The contents of this document are based on the *Introduction to Junos Software* course. This study guide provides students with the foundational knowledge required to work with the Junos operating system and to configure Junos devices. The study guide provides a brief overview of the Junos device families and discusses the key architectural components of the software. Additional key topics include user interface options with a heavy focus on the command-line interface (CLI), configuration tasks typically associated with the initial setup of devices, interface configuration basics with configuration examples, secondary system configuration, and the basics of operational monitoring and maintenance of Junos devices.

Agenda

Document Conventions

CLI and GUI Text

Frequently throughout this study guide, we refer to text that appears in a command-line interface (CLI) or a graphical user interface (GUI). To make the language of these documents easier to read, we distinguish GUI and CLI text from chapter text according to the following table.

Style	Description	Usage Example
Franklin Gothic	Normal text.	Most of what you read in the Student Guide.
`Courier New`	Console text: • Screen captures • Noncommand-related syntax	`commit complete` `Exiting configuration mode`
	GUI text elements: • Menu names • Text field entry	Select `File > Open`, and then click `Configuration.conf` in the `Filename` text box.

Input Text Versus Output Text

You will also frequently see cases where you must enter input text yourself. Often this will be shown in the context of where you must enter it. We use bold style to distinguish text that is input versus text that is simply displayed.

Style	Description	Usage Example
`Normal CLI` `Normal GUI`	No distinguishing variant.	`Physical interface:fxp0, Enabled` View configuration history by clicking `Configuration > History`.
`CLI Input` **`GUI Input`**	Text that you must enter.	`lab@San_Jose>` **`show route`** Select `File > Save`, and enter **`config.ini`** in the `Filename` field.

Defined and Undefined Syntax Variables

Finally, this study guide distinguishes between regular text and syntax variables, and it also distinguishes between syntax variables where the value is already assigned (defined variables) and syntax variables where you must assign the value (undefined variables). Note that these styles can be combined with the input style as well.

Style	Description	Usage Example
CLI Variable *GUI variable*	Text where variable value is already assigned.	`policy my-peers` Click on *my-peers* in the dialog.
CLI Undefined *GUI Undefined*	Text where the variable's value is the user's discretion and text where the variable's value might differ from the value the user must input.	Type **set policy *policy-name***. **ping 10.0.*x.y*** Select `File > Save`, and enter ***filename*** in the `Filename` field.

Additional Information

Education Services Offerings

You can obtain information on the latest Education Services offerings, course dates, and class locations from the World Wide Web by pointing your Web browser to: http://www.juniper.net/training/education/.

About This Publication

The *JNCIA-Junos Study Guide—Part 1* was developed and tested using software Release 10.1R1.8. Previous and later versions of software might behave differently so you should always consult the documentation and release notes for the version of code you are running before reporting errors.

This document is written and maintained by the Juniper Networks Education Services development team. Please send questions and suggestions for improvement to training@juniper.net.

Technical Publications

You can print technical manuals and release notes directly from the Internet in a variety of formats:

- Go to http://www.juniper.net/techpubs/.
- Locate the specific software or hardware release and title you need, and choose the format in which you want to view or print the document.

Documentation sets and CDs are available through your local Juniper Networks sales office or account representative.

Juniper Networks Support

For technical support, contact Juniper Networks at http://www.juniper.net/customers/support/, or at 1-888-314-JTAC (within the United States) or 408-745-2121 (from outside the United States).

JNCIA-Junos Study Guide—Part 1

Chapter 1: Junos Operating System Fundamentals

This Chapter Discusses:

- The Junos operating system and its basic design architecture;
- Traffic processing for transit and exception traffic; and
- Junos devices.

Robust, Modular, and Scalable

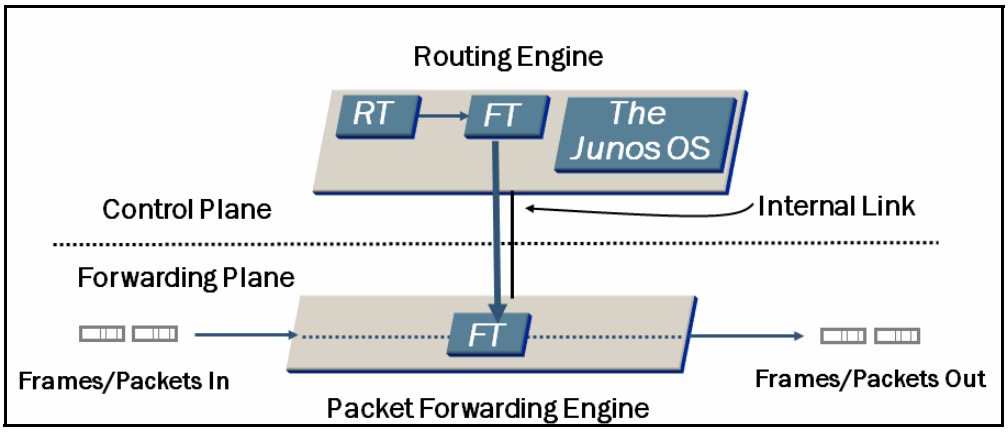
Junos OS functionality is compartmentalized into multiple software processes. Each process handles a portion of the device's functionality. Each process runs in its own protected memory space, ensuring that one process cannot directly interfere with another. When a single process fails, the entire system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

The Junos OS is the trusted, secure network operating system powering the high-performance network infrastructure offered by Juniper Networks. The Junos kernel is based on the FreeBSD UNIX operating system, which is an open-source software system.

Single Software Source Code Base

All platforms running the Junos OS use the same source code base within their platform-specific images. This design ensures that core features work in a consistent manner across all platforms running the Junos OS. 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.

Separate Control and Forwarding Planes



Another aspect of Junos modularity is the separation of the control plane and the forwarding or data plane. The processes that control routing and switching protocols are cleanly separated from the processes that forward frames, packets, or both through the device running the Junos OS. This design allows you to tune each process for maximum performance and reliability. The separation of the control and forwarding planes is one of the key reasons why the Junos OS can support many different platforms from a common code base.

The graphic illustrates a basic view of the Junos architecture and highlights the control and forwarding planes. The control plane, shown above the dashed line on the graphic, runs on the Routing Engine (RE). The RE is the brain of the platform; it 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 is based on an X86 or PowerPC architecture, depending on the specific platform running the Junos OS. The RE maintains the routing tables, bridging table, and primary forwarding table and connects to the Packet Forwarding Engine (PFE) through an internal link. Although all Junos devices share this common design goal, the actual components that make up the control and forwarding planes vary between the different Junos devices. For additional details about a specific Junos device, see the technical publications at http://www.juniper.net/techpubs/.

The PFE, shown below the dashed line on the graphic, usually runs on separate hardware and is responsible for forwarding transit traffic through the device. In many platforms running the Junos OS, the PFE uses application-specific integrated circuits (ASICs) for increased performance. Because this architecture separates control operations—such as protocol updates and system management—from forwarding operations, platforms running the Junos OS can deliver superior performance and highly reliable deterministic operation.

The PFE receives the forwarding table (FT) from the RE by means of an internal link. FT updates are a high priority for the Junos OS kernel and are performed incrementally.

Because the RE provides the *intelligence* side of the equation, the PFE can simply perform as it is instructed—that is, it forwards frames, packets, or both with a high degree of stability and deterministic performance. This architectural design also makes possible the incorporation of high availability features like graceful Routing Engine switchover (GRES), nonstop active routing (NSR), and unified in-service software upgrades (ISSUs).

Maintains Routing Engine Intelligence

The RE handles all protocol processes in addition to 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.

Controls and Monitors Chassis

The RE provides the CLI in addition to the J-Web GUI. These user interfaces run on top of the Junos kernel and provide user access and control of the device. We discuss user interfaces subsequently.

Manages Packet Forwarding Engine

The RE controls the PFE by providing accurate, up-to-date Layer 2 and Layer 3 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.

Forwards Traffic

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 platforms running the Junos OS to continue forwarding traffic during control plane instabilities.

Implements Services

In addition to forwarding traffic, the PFE also implements a number of advanced services. Some examples of advanced services implemented through the PFE include policers that provide rate limiting, stateless firewall filters, and class of service (CoS). Other services are available through special interface cards that you can add to the PFE complex.

Transit Traffic

Transit traffic consists of all traffic that enters an ingress network port, is compared against the forwarding table entries, and is finally forwarded out an egress network port toward its destination.

A forwarding table entry for a destination must exist for a device running the Junos OS to successfully forward transit traffic to that destination. Transit traffic passes through the forwarding plane only and is never sent to or processed by the control plane. By processing transit traffic through the forwarding plane only, platforms running the Junos OS can achieve predictably high performance rates.

Transit traffic can be both unicast and multicast traffic. Unicast transit traffic enters one ingress port and is transmitted out exactly one egress port toward its destination. Although multicast transit traffic also enters the transit device through a single ingress port, it can be replicated and sent out multiple egress ports depending on the number of multicast receivers and the network environment.

Exception Traffic: Part 1

Unlike transit traffic, exception traffic does not pass through the local device but rather requires some form of special handling. Examples of exception traffic include the following:

- Packets addressed to the chassis, such as routing protocol updates, Telnet sessions, pings, traceroutes, and replies to traffic sourced from the RE;
- IP packets with the IP options field (options in the packet's IP header are rarely seen, but the PFE was purposely designed not to handle IP options; packets with IP options must be sent to the RE for processing); and
- Traffic that requires the generation of Internet Control Message Protocol (ICMP) messages.

ICMP messages are sent to the packet's source to report various error conditions and to respond to ping requests. Examples of ICMP errors include destination unreachable messages, which are sent when no entry is present in the forwarding table for the packet's destination address, and time-to-live (TTL) expired messages, which are sent when a packet's TTL is decremented to zero. In most cases, the PFE process handles the generation of ICMP messages.

Exception Traffic: Part 2

The Junos OS sends all exception traffic destined for the RE over the internal link that connects the control and forwarding planes. The Junos OS rate limits exception traffic traversing the internal link to protect the RE from denial-of-service (DoS) attacks. During times of congestion, the Junos OS gives preference to the local and control traffic destined for the RE. The built-in rate limiter is not configurable.

Platforms Running the Junos OS

Platforms running the Junos OS come in many shapes and sizes and are targeted for a number of deployment scenarios. The platforms running the Junos OS span switching, routing, and security and are well suited for a variety of network environments. As the heart of all these platforms, the Junos OS provides a consistent end-to-end IP infrastructure in small enterprise environments and the largest service provider networks alike. The subsequent paragraphs introduce and provide some details for each product family.

M Series Multiservice Routers

The M Series multiservice routers provide up to 320 Gbps of aggregate half-duplex throughput. The M Series family can be deployed in both high-end enterprise and service-provider environments. Large enterprises deploy M Series

routers in a number of different roles, including Internet gateway router, WAN connectivity router, campus core router, and regional backbone and data center routers. In service-provider environments, the M Series router operates predominantly as a multiservice edge router, but you can also deploy it in small and medium cores, and in peering, route reflector, multicast, mobile, and data-center applications.

For additional, in-depth details on the M Series, go to http://www.juniper.net/us/en/products-services/routing/m-series/.

T Series Core Routers

The T Series core routers provide up to 25.6 Tbps of throughput. The T Series family is ideal for service provider environments and is deployed within the core of those networks.

For additional, in-depth details on the T Series, go to http://www.juniper.net/us/en/products-services/routing/t-tx-series/.

J Series Services Routers

The J Series services routers provide up to 2 Gbps of throughput. The J Series services routers 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, in-depth details on the J Series, go to http://www.juniper.net/us/en/products-services/routing/j-series/.

MX Series Ethernet Services Routers

The MX Series Ethernet services routers provide up to 960 Gbps of aggregate half-duplex throughput. The MX Series family is targeted for dense dedicated access aggregation and provider edge services in medium and large point of presence (POPs). Large enterprise environments and service providers can leverage MX Series Ethernet services routers for a variety of network functions including Ethernet transport and aggregation, and can use them to offer new Ethernet-based services.

For additional, in-depth details on the MX Series Ethernet, go to http://www.juniper.net/us/en/products-services/routing/mx-series/.

EX Series Ethernet Switches

The EX Series Ethernet switches provide up to 6.2 Tbps of full duplex throughput. The EX Series switches are designed for access, aggregation, and core deployments and are well suited for low-density to high-density enterprise and data center environments.

For additional, in-depth details on the EX Series Ethernet switches, go to http://www.juniper.net/us/en/products-services/switching/ex-series/.

SRX Series Services Gateways

The SRX Series services gateways provide up to 120 Gbps of full duplex throughput. The SRX Series family is 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, in-depth details on the SRX Series, go to http://www.juniper.net/us/en/products-services/security/srx-series/.

Review Questions

1. What are some advantages of the Junos OS?
2. What are the primary functions of the control plane and the forwarding plane on Junos devices?
3. How are transit and exception traffic processed?
4. Which platforms run the Junos OS?

Answers

1.

The Junos OS is compartmentalized into multiple software processes. Each process runs in its own protected memory space, ensuring that one process cannot directly interfere with another. This modularity also ensures that new features can be added with less likelihood of breaking current functionality.

2.

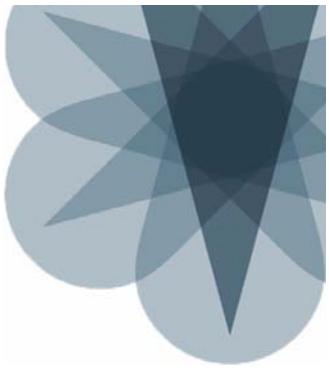
The primary functions of the control plane are to maintain routing intelligence, control and monitor the chassis, and manage the Packet Forwarding Engine (PFE). The primary functions of the forwarding plane are to forward packets and to implement advanced services.

3.

Transit traffic is forwarded through the PFE on platforms running the Junos OS, based on the forwarding table installed on the PFE. Exception traffic is processed locally by the platform running the Junos OS by either the PFE or the RE depending on the type of traffic. Host-bound packets, such as protocol and management traffic, are passed directly to the RE for processing, while traffic requiring ICMP error message responses is typically handled by the PFE.

4.

Platform families that run the Junos OS include M Series, T Series, J Series, MX Series, EX Series, and SRX Series.



JNCIA-Junos Study Guide—Part 1

Chapter 2: User Interface Options

This Chapter Discusses:

- Common user interface options available for platforms running the Junos operating system; and
- The Junos OS command-line system (CLI) and its related modes and features.

The Junos CLI

The Junos CLI is a text-based command shell. One option for accessing the CLI is through the out-of-band (OoB) serial console connection. The console port settings are predefined and are not user configurable.

A second option for accessing the CLI is over the network (in band) using access protocols such as Telnet or SSH. Unlike the console connection, these access options require configuration for a network port and the access protocol.

Many platforms running the Junos OS also offer a dedicated management Ethernet port. This management port provides OoB access; therefore, the software cannot forward transit traffic through this management port. The actual name of the dedicated management Ethernet port varies between platforms. For details on your specific platform, refer to http://www.juniper.net/techpubs/ for the technical publications.

J-Web Interface

The J-Web is a Web-based graphical user interface (GUI) that you access by using either Hypertext Transfer Protocol (HTTP) or HTTP over Secure Sockets Layer (HTTPS). It provides quick configuration wizards to simplify the most common configuration tasks. 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 platforms running the Junos OS.

Logging In

The Junos OS requires a username and a password for access. The administrator creates user accounts and assigns permissions. All platforms running the Junos OS have only the root user configured by default, without any password.

When configured, the console login displays the hostname of the device. When you have not configured a hostname, as is the case with a factory-default configuration, the software displays `Amnesiac` in place of the hostname:

```
Amnesiac (ttyu0)

login: root

--- JUNOS 10.1R1.8 built 2010-02-12 18:31:54 UTC
root@%
```

The root user has complete access and control of the device. When you log in as the root user, the software places you 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, ensure that you also log out of the shell by using the **exit** command.

Operational Mode

In operational mode, you use the CLI to monitor and troubleshoot the device. The **monitor**, **ping**, **show**, **test**, and **traceroute** commands let you display information and test network connectivity for the device.

Configuration Mode

In configuration mode, you can configure all properties of the Junos OS, including interfaces, protocols, and user access, as well as several system hardware properties.

Need Help?

```
user@host> ?
Possible completions:
  clear                Clear information in the system
  configure            Manipulate software configuration information
  file                 Perform file operations
  help                 Provide help information
  . . .

user@host> clear ?
Possible completions:
  arp                  Clear address resolution information
  bfd                  Clear Bidirectional Forwarding Detection information
  bgp                  Clear Border Gateway Protocol information
  dhcp                 Clear DHCP information
  . . .
```

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 do not need to press Enter. If you type the question mark at the command-line prompt, the CLI lists the available commands and options including user-defined variables at the appropriate context. 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. 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 and then redisplays the letters that you typed.

Help on General Concepts

```
user@host> help topic interfaces ?
Possible completions:
  accept-data          Accept packets destined for virtual address
  accept-source-mac    Policers for specific source MAC addresses
  accounting           Packet counts for destination and source classes
  accounting-profile   Accounting profile
  acknowledge-timer    Maximum time to wait for link acknowledgment message
  address              Interface address and destination prefix
  ...

user@host> help topic interfaces address
                       Configuring the Interface Address

   You assign an address to an interface by specifying the address when
   configuring the protocol family. For the inet family, configure the
   interface's IP address. For the iso family, configure one or more
   addresses for the loopback interface. For the ccc, tcc, mpls, tnp, and
   vpls families, you never configure an address.
   ...
```

You can use the **help** command in various ways. The **help topic** command displays usage guidelines for the statement. In the example on the graphic, we receive information on configuring an interface address.

Help with Junos OS Configuration

```
user@host> help reference interfaces address
address

 Syntax

     address address {
         arp ip-address (mac | multicast-mac) mac-address <publish>;
         broadcast address;
        ...
  Hierarchy Level

     [edit interfaces interface-name unit logical-unit-number family family],
     [edit logical-routers logical-router-name interfaces interface-name unit
     logical-unit-number family family]

...
```

The **help reference** command displays summary information for the referenced configuration statement. In the example on the graphic, once again, we are seeking help with interface addressing. Although not shown on the graphic, the **help reference** command displays a complete list of related configuration options along with several other details specific to the referenced command statement.

In addition to the **help topic** and **help reference** commands, the Junos OS also offers the **help apropos** command. The **help apropos** command displays the contexts (typically **set** commands) that reference a specified variable. The following is an example of the **help apropos** command:

```
[edit system archival configuration]
user@host# help apropos archive
set archive-sites
    List of archive destinations
set archive-sites <url> password <password>
    Password for login into the archive site
```

The **help apropos** command only displays contexts that are relevant to the configuration hierarchy level at which you are currently positioned. In other words, if you entered the sample command shown, at the `[edit]` hierarchy level you would see all possible references rather than just those that are applicable to the `[edit system archival configuration]` hierarchy level.

Spacebar Completion for Commands

The CLI provides a completion function. Therefore, you are not always required to type the full command or the command option name for the CLI to recognize it.

To complete a command or option that you have partially typed, press the Spacebar. 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 have entered an ambiguous command, and it displays the possible completions.

The command completion option is on by default, but you can turn it off. To disable command completion for an individual user's session, issue the **set cli complete-on-space off** command as follows:

```
user@host> set cli complete-on-space off
Disabling complete-on-space
```

Tab Completion for Commands and Variables

You can use the Tab key to complete system commands and user-defined variables. Examples of variables include policy names, AS paths, community names, and IP addresses. The Tab key also offers a list of possible completions

if multiple, ambiguous options exist. Command completion allows you to save time by reducing your keystrokes, and prevents errors by accurately referencing the desired user-defined variables.

EMACS-Style Control Keys

The CLI supports EMACS-style keyboard sequences that allow you to move the cursor on a command line and delete specific characters or words. The following are supported sequences:

- *Ctrl+b*: Moves the cursor left one character;
- *Ctrl+a*: Moves the cursor to the beginning of the command line;
- *Ctrl+f*: Moves the cursor right one character;
- *Ctrl+e*: Moves the cursor to the end of the command line;
- *Delete* and *Backspace*: Deletes the character before the cursor;
- *Ctrl+d*: Deletes the character over the cursor;
- *Ctrl+k*: Deletes from the cursor to the end of the line;
- *Ctrl+u*: Deletes all characters and negates the current command;
- *Ctrl+w*: Deletes the entire word to the left of the cursor;
- *Ctrl+l*: Redraws the current line;
- *Ctrl+p*, *Ctrl+n*: Repeats the previous and next command in the command history, respectively;
- *Esc+d*: Deletes the word to the right;
- *Esc+b*: Moves the cursor back one word with no delete; and
- *Esc+f*: Moves the cursor forward one word with no delete.

Please note that when using the Esc key, you must release the key and press it again for each occurrence. This action differs from the Ctrl key, which you can hold down for multiple occurrences.

VT100 Terminal Type

The Junos OS defaults to a VT100 terminal type. This terminal type enables the use of keyboard Arrow keys without any additional session or configuration modification.

Using Pipe

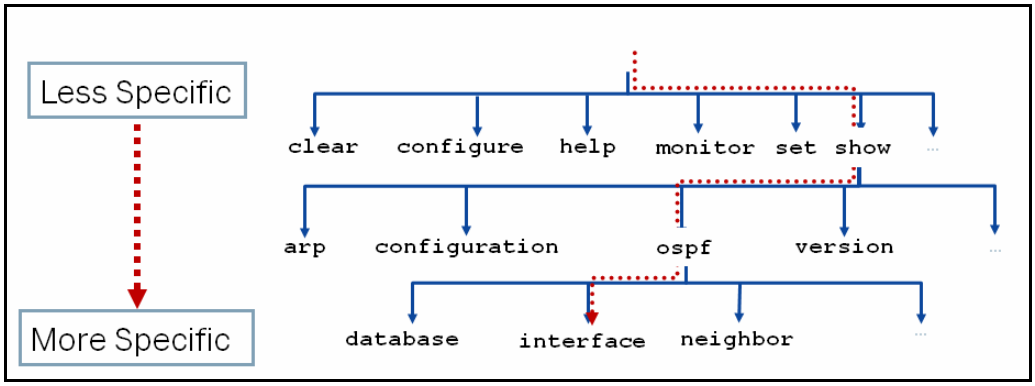
For operational and configuration commands that display output, such as the **show** commands, you can filter the output. When help is displayed for these commands, one of the options listed is |, called a pipe, which allows the command output to be filtered. To filter the output of an operational mode or a configuration mode command, add a pipe and an option to the end of the command. The following are available options:

- **compare** (***filename*** | **rollback** ***n***): Available in configuration mode using only the **show** command. Compares configuration changes with another configuration file.
- **count**: Displays the number of lines in the output.
- **display changed**: Available in configuration mode only. Tags changes with `junos:changed` attribute only for XML use.
- **display commit-scripts**: Shows data after the Junos OS applies commit scripts.
- **display detail**: Available in configuration mode only. Displays additional information about the contents of the configuration.
- **display inheritance**: Available in configuration mode only. Displays inherited configuration data and source group.

- **display omit**: Available in configuration mode only. Omits configuration statements with the **omit** option.
- **display set**: Available in configuration mode only. Shows **set** commands that created configuration statements.
- **display xml**: Displays the output in NETCONF/XML format.
- **except *regular-expression***: Ignores text matching a regular expression when searching the output. If the regular expression contains spaces, operators, or wildcard characters, you must enclose it in quotation marks.
- **find *regular-expression***: Displays the output starting at the first occurrence of text matching a regular expression. If the regular expression contains spaces, operators, or wildcard characters, you must enclose it in quotation marks.
- **hold**: Holds text without exiting the `--(more)--` prompt.
- **last**: Displays the last screen of information.
- **match *regular-expression***: Searches for text matching a regular expression. If the regular expression contains spaces, operators, or wildcard characters, you must enclose it in quotation marks.
- **no-more**: Displays output all at once rather than one screen at a time.
- **request message**: Displays output to multiple users.
- **resolve**: Converts IP addresses to Domain Name System (DNS) names. Truncates to fit original size unless you specify **full-names**.
- **save *filename***: Saves the output to a file or URL.
- **trim**: Trims specified number of columns from the start line.

You can cascade multiple instances of the CLI's pipe functionality, which can be very beneficial when you must search extensive outputs displayed through the CLI for specific information. Subsequently, we highlight the required syntax to evoke logical AND and logical OR searches within extensive outputs and files.

Operational Mode



You use operational mode CLI commands to monitor and control the operation of a device running the Junos OS. The operational mode commands exist in a hierarchical structure, as shown on the graphic. 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 (OSPF) protocol. Specifying the **interface** option, as in the **show ospf interface** command, outputs information on OSPF interfaces.

The Junos OS also adds additional flexibility through the **run** command, which allows you to issue operational mode commands while in configuration mode. We cover the **run** command in detail later.

Operational Mode Capabilities

Key operational mode capabilities include the following:

- Entering configuration mode;
- Controlling the CLI environment;
- Exiting the CLI;
- Monitoring and troubleshooting:
  - **clear**
  - **monitor**
  - **mtrace**
  - **ping**
  - **show**
  - **test**
  - **traceroute**;
- Connecting to other network systems;
- Copying files;
- Restarting software processes; and
- Performing system-level operations.

Batch Configuration Changes

Unlike software from other vendors, configuration changes made in the Junos OS do not take effect immediately. This design feature allows you to group together and apply multiple configuration changes to the running configuration as a single unit.

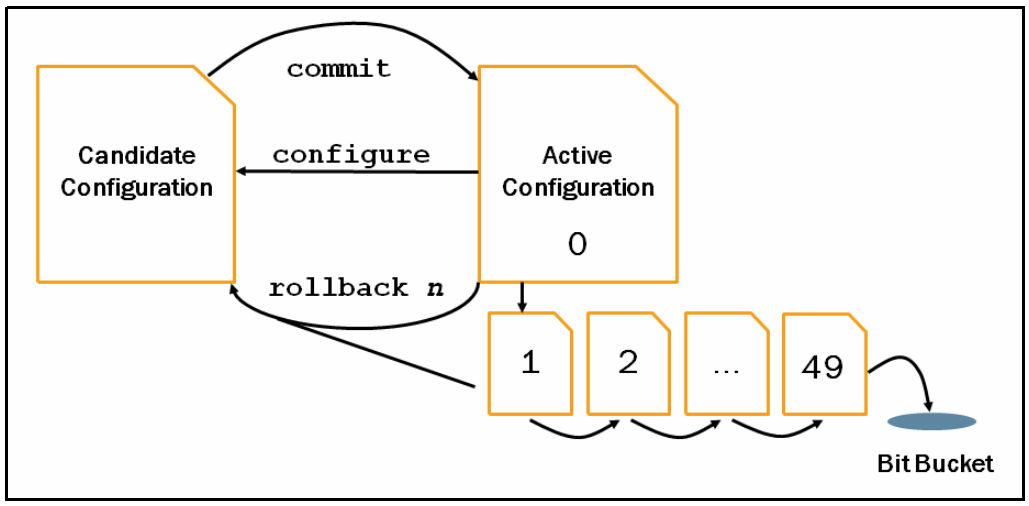
Active Configuration

The active configuration is the configuration currently operational on the system and is the configuration the system loads during the boot sequence. This concept is analogous to both the *running configuration* and *startup configuration* in software from other vendors.

Candidate Configuration

The candidate configuration is a temporary configuration that might possibly become the active configuration. When you configure a device running the Junos OS, the software creates a candidate configuration and initially populates it with the active configuration running on that device. You then modify the candidate configuration. Once satisfied with your modifications, you can commit the changes. This action causes the candidate configuration to become the active configuration.

The Life of a Configuration File: An Overview



The **configure** command causes a candidate configuration to be created and populated with the contents of the active configuration. You can then modify the candidate configuration with your changes.

To have a candidate configuration take effect, you must commit the changes. At this time, the Junos OS checks the candidate configuration for proper syntax and it installs it as the active configuration. If the syntax is not correct, an error message indicates the location of the error, and the software does not activate any part of the configuration. You must correct the errors before recommitting the configuration.

You can easily recover previous configurations by using a **rollback *n*** command. The Junos OS maintains a configuration history by storing previously active configurations. The software saves a maximum of 50 configurations. This number includes the current active configuration, which is also known as `rollback 0`, and up to 49 previously active configurations. If you perform a rollback operation, keep in mind that the related configuration does not become active until you issue a **commit**. When you issue a **commit** and 50 rollback configurations exist, the software purges the last rollback configuration—rollback 49.

Starting Configuration Mode

- Type `configure` at the operational mode prompt to enter configuration mode:

```
user@host> configure
Entering configuration mode

[edit]
user@host#
```

- Use `configure exclusive` to exclude other users from editing the configuration
  - Any uncommitted changes are discarded when users exit:

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

[edit]
user@host#
```

You enter configuration mode by issuing the **configure** command from the CLI's operational mode. If, when you enter configuration mode, another user is also in configuration mode, a message indicates who the user is and what portion of the configuration the user is viewing or editing.

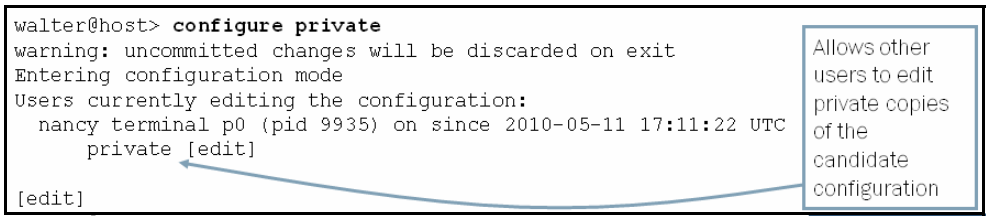
In configuration mode, the prompt changes from the angle bracket (>) of operational mode to the pound sign (#), preceded by the name of the user and the name of the device.

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

Exclusive Configuration

By default, multiple users can enter configuration mode and commit changes. Use the **configure exclusive** command to allow only a single user to edit the configuration. Uncommitted changes are always discarded when you use the **configure exclusive** command. In contrast, uncommitted changes are retained when you use the standard **configure** command.

Private Configuration



Entering configuration mode using the **configure private** command allows multiple users to edit the configuration while committing only their private changes. (You must issue a **commit** command from the `[edit]` hierarchy.) If private users issue a **rollback 0** command, the software discards only their changes.

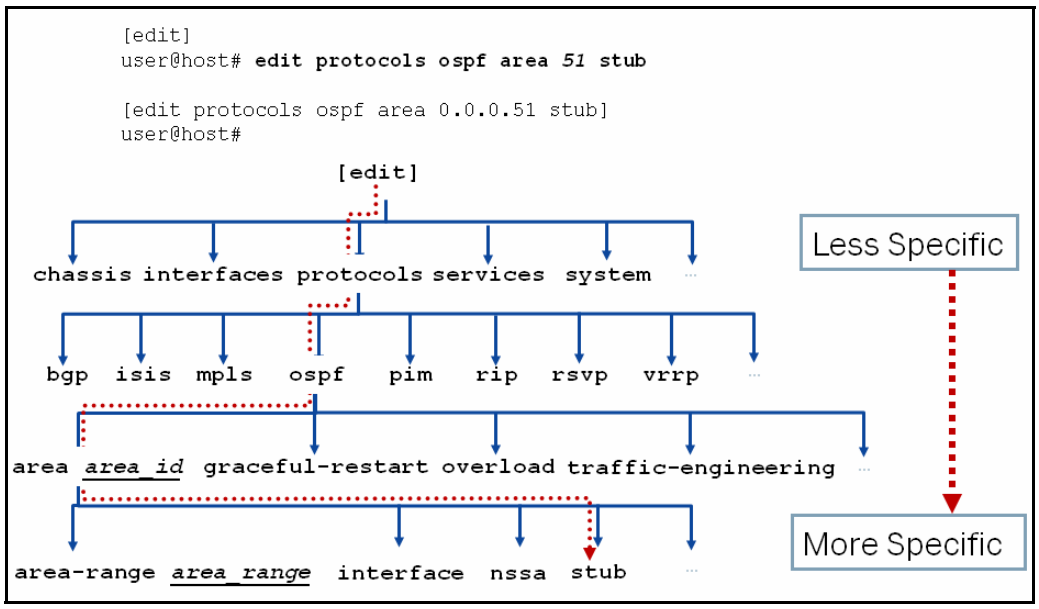
When a user is in private mode, other users must enter private mode or use **configure exclusive** to become the master, or they cannot modify the candidate configuration. Exiting private configuration without committing changes results in the loss of any modifications made to the private candidate configuration.

If two users are in private mode and both make the same change (For example, User 1 changes the system hostname to *apples* while User 2 sets the hostname to *oranges*), the second **commit** will fail with an error message to avoid configuration conflicts. The software places the second user's changes into effect if User 2 issues a second **commit** command.

When chassis clustering is in effect, the **configure private** command is automated. In some other environments, you might want to require users to use only **configure private**. When creating user accounts, it is possible to limit the commands available to users through the assigned properties. We discuss user accounts and their assigned properties later in this material.

If a user is in configuration mode and has altered the candidate configuration, other users cannot enter configuration mode using the **exclusive** or **private** options. The changes made by the first user must be committed or canceled prior to any other users entering configuration mode with the **exclusive** or **private** options.

Statement Hierarchy



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, CLI operational mode includes a **show** command to display specific operational information, while the CLI configuration mode provides a **show** command to display the statement hierarchy. The two commands are independent of each other.

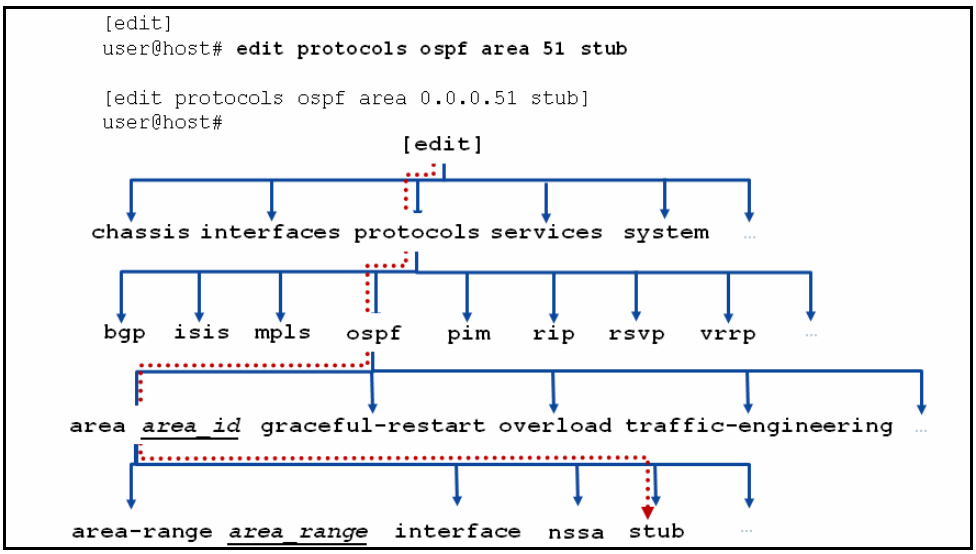
The software organizes the statement hierarchy in a tree structure similar to Windows folders or UNIX directories, grouping related information into a particular branch of the tree.

Hierarchical Configuration

Use **set** commands in the CLI configuration mode to modify the candidate configuration. Use the **show** command to display the candidate configuration. Both commands are relative to the current configuration hierarchy, shown by the `[edit]` prompt.

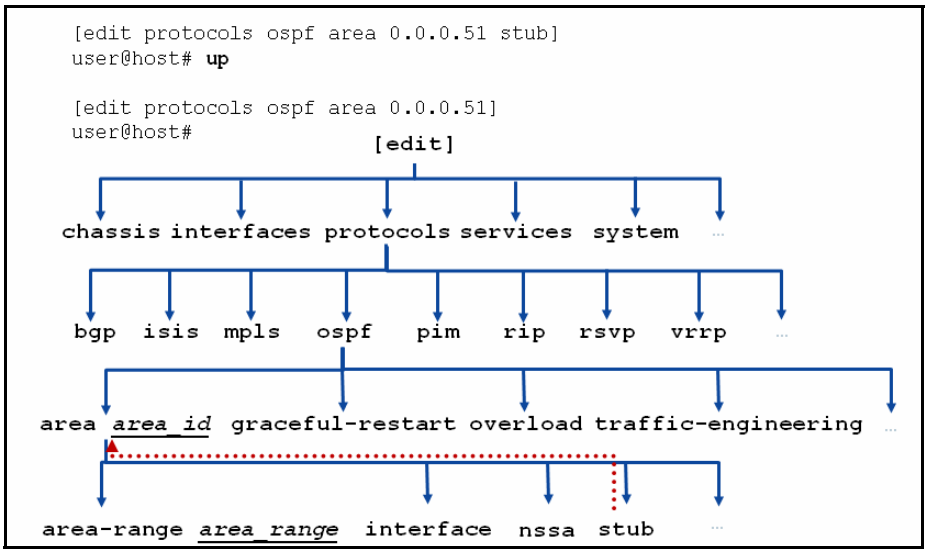
Configuration files use curly brackets (`{}`) and indentation to visually display the hierarchical structure of the configuration. Terminating—or leaf—statements in the configuration hierarchy are displayed with a trailing semicolon (`;`). You enter neither the curly brackets nor the semicolons as part of the **set** command.

Moving Between Levels Is Like Changing Directories



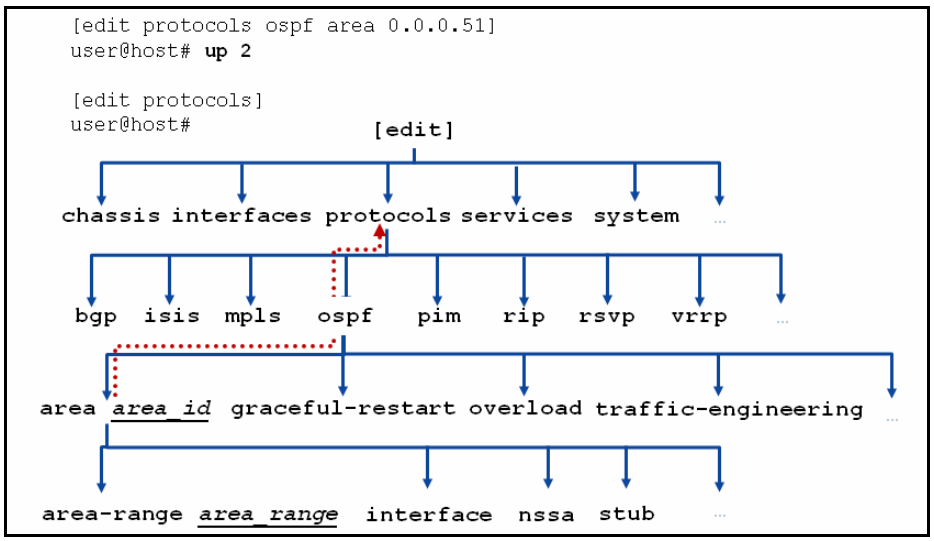
To move down through an existing configuration statement hierarchy or to create a hierarchy and move down to that level, use the **edit** command, specifying your desired hierarchy level. After you issue an **edit** command, the configuration mode banner changes to indicate your current level in the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

Moving Up One Level



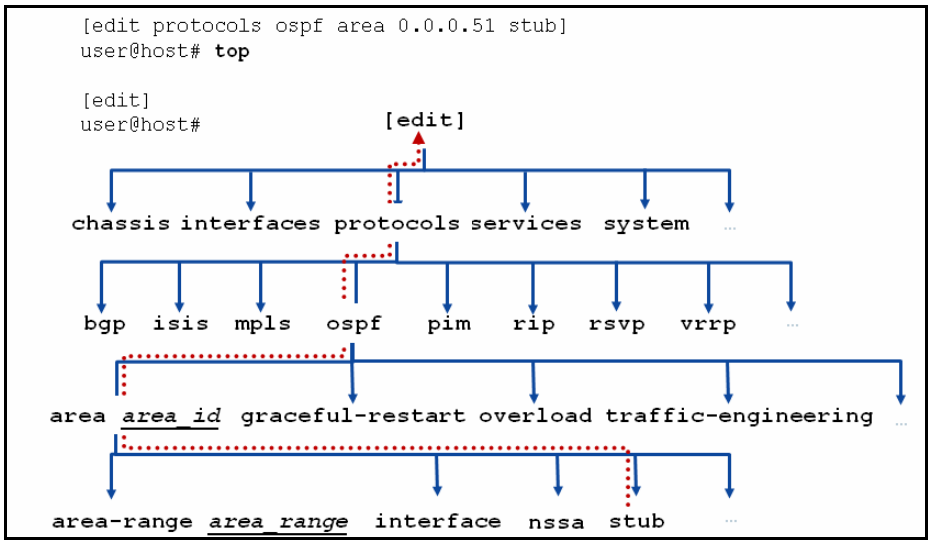
To move up one level from the current position in the hierarchy, use the **up** command.

Moving Up More Than One Level



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

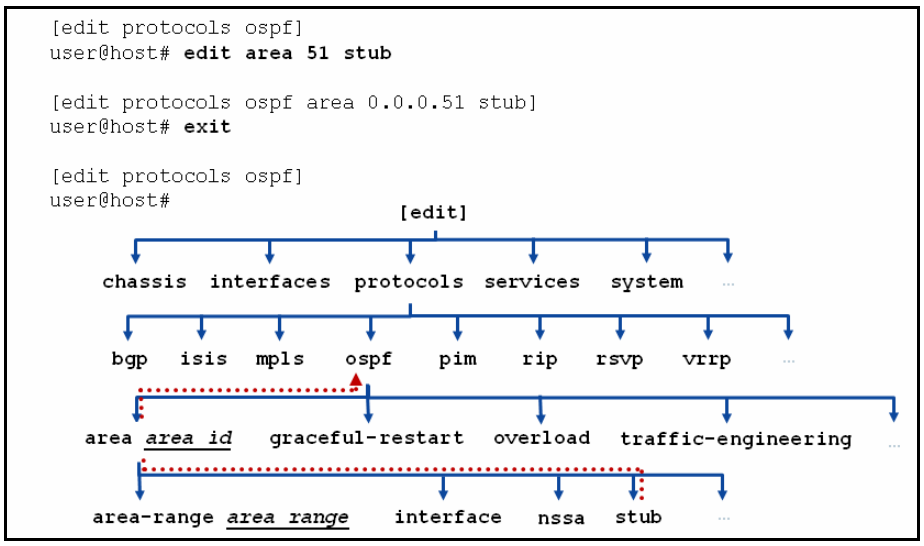
Take Me to the Top



The **top** command quickly moves you to the top of the configuration hierarchy. You can combine **top** with **edit** to move quickly to a different hierarchy or with **show** to display the configuration details for a different hierarchy, as in the following example:

```
[edit protocols ospf area 0.0.0.0 interface ge-0/0/0.0]
user@host# top edit system login

[edit system login]
user@host# top show system services
ftp;
ssh;
```

Back to Where I Was Before



As the example on the graphic illustrates, the **exit** command returns the user to the most recent, higher level of the hierarchy. Entering **exit** at the top level of the hierarchy exits configuration mode, as follows:

```
[edit]
user@host# exit
Exiting configuration mode

user@host>
Entering exit configuration-mode from any level of the hierarchy also allows
you to exit configuration mode, as in the following example:
[edit protocols ospf area 0.0.0.0 interface ge-0/0/0.0]
user@host# exit configuration-mode
Exiting configuration mode

user@host>
```

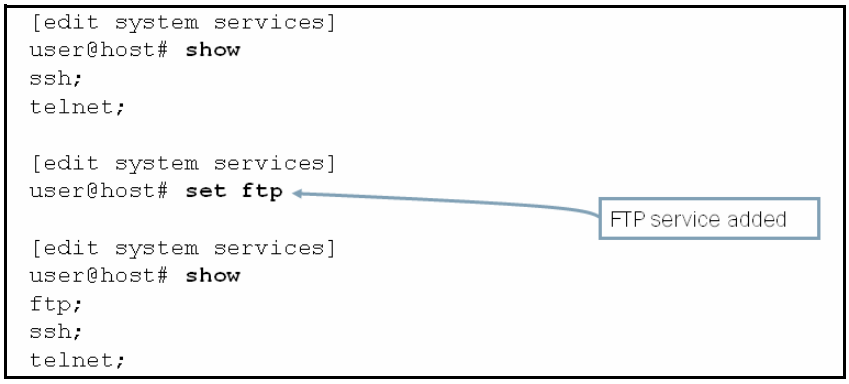
In Summary

- **edit** functions like a CD command
- **up** moves up one level
- **up** *n* moves up *n* levels
- **top** moves to the top of the hierarchy
- **exit** moves to the previous, higher level in the hierarchy or exits configuration mode if at the top level of the hierarchy

```
[edit]
user@host# edit protocols ospf area 51 stub
[edit protocols ospf area 0.0.0.51 stub]
user@host# up
[edit protocols ospf area 0.0.0.51]
user@host# up 2
[edit protocols]
user@host# top
[edit]
user@host# exit
The configuration has been changed but not committed
Exit with uncommitted changes? [yes,no] (yes)
```

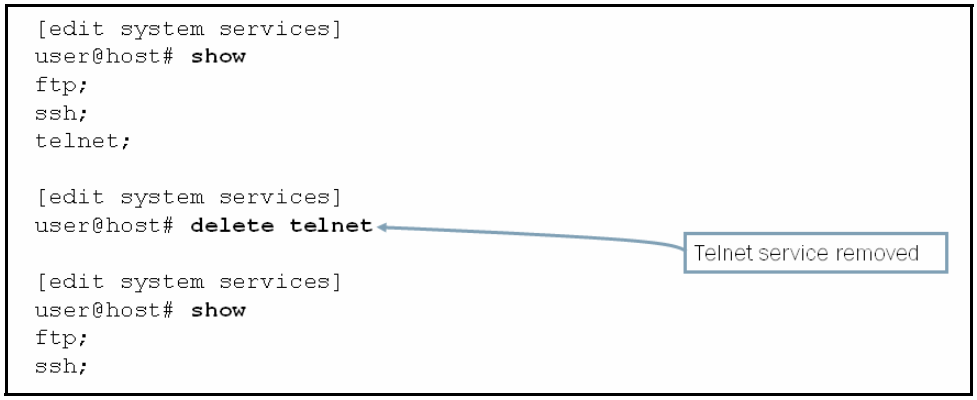
You can quickly navigate between levels of the configuration hierarchy using the **edit**, **up**, **top**, and **exit** commands.

Adding Configuration Statements



Use **set** commands in the CLI configuration mode to modify the candidate configuration.

Removing Configuration Statements



Use the configuration mode **delete** command to remove statements that you previously added to the configuration with a **set** command. This command deletes the statement and all its subordinate statements and identifiers. Deleting a statement or an identifier effectively unconfigures the functionality associated with that statement or identifier, returning that functionality to its default condition.

Consider using the **wildcard delete** function when deleting individual statements is too arduous and deleting an entire configuration subhierarchy lacks the granularity that you need. The following example shows sample syntax for a **wildcard delete**:

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

[edit]
user@host#
```

In addition to deleting configuration statements, you should also consider the use of the **deactivate** command to cause the specified portion of the configuration hierarchy to be ignored while still retaining the original configuration. Issue an **activate** command to place the configuration back into effect. We provide an example of the **deactivate** and **activate** commands in a subsequent section.

Using Configuration Mode Efficiently: Part 1

• **rename** a configuration statement

```
[edit]
user@host# rename interfaces ge-0/0/10 to ge-0/0/11
```

• **replace pattern** of configuration statements

```
[edit]
user@host# replace pattern ge-0/0/10 with ge-0/0/11
```

• **copy** a configuration statement to another statement

```
[edit]
user@host# copy interfaces ge-0/0/10 to ge-0/0/11
```

Using configuration commands can increase efficiency. The following output illustrates the full list of configuration mode commands:

```
[edit]
user@host# ?
Possible completions:
  <[Enter]>            Execute this command
  activate             Remove the inactive tag from a statement
  annotate             Annotate the statement with a comment
  commit               Commit current set of changes
  copy                 Copy a statement
  deactivate           Add the inactive tag to a statement
  delete               Delete a data element
  edit                 Edit a sub-element
  exit                 Exit from this level
  extension            Extension operations
  help                 Provide help information
  insert               Insert a new ordered data element
  load                 Load configuration from ASCII file
  quit                 Quit from this level
  rename               Rename a statement
  replace              Replace character string in configuration
  rollback             Roll back to previous committed configuration
  run                  Run an operational-mode command
  save                 Save configuration to ASCII file
  set                  Set a parameter
  show                 Show a parameter
  status               Show users currently editing configuration
  top                  Exit to top level of configuration
  up                   Exit one level of configuration
  wildcard             Wildcard operations
[edit]
user@host#
```

Regardless of the method and commands you use to update your configuration file, you must issue the **commit** command to activate changes.

Using Configuration Mode Efficiently: Part 2

- **deactivate** or ignore a configuration statement

```
[edit]
user@host# deactivate interfaces ge-0/0/10
```

- **insert** a configuration statement in another location

```
[edit policy-options policy-statement test]
user@host# insert term three before term two
```

- **annotate** a comment to a configuration statement

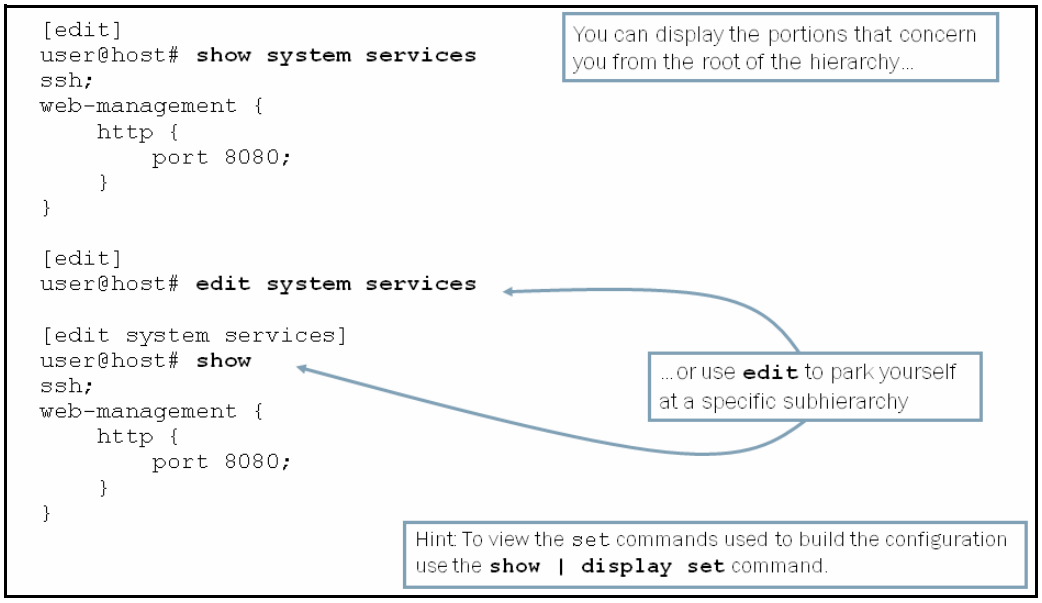
```
[edit system]
user@host# annotate name-server "adding new name servers"
```

The following example shows the **deactivate**, **activate**, **annotate**, and **commit** commands and their output:

```
[edit]
user@host# deactivate interfaces ge-0/0/0

[edit]
user@host# commit
commit complete

[edit]
user@host# show interfaces ge-0/0/0
##
## inactive: interfaces ge-0/0/0
##
unit 0 {
    family inet {
        address 10.210.11.177/28;
    }
    family inet6;
}

[edit]
user@host# activate interfaces ge-0/0/0

[edit]
user@host# commit
commit complete

[edit]
user@host# show interfaces ge-0/0/0
unit 0 {
    family inet {
        address 10.210.11.177/28;
    }
    family inet6;
}

[edit system]
user@host# annotate name-server "added new name servers"

[edit system name-server]
user@host# show
```

```
/* added new name servers */
205.152.144.23;
205.152.132.23;
```

Viewing the Candidate Configuration



Use the configuration mode **show** command to display the candidate configuration. This command displays the configuration at the current hierarchy level or at the specified level below the current location.

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 path.

You can display the individual set commands used to create the existing configuration file using the **show | display set** command. The following is an example of this command and its resulting output:

```
[edit]
user@host# show system services | display set
set system services ssh
set system services web-management http port 8080
```

Remember to Commit

Remember, Junos devices do not automatically apply your configuration changes. You must use the **commit** command to activate your candidate configuration. You can typically perform the commit operation 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.

On devices with redundant Routing Engines, you can perform a **commit synchronize**, which activates and synchronizes the configuration on both Routing Engines, as shown in the following capture:

```
{master:0}[edit]
user@host# commit s?
```

```
Possible completions:
  synchronize          Synchronize commit on both Routing Engines
```

Alternatively, you can configure the system to automatically perform the synchronize operation when a standard **commit** is issued through the **set system commit synchronize** command.

Checking Configuration Syntax

When you commit a candidate configuration, the software activates the entire configuration in its current form. Use the **commit check** command to validate the syntax of a candidate configuration without actually placing it into effect.

Remote Configuration Is Risky

Of course, **commit check** cannot catch logical errors in your configuration. What happens when you are configuring a device remotely and make a mistake that leaves that device inaccessible to remote connections? You can solve this scenario 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 a second **commit** does not occur within the time-out value specified (the software supports a range of 1 to 65,535 minutes, with 10 minutes being the default), the system performs a **rollback *1***, **commit** sequence on your behalf. After the automatic rollback, you can load the `rollback 1` file to look for your mistake. We discuss the **rollback** command and operation in detail later.

Scheduled Commits

- Use `commit at` to schedule a future commit:

```
[edit]
user@host# commit at 21:00:00
     configuration check succeeds
     commit at will be executed at 2010-05-11 21:00:00 UTC
     Exiting configuration mode
```

- Use `commit comment` to add comments:

```
[edit]
user@host# commit comment "Changed OSPF configuration"
commit complete

user@host> show system commit
0   2010-05-11 15:32:42 UTC by user via cli
    Changed OSPF configuration
…
```

- Use `commit and-quit` to save time:

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

user@host>
```

You can also schedule a commit that occurs at a specific time using the **commit at** ***time*** command. This command is useful for synchronizing commits with multiple routers. These routers must have their time synchronized to the same source (likely through NTP) for the commit operations to execute at the same time. To view and clear pending commits, use the **show system commit** and **clear system commit** commands:

```
user@host> show system commit
commit requested by user via cli at 2010-05-11 21:00:00 UTC
0   2010-05-11 15:32:42 UTC by user via cli
...
user@host> clear system commit
Pending commit cleared
```

Adding a Log Entry to Your Commit

You can also add a log entry to your commit using the **commit comment** ***"comment-string"*** option. As illustrated on the previous graphic, these logs are visible in the output of the **show system commit** command.

Exiting Configuration Mode

You can add the **and-quit** option to the **commit** command to activate your changes and exit configuration mode in a single step.

Viewing Differences

Using **show | compare** displays the differences between the candidate configuration and the active configuration, also known as `rollback 0`. Configuration comparison is patch-like and context sensitive. Thus, instead of showing the entire configuration, the display shows only the actual changes.

Comparing Active and Rollback Configurations

Using the operational mode **show configuration | compare rollback** ***number*** command allows you to view differences between the active configuration and the rollback configurations. The Junos OS can store up to forty-nine additional rollback configurations in addition to `rollback 0`, which is the active configuration.

Similarly, the **show configuration | compare** ***filename*** command allows you to compare the active configuration to an arbitrary file. You can also use **show | compare rollback** ***number*** and **show | compare** ***filename*** in configuration mode to compare the candidate configuration with rollback configurations and arbitrary files, respectively.

Viewing Differences in Other Files

The operational mode **file compare files** command allows you to view differences between any two text files, including log files. The output of this command is in the same patch-like format as the **show | compare** command.

Restoring a Previous Configuration

The software saves the last 50 committed versions of the configuration. To overwrite the candidate configuration with one of these previously committed versions, 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, include the version number in the **rollback** command.

The ***version*** argument can be a number in the range 0 through 49. The most recently saved configuration is version 0, which is the active configuration. The oldest committed configuration the software automatically saves is version 49.

The factory-default configuration on some of the smaller Junos devices restricts the number of rollback files stored by the system. This default setting can be changed to increase the number of rollback files as shown in the following capture:

```
[edit system]
user@host# set max-configurations-on-flash ?
Possible completions:
  <max-configurations-on-flash>  Number of configuration files stored on flash
```

You Must Commit

The **rollback** command modifies only the candidate configuration. To activate the changes loaded through the rollback operation, issue the **commit** command.

The Life of a Configuration File: A Review

The **configure** command causes a candidate configuration to be created and populated with the contents of the active configuration. You can then modify the candidate configuration with your changes.

To have a candidate configuration take effect, you must commit the changes. At this time, the Junos OS checks the candidate configuration for proper syntax and it installs it as the active configuration. If the syntax is not correct, an error message indicates the location of the error, and the software does not activate any part of the configuration. You must correct the errors before recommitting the configuration.

You can easily recover previous configurations with a **rollback *n*** command. The Junos OS maintains a configuration history by storing previously active configurations. The software saves a maximum of 50 configurations. This number includes the current active configuration, which is also known as `rollback 0`, and up to 49 previously active configurations. If you perform a rollback operation, keep in mind that the related configuration does not become active until you issue a **commit** command. When you issue a **commit** command and there are 50 rollback configurations, the software purges the last rollback configuration—rollback 49.

Saving Configuration Files

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 you are saving only the configuration statements at the current hierarchy level and below. To save the entire candidate configuration, you must be at the top level of the configuration hierarchy. If you do not specify a path, the Junos OS 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.

You can specify a filename in one of the following ways:

- *filename* or *path/filename*.
- `ftp://`*user:password@host/path/filename*: 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.
- `scp://`*user@host/path/filename*: Puts the file on a remote system using the SSH protocol. The software prompts you for the user's password.

Loading Configuration Files

You can use the configuration mode **load** command to load 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.

The following list provides details for some of the arguments to the **load** command:

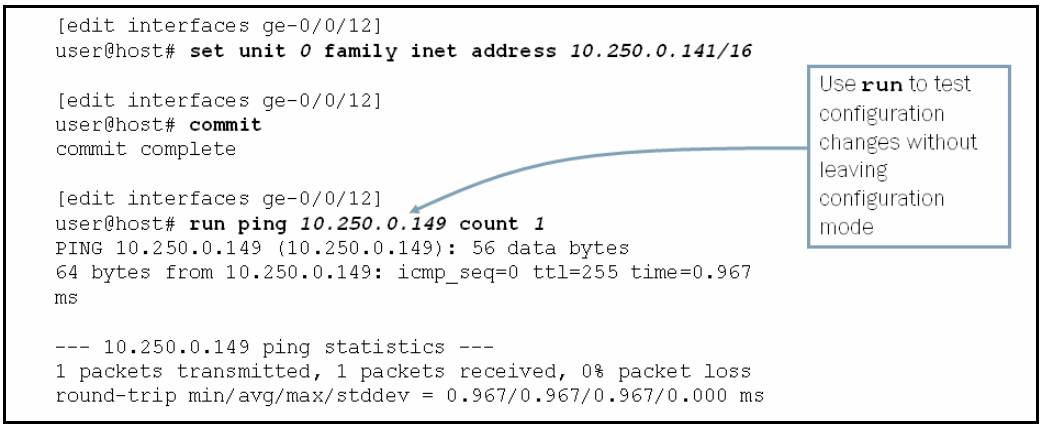
- **factory-default**: Replaces the full current configuration with the factory-default configuration.
- **merge**: Combines the current configuration with the configuration you load.
- **override**: Completely overwrites the current configuration with the configuration you load. You must perform override 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 you load. The software replaces existing statements of the same name 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 that consists of **set** configuration statements.
- **update**: Updates the existing configuration with the configuration you load. When the **update** option is used, the Junos OS attempts to notify only those processes affected by the configuration changes. When the **override** option is used, the Junos OS makes no such attempt. You can use the **update** option from any hierarchy, but you can use the **override** option only 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-and-paste functionality to copy and paste configuration data from one system to another.
- **relative**: Normally, a **load merge** or **load replace** operation requires that the data you load contains a full path to the related configuration hierarchy. The **relative** option negates this need by telling the device to add the data you load *relative* to the current configuration hierarchy.

commit Activates Candidate Configuration

In all cases, after the **load** operation is complete, you must issue a **commit** to activate the changes made to the configuration.

run Baby run



The **run** command allows you to execute operational mode commands while in configuration mode. It is similar to the **do** command on equipment from other vendors, but much more flexible. This extremely handy time-saver works for all operational mode commands and the software supports it at all configuration hierarchies. In the example on the graphic, we are editing the configuration for the device'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.

Review Questions

1. Which modes exist within the Junos OS?
2. Which operations can be performed in each mode?
3. Which keystrokes complete a system command and a user-defined variable?
4. Which command provides the quickest method of returning to the top of the hierarchy?
5. What is the difference between the active and candidate configurations?
6. Which command displays the differences between the candidate and active configurations?

Answers

1.

Two primary modes exist within the Junos OS: the operational mode and the configuration mode. A third mode also exists in the form of the FreeBSD shell.

2.

You use the operational mode to monitor and troubleshoot the software, network connectivity, and hardware. You use the configuration mode to configure a device running the Junos OS, including interfaces, protocols, user access, and system hardware.

3.

You use the Spacebar to complete a command and the Tab key to complete a variable.

4.

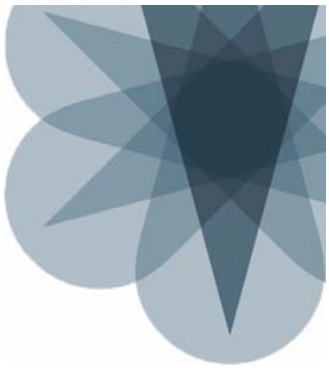
The **`top`** command is the quickest method of returning to the top of the hierarchy.

5.

The active configuration has been committed and is in use, whereas the candidate configuration is not active until you perform a **`commit`** operation.

6.

The **`show | compare`** command displays the differences between the current active and candidate configurations



JUNIPER NETWORKS

JNCIA-Junos Study Guide—Part 1

Chapter 3: Initial Configuration

This Chapter Discusses:

- The factory-default configuration for platforms running the Junos operating system;
- Initial configuration tasks performed on devices running the Junos OS; and
- Interface types and interface configuration basics.

The Factory-Default Configuration

All platforms running the Junos OS are shipped with a factory-default configuration. All factory-default configurations allow access using the root account. The root account does not include a password by default. Setting a root password is required before activating any changes to the configuration file.

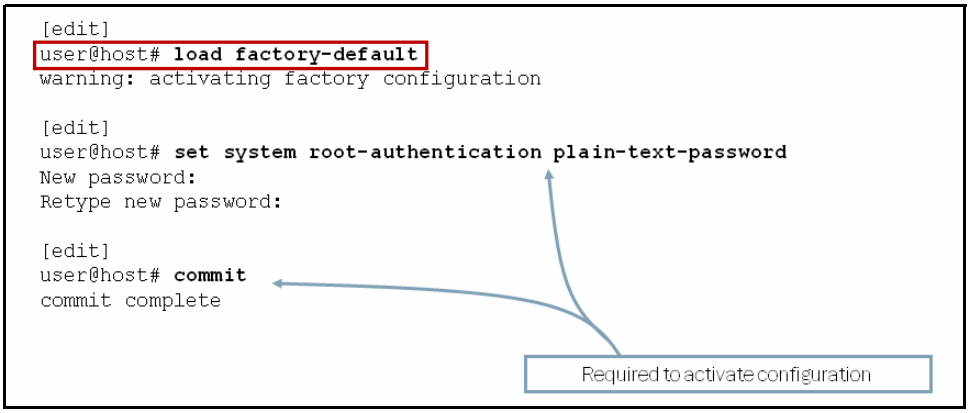
All factory-default configurations also include system logging, which tracks system events and writes those events to predefined log files. The following is an example of a typical syslog configuration found within a factory-default configuration:

```
[edit]
user@host# show system syslog
user * {
    any emergency;
}
file messages {
    any any;
    authorization info;
}
file interactive-commands {
    interactive-commands any;
}
```

We discuss system logging in greater detail in “Secondary System Configuration.”

Factory-default configurations can vary from one platform family to another or even between the different models within the same platform family. All platforms running the Junos OS are designed for specific roles within a network environment and their factory-default configurations are created with those specific roles in mind. One example is the EX Series switches, which are designed to operate as Layer 2 switches right out of the box. To meet this default operational requirement, the associated factory-default configurations have all interfaces configured for Layer 2 operation and also include protocol configuration commonly used on switches, such as the Rapid Spanning Tree Protocol (RSTP) and the Link Layer Discovery Protocol (LLDP). Other platforms do not have these same default operational requirements and thus do not include those configuration parameters in their factory-default configurations.

Loading a Factory-Default Configuration



Under certain conditions, you might want to return a device running the Junos OS to its factory-default configuration. You can overwrite the candidate configuration while in configuration mode using the **load factory-default** command. The Junos OS does not allow you to save the configuration until you configure root authentication information. Do not forget to issue a **commit** to activate your changes.

Powering On a Device Running the Junos OS

```
user@host> request system halt ?
Possible completions:
  <[Enter]>            Execute this command
  at                   Time at which to perform the operation
  in                   Number of minutes to delay before operation
  media                Boot media for next boot
  message              Message to display to all users
  |                    Pipe through a command
```

Always refer to your platform-specific documentation and follow the safety guidelines when connecting power and powering on your device running the Junos OS. Once a device running the Junos OS is powered on and if power to that system is interrupted, the device automatically powers on when the power is restored. In other words, no manual intervention is required for the system to reboot in this situation.

Gracefully Shutting Down the Junos OS

The Junos OS is a multitasking environment. To ensure file system integrity, you should always gracefully shut down platforms running the Junos OS. Although unlikely, failure to gracefully shut down the system could possibly leave it unable to boot. As illustrated in the graphic, you use the **request system halt** command to gracefully shut down the Junos OS. This command provides options that allow you to schedule the shutdown in a specified number of minutes or at an exact time, to specify the media from which the next boot up operation will use, and to log a message to the console and to the messages file.

For Junos devices that offer redundant Routing Engines (REs), you can halt both REs simultaneously using the **request system halt both-routing-engines** command. For EX Series switches participating in a virtual chassis, where multiple switches function as a single virtual device, you can halt all participating members simultaneously with the **request system halt all-members** command.

Initial Configuration Checklist

When you receive a device running the Junos OS from the factory, the Junos OS is preinstalled. Once you power on the device, it is ready to be configured. When the initial configuration is performed, the root authentication must be included. In addition to root authentication, we also recommend that you configure the following items:

- Hostname;
- System time;

- System services for remote access (Telnet, SSH); and
- Management interface and static route for management traffic.

The Junos OS enforces password restrictions. All passwords are required to be no less than six characters and must include a change of case, digits, or punctuation.

The subsequent sections provide sample configuration syntax for the initial configuration tasks.

Logging In as Root

Remember when you receive a platform running the Junos OS from the factory, the root password is not set. To log in to the CLI for the first time, you must log in through the console port using the root username with no password.

When configured, the console login displays the hostname of the device. When no hostname is configured, such as is the case with a factory-default configuration, `Amnesiac` is displayed in place of the hostname.

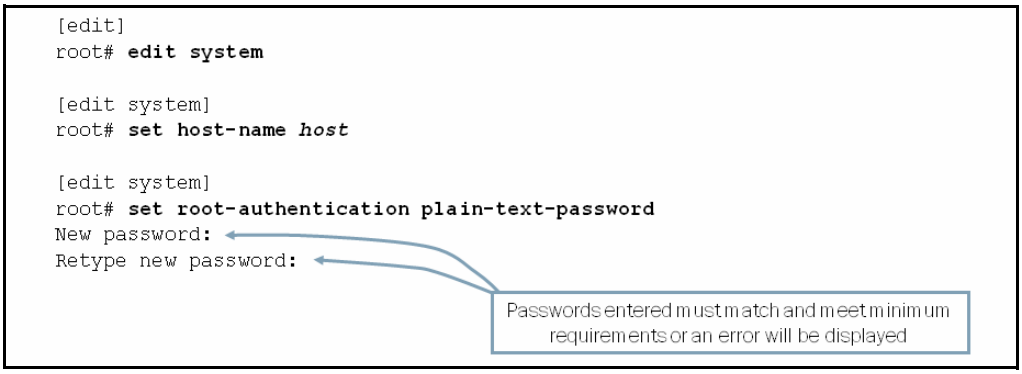
Starting the CLI

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 also log out of the shell using the **`exit`** command.

Entering Configuration Mode

After starting the CLI, you enter operational mode. You can make changes to the configuration only in configuration mode. Enter configuration mode by entering **`configure`** at the operational mode prompt.

Identification Parameters



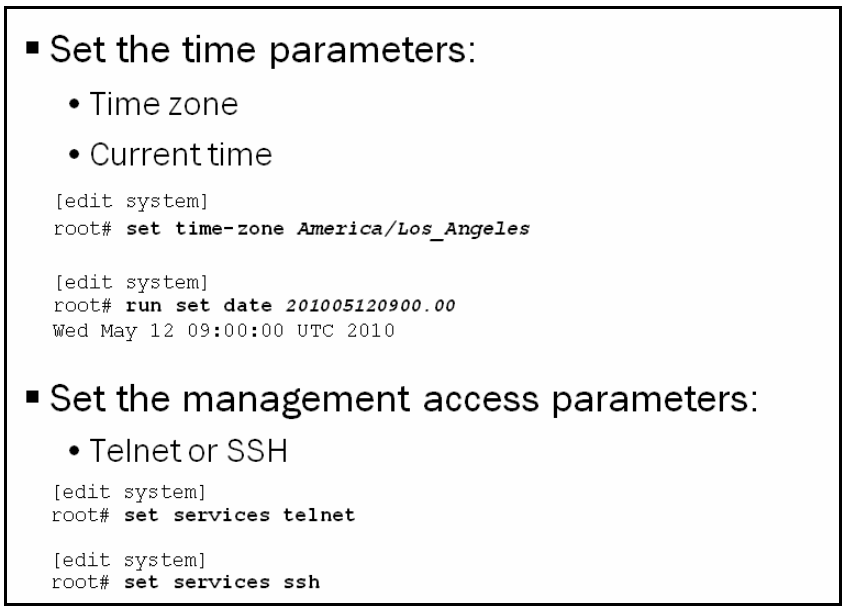
The graphic shows how to use the CLI to configure the hostname and a root password. As displayed on the graphic, a check is made when the root password is entered to ensure that it has been entered correctly. In the event that both entered passwords do not match, an error will be generated, the change is not made, and the password will need to be reentered.

The example on the graphic uses the plain-text authentication option. Unlike the software from some vendors, the Junos OS never actually displays the password in its plain-text format but rather encrypts the password for you. You can see the encrypted password by viewing the relevant configuration:

```
[edit system]
root# show root-authentication
encrypted-password "$1$ti58nUSg$8xnQtTJeA0dA/.eUjjZOq1"; ## SECRET-DATA
```

Because you cannot retrieve the passwords by looking at the configuration file, you should keep the configured passwords in a secure location. If you do forget the password and cannot login, you can always perform the password recovery process, which we cover later.

Time Parameters



The graphic shows how to use the CLI to configure the time settings. You can configure the current date and time information along with the proper time zone for the device. The default time zone on Junos devices is UTC (Coordinated Universal Time, formerly known as Greenwich Mean Time, or GMT). When you define the local time on a Junos device, you must account for the time difference between the defined time zone and the default time zone. Once the time zone is changed and committed, the local time is adjusted accordingly to account for the difference. If you do not want to make the necessary adjustments, you can simply set the system's time after the defined time zone parameter has been committed.

Instead of setting the local time on each network device in your network, you might consider implementing the Network Time Protocol (NTP). We cover NTP in detail in "Secondary System Configuration."

Management Access Parameters

The graphic also shows how to use the CLI to enable SSH and Telnet access to a device running the Junos OS. Although not shown on the graphic, you could also enable the HTTP service, which allows the device to be accessed and managed through a Web browser. When connecting to a device running the Junos OS using one of these access protocols, use the same user logins defined under the `[edit system login]` hierarchy.

In operational mode, you can control the Junos CLI environment. By default, an individual CLI session never times out after extended times, unless the `idle-timeout` statement has been included in the user's login class configuration. The timeout can be 0–100,000 minutes. Setting the timeout to **0** disables the timeout.

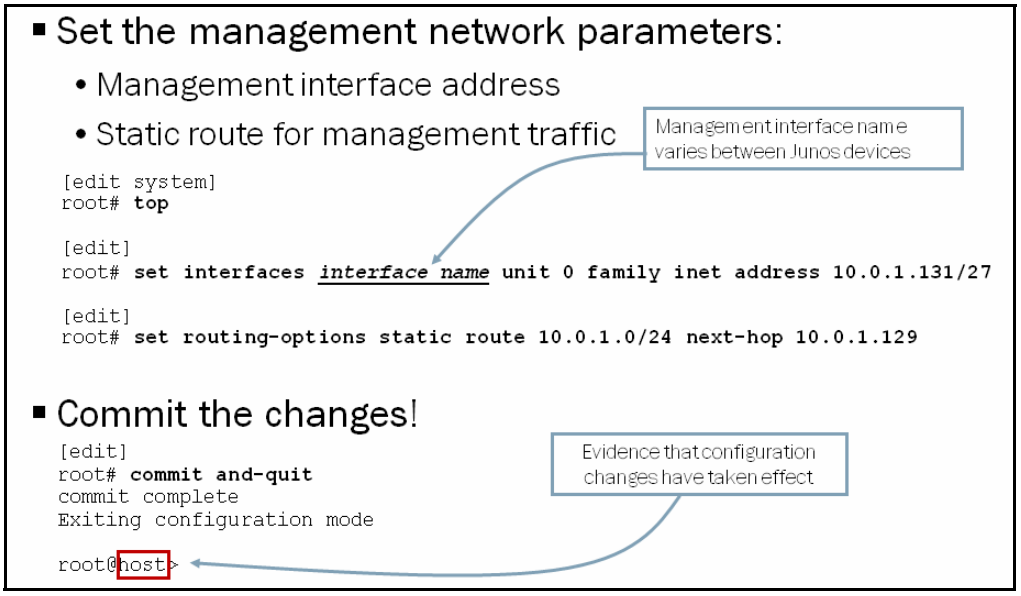
```
user@host> set cli idle-timeout 60
Idle timeout set to 60 minutes

user@host> set cli idle-timeout 0
Idle timeout disabled
```

The CLI provides a method to display a login message to users. The login message is displayed when a user connects to the host using Telnet or SSH.

```
[edit system]
user@host# set login message "Insert login message here..."
```

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


Insert login message here...

host (ttyp2)

login:
login: user
Password:********

--- JUNOS 10.1R1.8 built 2010-02-12 18:31:54 UTC
user@host>
```

Management Network Parameters



The graphic shows how to use the CLI to configure a management interface and a static route for management traffic. Note that we highly discourage using a default static route for management traffic! You should be as specific as possible. You can also use the **no-readvertise** option for the static route used for management traffic. The **no-readvertise** option marks the route ineligible for readvertisement through routing policy.

Note that the static route defined for management traffic (or any other traffic) is only available when the system's routing protocol process (rpd) is running. When Junos devices boot, the routing protocol process is not running; therefore, the system has no static or default routes. To allow the device to boot and to ensure that it is reachable over the network if the routing protocol process fails to start properly, you configure a backup router, which is a router or gateway device that is directly connected to the local system (that is, on the same subnet). To configure a backup router running IPv4, include the `backup-router` statement at the `[edit system]` hierarchy level:

```
[edit system]
root# show backup-router
10.0.1.129 destination 10.0.1.0/24;
```

In this sample configuration, hosts on the 10.0.1.0/24 subnet are reachable through the backup router. If the destination statement is omitted, then all hosts are reachable through the backup router.

To eliminate the risk of installing a default route in the forwarding table, you should always include the destination option, specifying an address that is reachable through the backup router. Specify the address in the format network/mask-length, as shown in the previous example, so that the entire network is reachable through the backup router.

When the routing protocols start, the address of the backup router is removed from the local routing and forwarding tables. To have the address remain in these tables, configure a static route for the desired destination prefix with the backup router as the next hop and the retain option as shown in the following capture:

```
[edit routing-options]
root# show
static {
    route 10.0.1.0/24 {
        next-hop 10.0.1.129;
        retain;
        no-readvertise;
    }
}
```

Activating Your Configuration

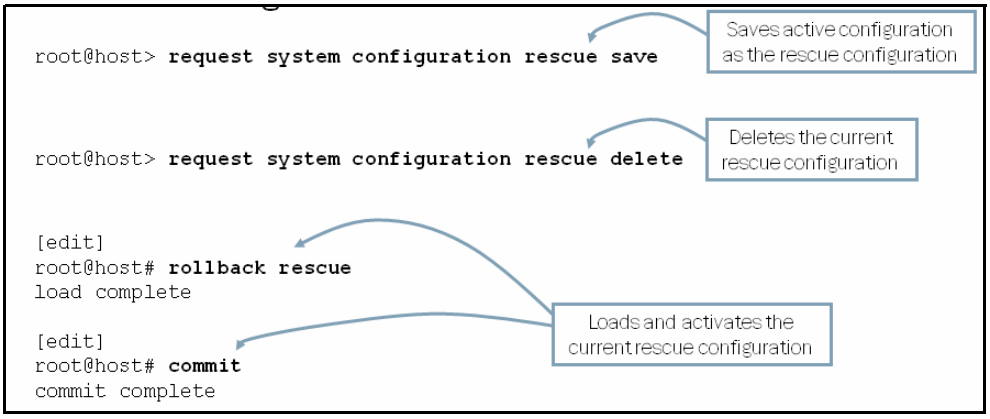
Once you complete your initial configuration, use the **commit** command to apply your changes. You can include the **and-quit** option, as shown, to return to operational mode. In the example on the graphic, we see that once the configuration changes are activated and the user returns to operational mode, the configured hostname is displayed. This displayed hostname is a sure sign that the active configuration has changed.

Viewing the Resulting Configuration

```
root@host> show configuration
## Last commit: 2010-05-11 21:00:46 UTC by root
version 10.1R1.8;
system {
    host-name host;
    time-zone America/Los_Angeles;
    root-authentication {
        encrypted-password "$1$e/FUEOVo$JF6NiAZxuufGFxDs1OMAr/"; ##
SECRET-DATA
    }
    services {
        ssh;
        telnet;
    }
    syslog {
...
```

As the graphic directs, use the operational-mode **show configuration** command to display the hierarchical configuration file as created by the initial configuration **set** statements. The complete configuration is not shown for the sake of brevity.

To the Rescue



A rescue configuration is a user-defined, known-good configuration that is 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. By default, no rescue configuration is defined.

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. Because the rollback operation only replaces the contents of the candidate configuration, you must issue **commit** to activate the configuration.

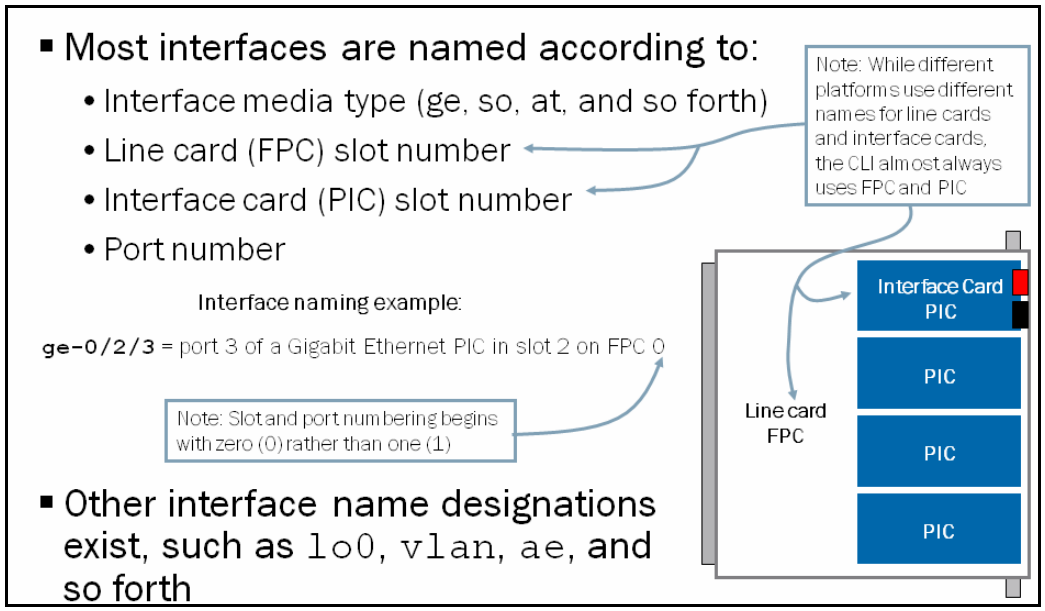
Interface Overview

Interfaces are primarily used to connect a device to a network; however, some interfaces are used to provide a service or a specific function for the system on which it operates. On platforms running the Junos OS, several types of interfaces exist, including:

- *Management interfaces*: Used to connect the device running the Junos OS to a management network. The actual designation for this interface is platform-specific; examples include `fxp0` and `me0`.
- *Internal interfaces*: Used to connect the control and forwarding planes. The actual designation for this interface is platform-specific; examples include `fxp1` and `em0`.
- *Network interfaces*: Used to provide media-specific network connectivity. Some media examples include Ethernet, SONET, Asynchronous Transfer Mode (ATM), T1, and DS3. We cover examples of network interfaces on subsequent sections.
- *Services interfaces*: Used to provide one or more user-configurable services such as encryption, tunneling, and link services. Services interfaces can be provided through a physical services interface card or through software. Services interfaces provided through a PIC do not have ports or media associated with them, but have two-letter interface type designations as shown in the list that follows (actual coverage of the services provided by these interfaces is beyond the scope of this material):
  - `es`: Encryption interface;
  - `gr`: Generic route encapsulation tunnel interface;
  - `ip`: IP-over-IP encapsulation tunnel interface;
  - `ls`: Link services interface;
  - `ml`: Multilink interface;

  - mo: Passive monitoring interface;
  - mt: Multicast tunnel interface;
  - sp: Adaptive services interface; and
  - vt: Virtual loopback tunnel interface.
- *Loopback interfaces*: Used to provide a constant and dependable hardware-independent interface. The loopback interface uses the lo0 designation on all platforms running the Junos OS. Use the lo0 interface in conjunction with routing protocols to facilitate routing in a redundant environment that is independent of the individual physical links within that environment. You can configure a single logical unit for the lo0 interface for each routing instance. Each logical unit associated with a given routing instance can, however, have multiple configured IP addresses.

Interface Naming



The Junos OS uses a standard naming convention. Most interfaces have names based on the interface media type, the system slot number in which the line card is installed, the line card slot number in which the interface card is installed, and the port number for the interface card. The CLI almost always refers to line cards as Flexible PIC Concentrators (FPCs) and interface cards as PICs even though the actual names of these physical components might vary between Junos devices. For platform-specific information, including details pertaining to the interface naming convention for your specific device, see http://www.juniper.net/techpubs/ for the technical publications.

In typical deployments, the slot and port numbering begins with zero (0) and increments based on the system hardware configuration. The graphic shows a sample interface name that illustrates the interface naming format. The highlighted interface name is for the fourth physical port (number 3) on a Gigabit Ethernet interface card installed in the third slot (number 2) of a line card that resides on the first available line card slot (number 0) of a chassis.

Other Interface Name Designations

Other interface name designations exist that do not adhere to the naming convention illustrated on the top of the graphic. Interfaces with specific designations are created by the Junos OS and are not directly associated with or dependant on physical interfaces. The following are some examples:

- *lo0*: Loopback interface;
- *ae*: Aggregated Ethernet interface;

- *as*: Aggregated SONET interface; and
- *vlan*: VLAN interface.

The Junos OS also creates a number of internal interfaces. These internally generated interfaces are nonconfigurable. The following are some examples:

- *gre*;
- *mtun*;
- *ipip*; and
- *tap*.

Note that interface support varies between the different Junos devices. For support information, always refer to the technical documentation for your specific product.

Logical Interfaces

Each physical interface descriptor can contain one or more logical interface descriptors. These descriptors allow you to map one or more logical (sometimes called virtual) interfaces to a single physical device. Creating multiple logical interfaces is useful in environments where multiple virtual circuits or Data Link Layer connections are associated with a single physical interface, such as in ATM and Frame Relay networks.

Logical Units and Encapsulation

Some encapsulations, such as the Point-to-Point Protocol (PPP) and the Cisco High-Level Data Link Control (Cisco HDLC) protocol, support only a single logical interface, and its logical unit number must be zero. Other encapsulations, such as Frame Relay, ATM, and tagged Ethernet, support multiple logical interfaces, so you can configure one or more logical unit numbers.

Circuit Identifier Versus Unit Number

The unit number and the circuit identifier are different in meaning. The circuit identifier identifies the logical tunnel or circuit, whereas 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.

Multiple Addresses

Junos devices 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. The following is an example:

```
[edit interfaces ge-0/0/1 unit 0]
user@host# set family inet address 10.1.1.1

[edit interfaces ge-0/0/1 unit 0]
user@host# show
family inet {
    address 10.1.1.1/32;
}

[edit interfaces ge-0/0/1 unit 0]
user@host# rename family inet address 10.1.1.1/32 to address 10.1.1.1/24

[edit interfaces ge-0/0/1 unit 0]
user@host# show
```

```
family inet {
    address 10.1.1.1/24;
}
```

Also note that the Junos OS forms interior gateway protocol (IGP) adjacencies over all subnets when the IGP is configured on a logical interface; this behavior is worth noting because some vendors form an adjacency over only the primary address of an interface.

Physical Properties

The following list provides details for some physical interface properties.

- *Data Link Layer protocol* and *keepalives*: You can change the Data Link Layer protocol for the particular media type (for example, PPP to Cisco HDLC), and you can turn keepalives on or off.
- *Link mode*: On Ethernet interfaces you can hardcode the duplex setting to either half-duplex or full-duplex.
- *Speed*: You can specify the link speed on certain interface types.
- *Maximum transmission unit (MTU)*: You can vary the size from 256 to 9192 bytes.
- *Clocking*: Refers to the interface clock source, either internal or external.
- *Scrambling*: Refers to payload scrambling, which can be on or off.
- *Frame check sequence (FCS)*: You can modify to 32-bit mode (the default is 16-bit mode).
- *Diagnostic characteristics*: You can enable local or remote loopbacks or set up a BERT test.

Logical Properties

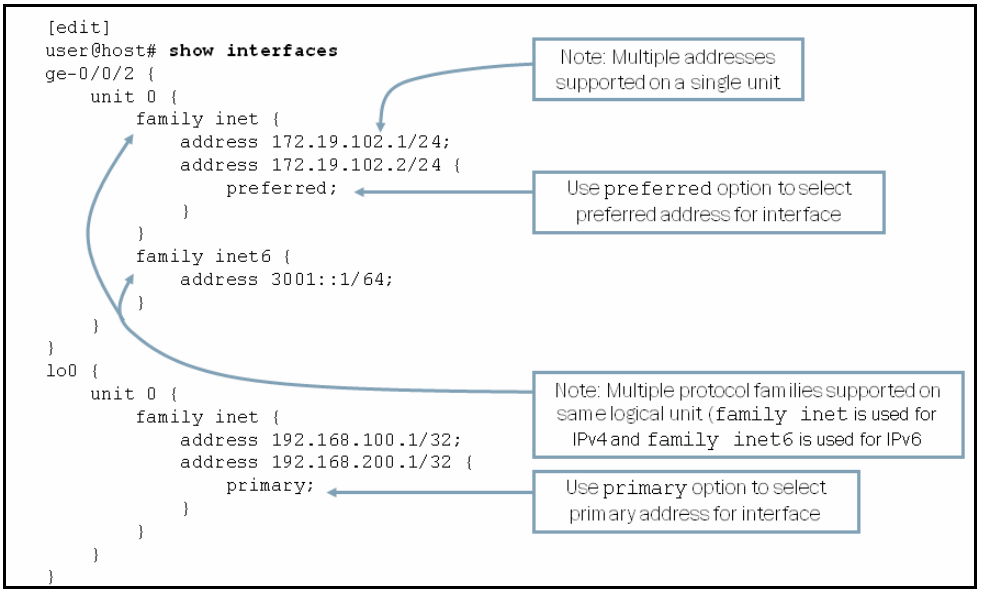
The following list provides details for some logical interface properties:

- *Protocol family*: Refers to the protocol family you want to use, such as family inet, inet6, iso, mpls, or ethernet-switching.
- *Addresses*: Refers to the address associated with the particular family (for example, IP address using family inet).
- *Virtual circuits*: Refers to the virtual circuit identifier, such as a data-link connection identifier (DLCI), virtual path identifier (VPI), virtual channel identifier (VCI), or virtual LAN (VLAN) tag.
- *Other characteristics*: Some other configurable options include Inverse ARP, traps, and accounting profiles.

Configuration Hierarchy

All interfaces have the same configuration hierarchy organization. The Junos OS considers all properties defined directly under the interface name to be the physical properties of that interface. The unit number represents a particular logical interface or subinterface. The Junos OS considers all properties defined directly under the unit number to be the logical properties of each particular subinterface.

Configuration Example



The graphic provides a basic configuration example that includes multiple interfaces, multiple protocol families configured under a single logical unit, and multiple IP addresses configured for a single protocol family.

While a single logical unit does support multiple protocol families, such as `inet` and `inet6`, you cannot configure a second protocol family in conjunction with the `ethernet-switching` protocol family. The following example illustrates this point:

```
[edit]
user@host# commit
[edit interfaces ge-0/0/2 unit 0]
  'family'
    When ethernet-switching family is configured on an interface, no other
family type can be configured on the same interface.
error: configuration check-out failed
```

The example on the graphic also highlights the use of the **`preferred`** and **`primary`** configuration options. The **`preferred`** option is used when you have multiple IP addresses belonging to the same subnet on the same interface. This option allows you to select which address will be used as the source address for packets sent by the local system to hosts on the directly connected subnet. By default, the numerically lowest local address is chosen. In the example on the graphic, the default behavior has been overridden with the **`preferred`** option making 172.19.102.2/24 the preferred address.

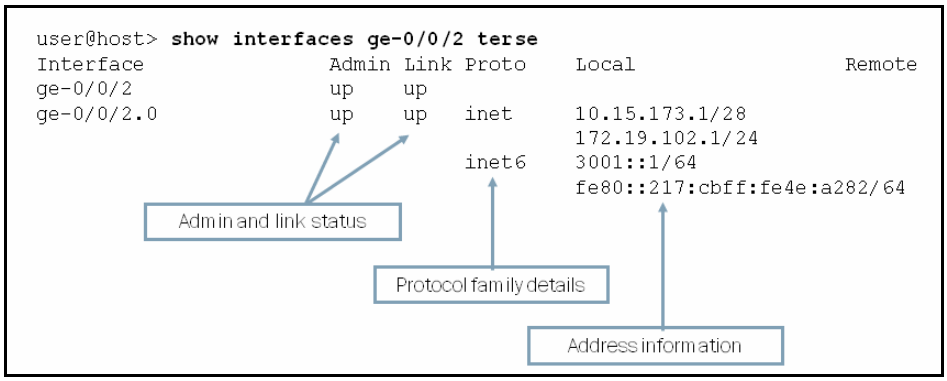
The primary address on an interface is the address that is used by default as the local address for broadcast and multicast packets sourced locally and sent out the interface. The primary address flag also can be useful for selecting the local address used for packets sent out unnumbered interfaces when multiple non-127 addresses are configured on the loopback interface, lo0. By default, the primary address on an interface is selected as the numerically lowest local address configured on the interface. In the example on the graphic, 172.19.102.1/24 is the primary address for the ge-0/0/2.0 interface, because it is the numerically lowest address configured on that interface; 192.168.200.1/32 is the primary address for the lo0.0 interfaces, because it has the **`primary`** option. The following capture verifies the primary state:

```
user@host> show interfaces ge-0/0/2.0 | find addresses
      Addresses, Flags: Is-Primary
        Destination: 172.19.102/24, Local: 172.19.102.1,
        Broadcast: 172.19.102.255
      Addresses, Flags: Preferred Is-Preferred
        Destination: 172.19.102/24, Local: 172.19.102.2,
```

```
          Broadcast: 172.19.102.255
      Protocol inet6, MTU: 1500
        Flags: Is-Primary
        Addresses, Flags: Is-Default Is-Preferred Is-Primary
          Destination: 3001::/64, Local: 3001::1
        Addresses, Flags: Is-Preferred
          Destination: fe80::/64, Local: fe80::217:cbff:fe4e:ab02

user@host> show interfaces lo0.0 | find addresses
      Addresses
        Local: 192.168.100.1
      Addresses, Flags: Primary Is-Default Is-Primary
        Local: 192.168.200.1
```

Interface support varies between Junos devices. Refer to the technical publications for detailed information for your specific product.

Tracking the State of an Interface



To quickly verify the state of an interface you can issue the **show interfaces terse** command. To filter the displayed output to an individual interface, add the name of the interface, as shown on the graphic. In the sample output, we see the admin and link status, the protocol family details, and the address information for the specified interface. We cover interface monitoring in greater detail in subsequent material.

Review Questions

1. Which command do you use at the shell prompt to enter operational mode?
2. Which configuration parameter is required during an initial configuration?
3. Which final configuration mode command must you enter to enable your initial configuration?
4. Which parameters might be configured under the logical unit hierarchy level for an interface?

Answers

1.

Use the **cli** command at the shell prompt to enter operational mode.

2.

The root authentication is the only required parameter during the initial configuration.

3.

As always, you must issue a **commit** for any configuration changes to take effect.

4.

Some examples of logical interface properties you might configure include the protocol family (such as `inet`, `inet6`, `iso`, `mpls`, or `ethernet-switching`), addresses, and virtual circuit identifiers (such as VPI, VCI, DLCI, and VLAN tag).

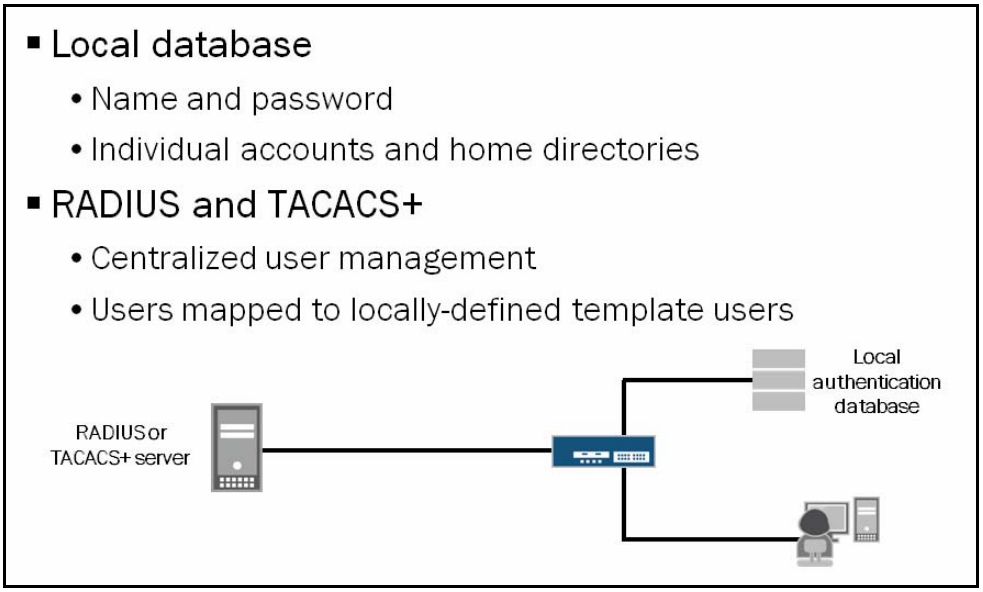
JNCIA-Junos Study Guide—Part 1

Chapter 4: Secondary System Configuration

This Chapter Discusses:

- User authentication methods and configuration;
- Configuration and analysis of system logging and tracing;
- Network Time Protocol (NTP) configuration and operation;
- Archiving of configurations on remote devices; and
- Configuration and monitoring of SNMP.

Local Password Authentication



With local password authentication, you can configure usernames and passwords individually for each user to log in to a device running the Junos operating system. The Junos OS 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
- Passwords must contain at least one change of case or character class.

When a user is configured on a device running the Junos OS, the system automatically generates a home directory for that user. The home directory serves as the default working directory for each locally configured user. The user's working directory can be changed for individual sessions using the operational mode **set cli directory *directory*** command.

RADIUS and TACACS+

RADIUS and TACACS+ are distributed client and server systems used as authentication methods to validate users. The RADIUS and TACACS+ clients run on devices running the Junos OS; the server runs on a host connected to a remote network. A locally defined user account determines authorization. Multiple RADIUS or TACACS+ authenticated users can be mapped to a locally defined template user account. Local template user accounts avoid the need for each RADIUS or TACACS+ user to also have a locally defined user account. With the appropriate Juniper Networks extensions loaded on the server, both RADIUS and TACACS+ can override these template user authorization parameters by passing extended regular expressions. Coverage of regular expressions is outside of the scope of this material.

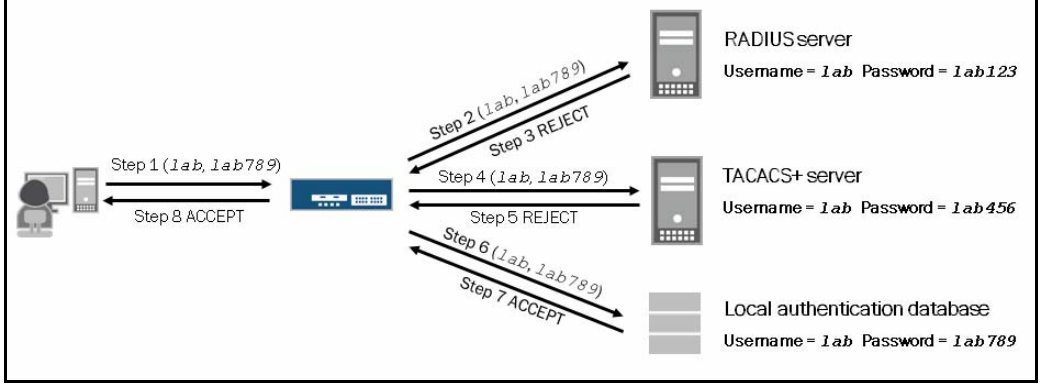
Authentication Order

You can configure devices running the Junos OS to be both a RADIUS and TACACS+ client, and you can prioritize the order in which the software tries one or more of the three different authentication methods.

For each login attempt, the Junos OS tries the authentication methods in order, until the password is accepted. The next method in the authentication order is consulted if the previous authentication method failed to reply or if the method rejected the login attempt. If no reply (accept or reject) is received from any of the listed authentication methods, the Junos OS consults local authentication as a last resort.

Example 1

```
[edit]
user@host# show system authentication-order
authentication-order [ radius tacplus password ];
```



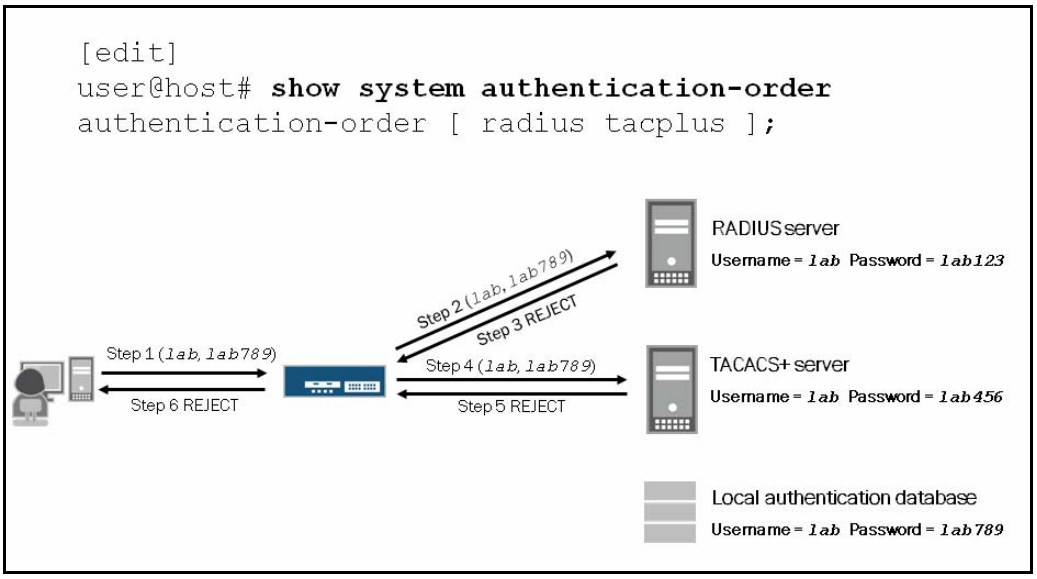
In the example shown on the graphic, we configured `authentication-order [ radius tacplus password ]`. We entered a username of *lab* and a password of *lab789*. We successfully authenticated because each configured authentication method is attempted until the password is accepted by the local authentication database.

In addition to the authentication order shown on the graphic, you would also need to configure the RADIUS and TACACS+ servers as well as the lab user. The following is a sample of these configuration parameters:

```
[edit system]
user@host# show radius-server
172.18.102.13 secret "$9$9ZKntpBvMX7Nb1RcleW-dbs2gaU"; ## SECRET-DATA
```

```
[edit system]
user@host# show tacplus-server
172.17.32.14 secret "$9$m5T31Icyrvn/A0ORlevWLXNb"; ## SECRET-DATA

[edit system]
user@host# show login user lab
class super-user;
authentication {
    encrypted-password "$1$dJ3NA9BW$nZGLZAp9kpiG52kru34IT."; ## SECRET-DATA
}
```

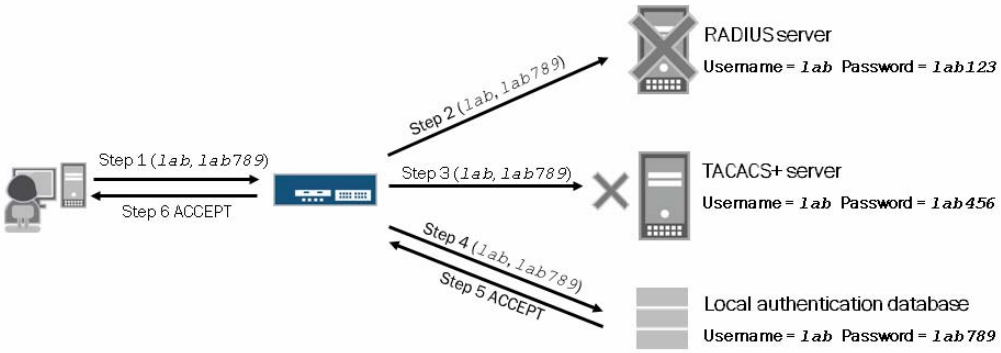
Example 2



In this example, we configured `authentication-order [ radius tacplus ]`. We entered a username of *lab* and a password of *lab789*. The Junos OS attempted to authenticate the password against the RADIUS server, which rejected it. It then attempted to authenticate the password against the TACACS+ server, which also rejected it. The Junos OS does not consult local authentication because it is not listed in the authentication order, and because at least one of the configured authentication methods did respond. The password was rejected.

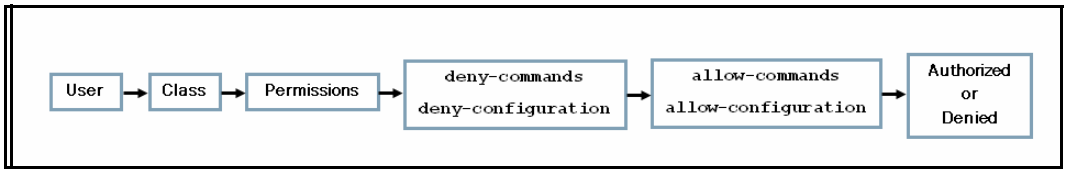
Example 3

```
[edit]
user@host# show system authentication-order
authentication-order [ radius tacplus ];
```



In this example, `authentication-order [ radius tacplus ]` is still configured. We entered a username of *lab* and a password of *lab789*. The Junos OS attempted to authenticate the password against the RADIUS server, which is down. The device running the Junos OS received no response, and after a timeout period, tried the TACACS+ server. A temporary network problem caused the TACACS+ server to be unreachable. After a timeout period, local authentication was consulted and the password was accepted. The Junos OS consulted local authentication because none of the configured authentication methods responded.

Authorization Overview



Each command or configuration statement is subject to authorization. The Junos OS applies authorization to all nonroot users, and you cannot disable this feature. Authorization applies to both the J-Web interface and the command-line interface (CLI). A configured hierarchy of authorization components, as shown by the graphic, defines whether a command is authorized.

Users

At the highest level, the configuration of user accounts define authorization parameters. As previously mentioned, multiple remotely authenticated users can be mapped to a locally defined template user. Users are members of a single login class.

Class

A login class is a named container that groups together a set of one or more permission flags. Login classes can also specify that the permission flags should be overridden for certain commands. You can configure custom login classes, but there are four predefined login classes that exist to handle most situations. These classes and associated permission flags are the following:

- `super-user`: All permissions;
- `operator`: Clear, network, reset, trace, and view permissions;

- `read-only`: View permissions; and
- `unauthorized`: No permissions.

Permissions

The following predefined permission flags group together the authorization of related commands:

- `access`: Allows the viewing of network access configuration;
- `access-control`: Allows the modifying of network access configuration;
- `admin`: Allows the viewing of user accounts;
- `admin-control`: Allows the modifying of user accounts;
- `all`: Enables all permission bits to be turned on;
- `clear`: Allows the clearing of learned network information;
- `configure`: Allows the entering of configuration mode;
- `control`: Allows the modifying of any configuration values (must be used in conjunction with the `configure` permission);
- `field`: Is reserved for field (debug) support;
- `firewall`: Allows the viewing of firewall configuration;
- `firewall-control`: Allows the modifying of firewall configuration;
- `floppy`: Allows the reading and writing of information to the floppy drive;
- `flow-tap`: Allows the viewing of flow-tap configuration;
- `flow-tap-control`: Allows the modifying of flow-tap configuration;
- `flow-tap-operation`: Enables the tapping of flows;
- `idp-profiler-operation`: Enables IDP profiler;
- `interface`: Allows the viewing of interface configuration;
- `interface-control`: Allows the modifying of interface configuration;
- `maintenance`: Allows system maintenance, including starting a local shell on the device and becoming the superuser in the shell, and can halt and reboot the system;
- `network`: Allows network access;
- `reset`: Allows the resetting and restarting of interfaces and processes;
- `rollback`: Allows the ability to roll back for depth greater than zero;
- `routing`: Allows the viewing of routing configuration;
- `routing-control`: Allows the modifying of routing configuration;
- `secret`: Allows the viewing of secret configuration;
- `secret-control`: Allows the modifying of secret configuration;
- `security`: Allows the viewing of security configuration;
- `security-control`: Allows the modifying of security configuration;
- `shell`: Allows the starting of a local shell;
- `snmp`: Allows the viewing of SNMP configuration;
- `snmp-control`: Allows the modifying of SNMP configuration;
- `system`: Allows the viewing of system configuration;
- `system-control`: Allows the modifying of system configuration;
- `trace`: Allows the viewing of trace file settings;
- `trace-control`: Allows the modifying of trace file settings;
- `view`: Allows the viewing of current values and statistics; and
- `view-configuration`: Allows the viewing of all configuration (not including secrets).

The configurable permissions might vary between Junos devices and software versions. Refer to the technical publications for your specific device and version of the Junos OS.

Allow and Deny Overrides

You can 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. The Junos OS applies the **deny-** statements before the corresponding **allow-** statements, resulting in the authorization of commands that match both.

Authorization Example

```
[edit system login]
root@host# show
class noc-admin {
    permissions [ clear network reset view ];
    allow-commands "(configure private)";
    deny-commands "(file)";
    allow-configuration "(interfaces) | (firewall)";
    deny-configuration "(groups)";
}
user nancy {
    uid 2002;
    class noc-admin;
    authentication {
        encrypted-password "$1$KQXKa/VQ$ijv77WXLnyf7XR/.1IbTq0"; ## SECRET-DATA
    }
}
```

The configuration example shows how the various authorization components are configured:

- User *nancy* is a member of the *noc-admin* class;
- The *noc-admin* class has the `clear`, `network`, `reset`, `and view` permissions;
- In addition, 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; and
- 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.

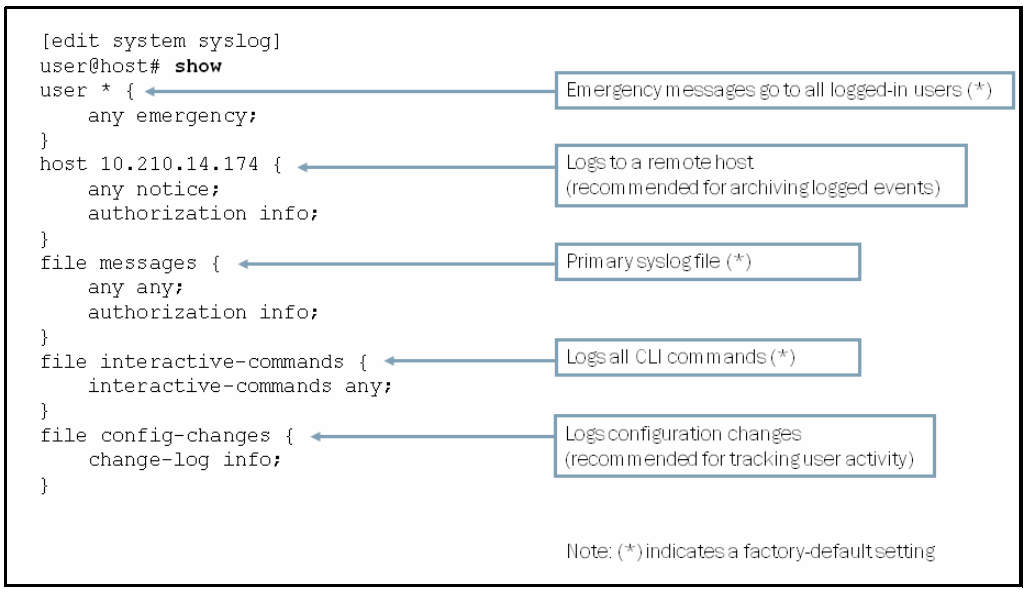
System Logging

System logging (syslog) operations use a UNIX syslog-style mechanism to record system-wide, high-level operations, such as interfaces going up or down or users logging in to or out of the device. The Junos OS places the results of the logging operations in files that are stored in the `/var/log` directory. The primary syslog file, which is included in all factory-default configurations, is the `/var/log/messages` file.

The Junos OS supports a number of facilities and severity levels. The facility is listed first and defines the class of log messages. The severity level is listed second and determines the level of detail to be logged.

Syslog information can be logged to individual files, such as the `/var/log/messages` file, or it can be sent to a remote server. Remote logging and log file archiving is recommended to aid in troubleshooting efforts.

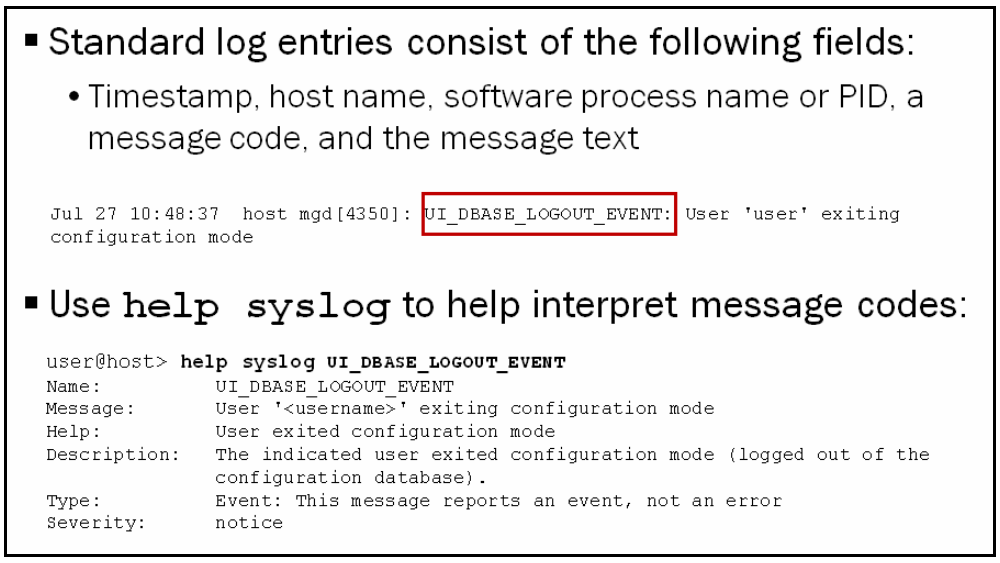
Syslog Configuration Example



The graphic shows various syslog configuration examples including a number of the default settings. Syslog operations can be enabled or modified at the `[edit system syslog]` hierarchy level and the `[edit routing-options options syslog]` hierarchy level. General syslog configuration options include the following:

- **host *name*** or ***IP address***: Sends syslog messages to a remote host—typically a UNIX device configured to receive incoming syslog messages;
- **archive**: Configures how to archive system logging files (default is to keep 10 archive files with a maximum size of 128 K each);
- **console**: Configures the types of syslog messages to log to the system console;
- **facility**: Displays the class of log messages;
- **severity**: Displays the severity level of log messages;
- **file *filename***: Configures the name of the log file; and
- **files *number***: Displays the maximum number of system log files.

Interpreting System Log Entries



When using the standard syslog format, each log entry written to the messages file consists of the following fields:

- `timestamp`: Indicates when the message was logged;
- `name`: Displays the configured system name;
- `Process name` or `PID`: Displays the name of the process (or the process ID when a name is not available) that generated the log entry;
- `message-code`: Provides a code that identifies the general nature and purpose of the message (in the example shown, the message code is `UI_DBASE_LOGOUT_EVENT`); and
- `message-text`: Provides additional information related to the message code.

When you add the **`explicit-priority`** statement, the Junos OS alters the syslog message format to include a numeric priority value. In this situation, the value 0 indicates the most significant and urgent messages (emergency), and 7 indicates debug-level messages.

Interpreting Message Codes

Consult the *System Log Messages Reference* documentation for a full description of the various message codes and their meanings, or, better yet, use the CLI's **`help`** function to obtain this information. The example shows the operator obtaining help on the meaning of the `UI_DBASE_LOGOUT_EVENT` message code. Based on the output, you can clearly see that the message code shows a command that a user entered at the CLI prompt.

Hear Tracing, Think Debug

Tracing is the Junos term for what other vendors sometimes call *debug*. In most cases, when you enable tracing (through configuration), you create a trace file that is used to store decoded protocol information received or sent by the routing engine. The Junos OS sends the tracing results to a specified file stored in the `/var/log` directory or to a remote syslog server. To enable remote logging, specify a syslog server at the `[edit system tracing]` hierarchy level as shown in the following screen capture:

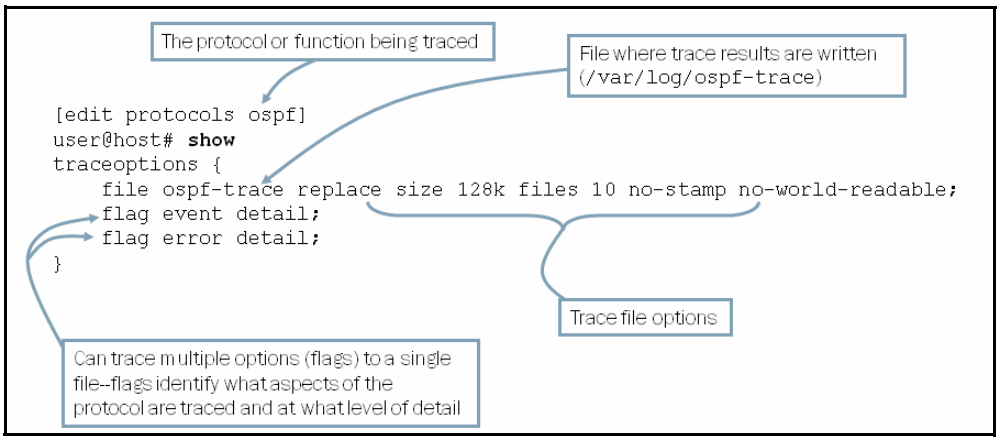
```
[edit system tracing]
user@host# show
destination-override syslog host 1.1.1.1;
```

You might see a warning when using the remote syslog server option. If the syslog server is configured properly and you have verified that the logs are being received on the server, you can safely ignore the warning. The following is a sample warning:

```
[edit]
user@host# commit
[edit protocols ospf]
  'traceoptions'
    warning: No file specified.
commit complete
```

Because of the design of the Junos OS, you can enable detailed tracing in a production network without significantly impacting performance. Even so, you should always remember to turn off tracing once you have completed your testing to avoid unnecessary resource consumption.

Traceoptions Configuration Example



Trace the operations of a specific protocol by including the **traceoptions** statement at the [edit protocols *protocol-name*] hierarchy. In most cases you should be selective in what you trace because selecting the **all** keyword will likely provide too much detail. The sample Open Shortest Path First (OSPF) Protocol stanza on the graphic reflects a typical tracing configuration that provides details about OSPF events and errors. In many cases you will want to use the **detail** switch with a given protocol flag for the added information often needed in troubleshooting scenarios.

The following are configuration options for tracing files:

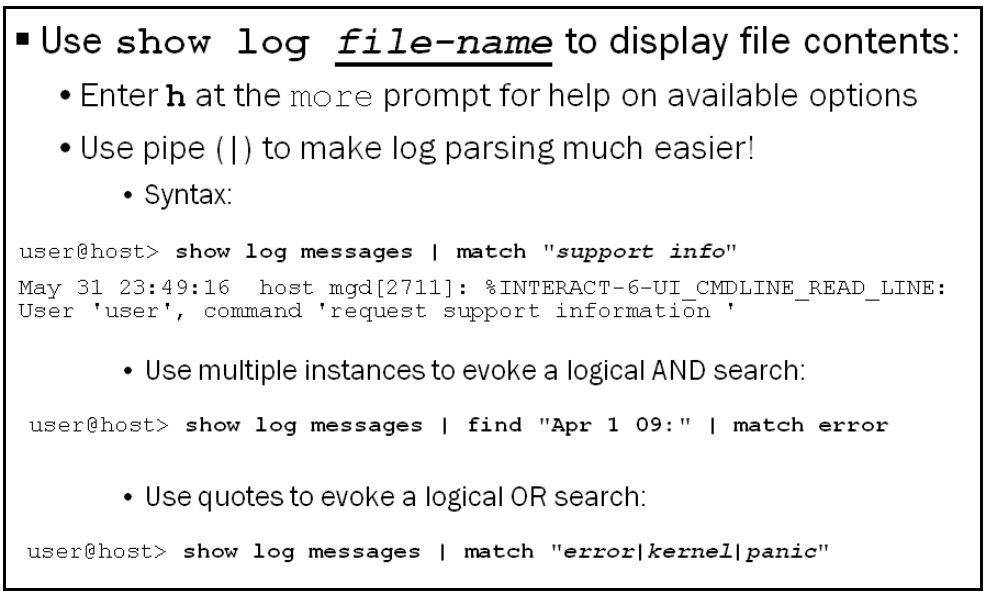
- **file *filename***: Specifies the name of the file in which to store information.
- **size *size***: Specifies the maximum size of each trace file, in kilobytes (KB), megabytes (MB), or gigabytes (GB). When a trace file named *trace-file* reaches this size, it is renamed *trace-file.0*. When the trace 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 is 128 KB.
- **files *number***: Specifies the maximum number of trace files. When a trace file named *trace-file* reaches its maximum size, it is renamed *trace-file.0*, then *trace-file.1*, and so forth, until the maximum number of trace files is reached. The software then overwrites the oldest trace file. The default is ten files.
- **no-stamp**: 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.

- **replace**: Replaces an existing trace file if one exists. By default, if you omit this option, tracing output is appended to an existing trace file.
- **readable**: Allows any user to view the file.
- **no-world-readable**: 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. You can also trace the operations of the interface process, which is the device-control process (dcd).

When tracing a specific interface, the specification of a trace file is not supported. The Junos kernel does the logging in this case, so the tracing information is placed in the system's messages file. In contrast, global interface tracing supports an archive file; by default, /var/log/dcd is used for global interface tracing.

Viewing Log and Trace Files

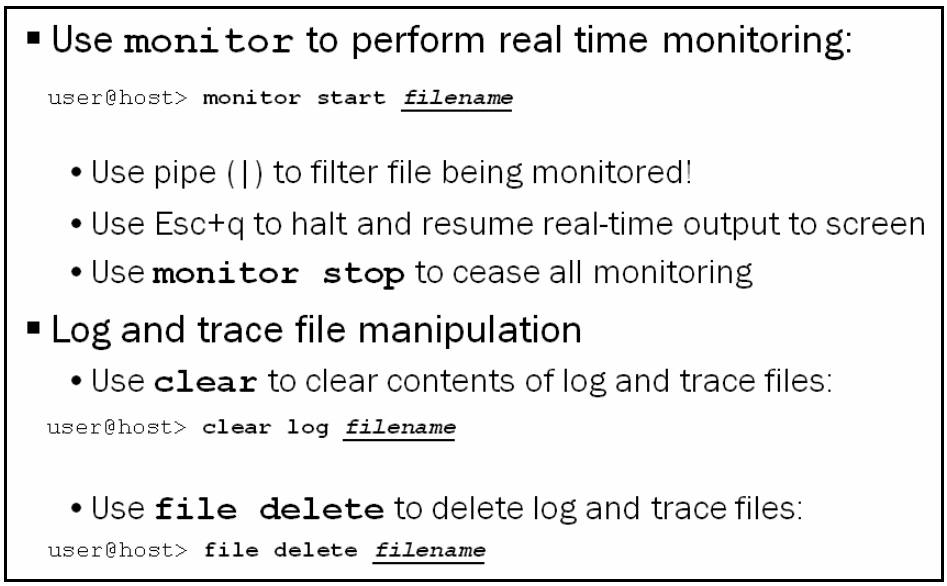


By default, the Junos OS stores log and trace files in /var/log. To view stored log files, use the **show log** command. Recall that the CLI automatically pauses when more than one screen of information exists, and that at this more prompt, you can enter a forward slash (/) character to conduct a forward search. As a hint, enter **h** at a more prompt to view the context help screen of available commands, shown in the following example:

```
---(Help for CLI automore)---
    Clear all match and except strings:                c or C
    Display all line matching a regexp:                m or M <string>
    Display all lines except those matching a regexp:  e or E <string>
    Display this help text:                            h
    Don't hold in automore at bottom of output:        N
    Hold in automore at bottom of output:              H
    Move down half display:                            TAB, d, or ^D
    Move down one line:          Enter, j, ^N, ^X, ^Z, or Down-Arrow
   . . .
```

The ability to cascade multiple instances of the CLI's pipe functionality is a real benefit when you must search a long file for specific information. The graphic shows the required syntax to evoke logical AND and logical OR searches within extensive outputs and files.

Monitoring Log and Trace Files



Use the **monitor start** CLI command to view real-time log information. You can monitor several log files at one time. The messages from each log are identified by *filename*, where *filename* is the name of the file from which entries are being displayed. The Junos OS displays this line initially and when the CLI switches between log files. To determine which log files are being monitored, you can issue the **monitor list** command.

For a user to monitor a log file using the **monitor start** command, the user must have the required access permissions to view the referenced log file. Also, because the **monitor start** command depends on the logged information being written to the log file first, the system must have the needed storage space for the log file and the log file must actually exist.

Note that you can use the CLI's **match** functionality to monitor a file in real time while displaying only entries that match your search criteria. To use this functionality, use a command in the following format:

```
user@host> monitor start messages | match fail
```

Use Esc+q to enable and disable syslog output to the screen; use the **monitor stop** command to cease all monitoring. If no output sends to the screen after issuing the **monitor start** command, you might want to issue the Esc+q key sequence to check if a previously initiated monitoring session was frozen rather than stopped.

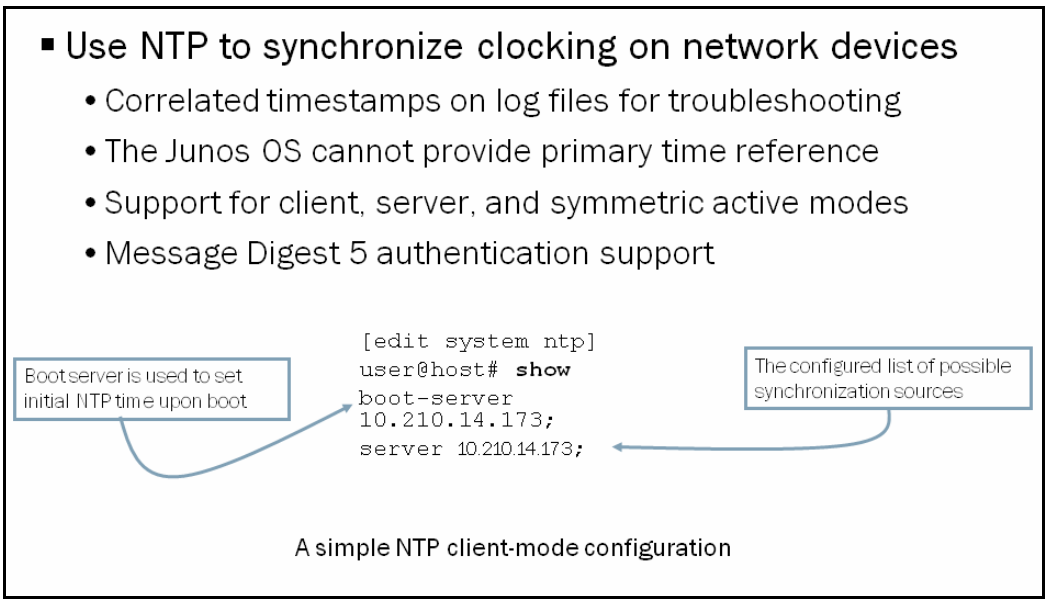
If you do not delete or disable all trace flags, tracing continues in the background and the output continues to be written to the specified file. The file remains on the storage device of the system until you either manually deleted or overwrite it according to the `traceoptions` file parameters. To disable all tracing at a particular hierarchy, issue a **delete traceoptions** command at that hierarchy and commit the change.

Log and Trace File Manipulation

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

To delete a file, use the **file delete** command. If you want, you can also use wildcards with the file command's **delete**, **compare**, **copy**, **list**, and **rename** operations.

What Time Is It?



A simple NTP client-mode configuration

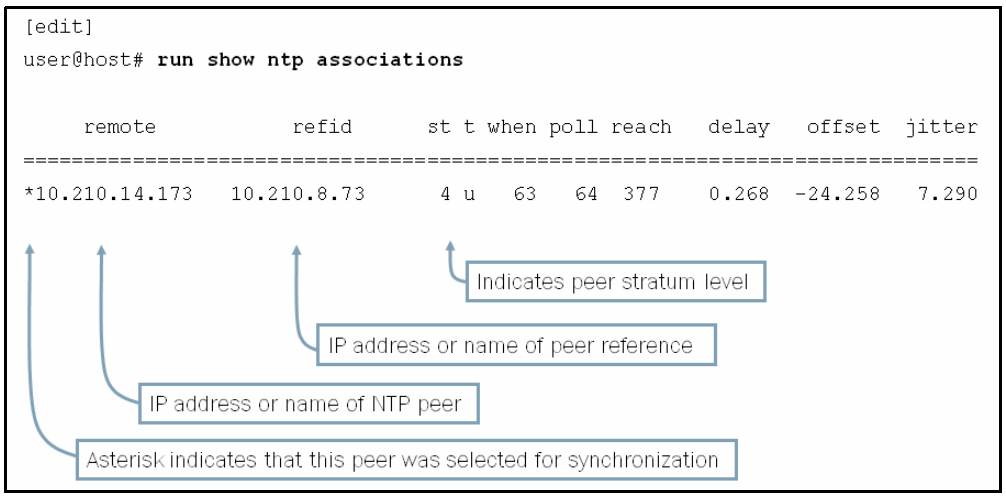
Use the Network Time Protocol (NTP) to synchronize network devices to a common, and preferably accurate, time source. By synchronizing all network devices, timestamps on log messages are both accurate and meaningful.

NTP is based on a series of timing hierarchies, with a Stratum 1 (atomic) timing source at the very top. While accuracy is desirable, there is no need to synchronize to a Stratum 1 reference to benefit from synchronizing to the time of day. The Junos OS cannot provide its own timing source because the definition of a local, undisciplined clock source (for example, the local crystal oscillator) is not supported. If needed, obtain a commodity UNIX or Windows device configured to provide a timing reference based on its local clock. Any synchronization, even if based on an inaccurate local clock, is better than none.

The Junos OS supports client, server, and symmetric modes of NTP operation, and can also support broadcast and authentication. We recommend that authentication be used to ensure that an attacker cannot compromise synchronization on a system.

The graphic provides a typical NTP-related configuration stanza. Two machines can synchronize only when their current clocks are relatively close. By default, if the time difference between the local device's clock and the NTP server's clock is more than 128 milliseconds, the clocks are slowly stepped into synchronization. However, if the difference is more than 1000 seconds, the clocks are not synchronized. A boot server is used to set a system clock at boot time to ensure that it is close enough to later synchronize to the configured time server. Issue the operational mode **`set date ntp`** ***`address`*** command as a substitute for a boot server.

Monitoring NTP

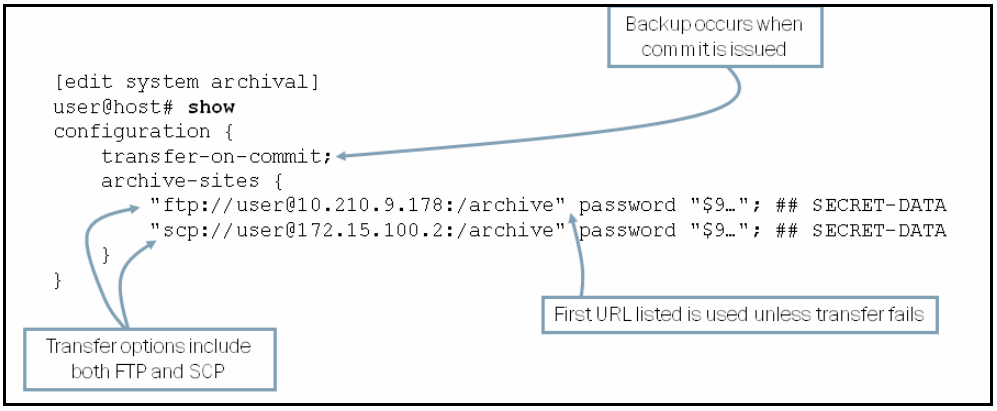


Use the **show ntp associations** command to display synchronization status. The address column shows the hostname or IP address of remote NTP peers. The symbol next to the hostname or IP address gives the status of peers in the clock selection process. The following are possible symbols:

- Space: Discarded because of a high stratum value or failed sanity check;
- x: Designated *falseticker* by the intersection algorithm;
- . (period): Culled from the end of the candidate list;
- - (hyphen): Discarded by the clustering algorithm;
- + (plus): Included in the final selection set;
- # (pound): Selected for synchronization, but the distance exceeds the maximum;
- * (asterisk): Selected for synchronization; and
- o: Selected for synchronization, but the packets-per-second (pps) signal is in use.

You can view further synchronization details with the **show ntp status** command.

Automated Configuration Backup

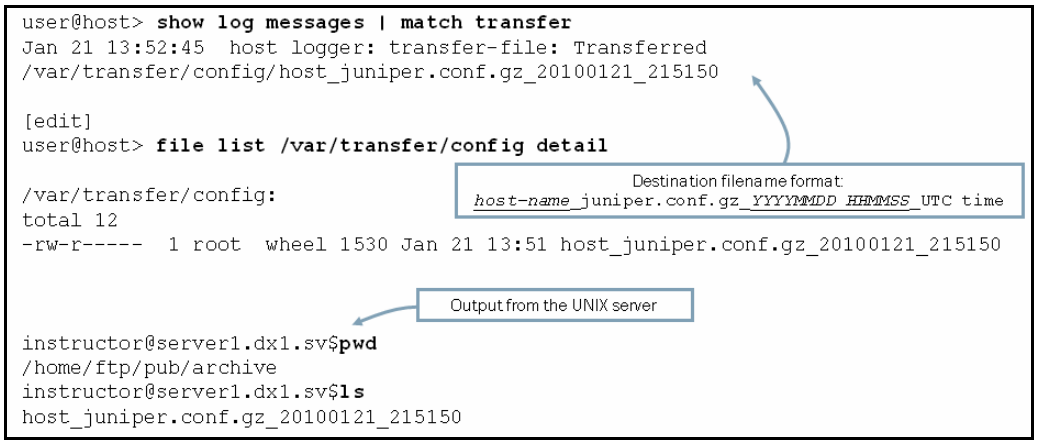


Certain failures might render the storage device, which holds the configuration files, unusable. In the event of such a disaster, it might be helpful to have the most recent configuration file stored on a separate device, such as an FTP or SCP server. To automatically back up a system's configuration file to a remote device, configure the necessary configuration archival parameters at the `[edit system archival]` hierarchy level. When you configure the system to transfer its configuration files, you specify an archive site, in the form of a URL, to which the files are

transferred. If you specify more than one archive site, the system attempts to transfer the configuration file to the first archive site in the list, moving to the next site only if the transfer fails.

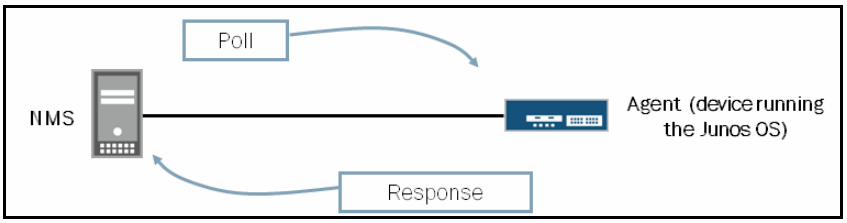
Backups occur at regular intervals with the use of the **transfer-interval** statement. The frequency at which the file transfer occurs can be from 15 to 2880 minutes, and you can define this frequency. Alternatively, the configuration file can be transferred every time a new configuration becomes active with the use of the **transfer-on-commit** statement.

How It Works



Upon entering a **commit** command or reaching the specified time interval, the system copies the configuration file into the `/var/transfer/config` directory and an FTP or SCP session is opened with the remote storage device. Once the configuration file is transferred to the remote storage device, a system log message is generated, confirming success or failure of the transfer. The destination filename format, as shown, cannot be altered by configuration.

SNMP Operation



Devices running the Junos OS act as SNMP agents. An SNMP agent exchanges network management information with SNMP manager software running on a network management system (NMS) or host. The agent responds to requests for information and actions from the manager. An agent communicates with the SNMP manager using the following message types.

- *Get, Getbulk, or Getnext requests*: The SNMP manager requests information from an SNMP agent. The agent responds with a Get response message.
- *Set requests*: The SNMP manager changes the value of a Management Information Base (MIB) object controlled by the agent. The agent returns the status in a Set response message.
- *Notifications*: The SNMP agent sends traps to notify the manager of significant events regarding the network device. SNMP version 3 uses *informs* to notify the manager of significant events. Informs increases SNMP reliability by requiring the receiver to acknowledge the receipt of an inform notification.

By polling managed network devices, the NMS collects information about network resources. An SNMP agent can also notify the NMS of events and resource constraints through the use of SNMP traps.

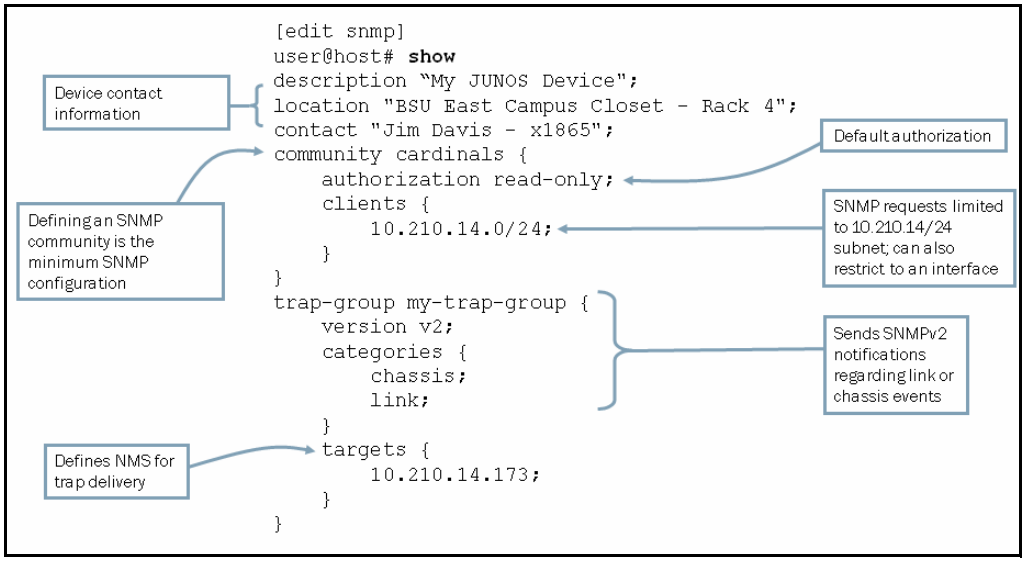
Management Information Bases

A MIB is a collection of objects maintained by the SNMP agent in a hierarchical fashion. The SNMP manager views or changes objects within the MIB structure. MIBs can be defined at the enterprise level to provide enterprise-specific information about the managed network device, or MIBs can be standardized to provide common information across multiple vendor network devices. NMS devices poll object identifiers (OIDs) to retrieve management information. An OID is considered a leaf in the tree-like hierarchy of a MIB. The Internet Engineering Task Force (IETF) provides standard MIBs you can download at http://www.ietf.org. You can download Juniper Networks enterprise MIBs at http://www.juniper.net/techpubs.

The Junos OS SNMP Support

The Junos OS provides support for SNMP versions 1, 2c, and 3. Version 1 is the initial implementation of SNMP that defines the architecture and framework for SNMP. Version 2 added support for community strings, which act as passwords determining access to SNMP agent MIBs. SNMPv3 is the most up-to-date version and provides enhanced security features including the definition of a user-based security model (USM) and a view-based access control model (VACM). SNMPv3 provides message integrity, authentication, and encryption, and is a superior security model over SNMPv2c, which uses plain text community strings. The Junos OS also provides support for remote monitoring (RMON) events, alarms, and history.

Sample SNMP Configuration



The graphic shows a sample SNMP configuration using some common SNMP configuration options. When configuring contact information, you must be as specific as possible. This information is useful when trying to resolve issues with a network device. The example restricts SNMP access to the 10.210.14.0/24 network with read-only authorization. The example also shows the configuration of an SNMP trap group, necessary for the delivery of SNMP traps to an NMS.

Monitoring SNMP Operation

```
user@host> show snmp mib walk jnxOperatingDescr
jnxOperatingDescr.1.1.0.0 = midplane
jnxOperatingDescr.2.1.1.0 = Power Supply 0
jnxOperatingDescr.2.1.2.0 = Power Supply 1
jnxOperatingDescr.4.1.1.1 = FAN 0
jnxOperatingDescr.7.1.0.0 = FPC: EX3200-24T, 8
POE @ 0/*/*
jnxOperatingDescr.8.1.1.0 = PIC: 24x
10/100/1000 Base-T @ 0/0/*
jnxOperatingDescr.8.1.2.0 = PIC: 4x GE SFP @
0/1/*
jnxOperatingDescr.9.1.0.0 = RE-EX3200-24-T
```

An NMS or host provides the interface for most SNMP monitoring. To monitor SNMP operation directly from a device running the Junos OS, you can use traceoptions, system logging, and various **show snmp** commands. When a trap condition occurs, some traps are logged if the system logging is configured with the appropriate facility and severity levels, regardless of whether a trap group is configured. The sample **show** command output on the graphic illustrates that you can also issue standard SNMP manager commands to view agent OID values. You can specify the OIDs in ASCII text format or dotted-decimal notation.

Review Questions

1. Which user authentication methods are available?
2. Which command displays the primary syslog file?
3. Why should you use configuration archival?
4. What is the purpose of SNMP traps?

Answers

1.

Users can be authenticated using the local password database, RADIUS authentication, and TACACS+ authentication.

2.

The `messages` log is the primary syslog file, and is stored in `/var/log` directory. Use the **show log messages** command to view the `messages` log.

3.

Configuration archival allows for disaster recovery in situations where a system storage device becomes unusable. Archiving configurations can also be a useful part of a company's configuration management policy.

4.

An SNMP trap is an agent-initiated notification of network events relative to the sending agent.

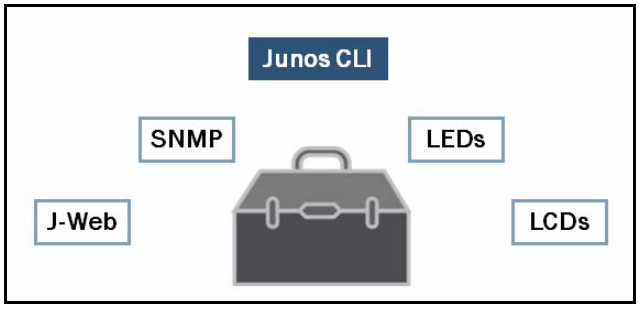
JNCIA-Junos Study Guide—Part 1

Chapter 5: Operational Monitoring and Maintenance

This Chapter Discusses:

- Monitoring of platform and interface operations;
- Network utilities and usage guidelines;
- Maintenance of the Junos operating system; and
- Password recovery.

Monitoring Tools



The primary monitoring tool for avid Junos users is the Junos command-line interface (CLI). The Junos CLI includes several **show** and **monitor** commands that facilitate system monitoring. We highlight many of the monitoring capabilities available through the Junos CLI in this material.

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. Check the technical publications at http://www.juniper.net/techpubs/ for specific details on a particular platform.

Monitoring System Operation

You can obtain most system information using **show system *argument*** commands. The following arguments are some of the most common:

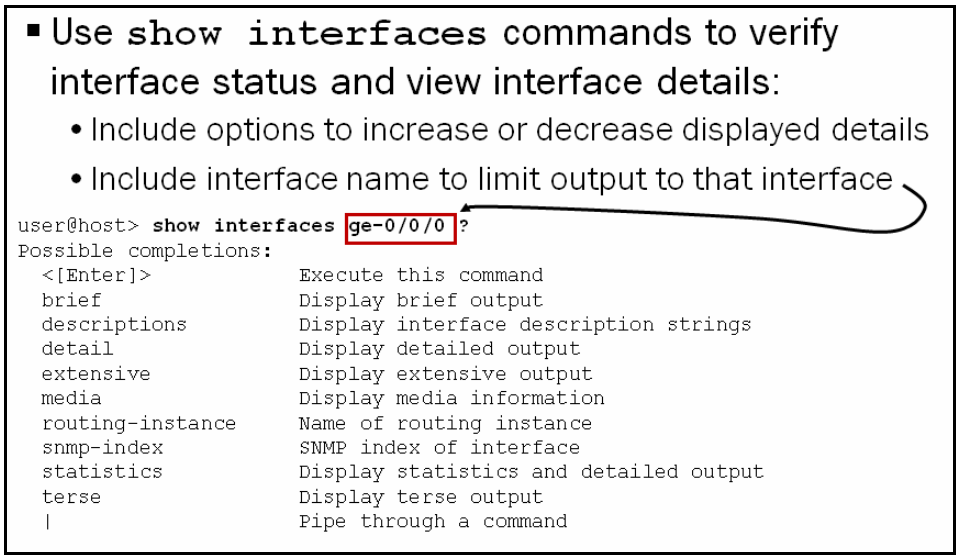
- **alarms**: This argument displays current system alarms;
- **boot-messages**: This argument displays the messages seen during the last system boot;
- **connections**: This argument displays the status of local TCP and UDP connections;
- **statistics**: This argument provides options for viewing various protocol statistics; and
- **storage**: This argument displays the status of the file system storage space.

Monitoring the Chassis

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

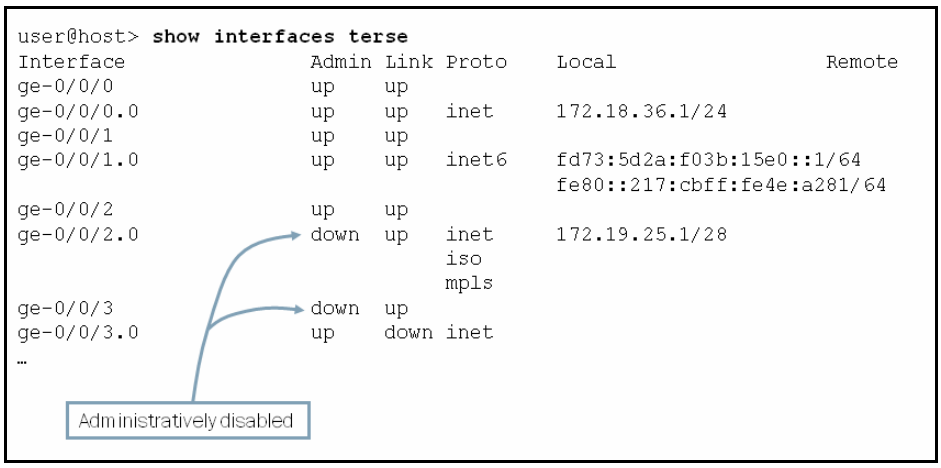
- **alarms**: This argument displays current chassis alarms;
- **environment**: This argument displays component and environmental status as well as the operational speeds of the cooling system;
- **hardware**: This argument displays an inventory of the installed hardware components along with the serial number of each component; and
- **routing-engine**: This argument provides operational status and utilization details for the Routing Engine (RE).

Interface Status Verification



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. The example on the graphic illustrates the use of 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 provides interface details for all installed interfaces.

Terse Output Example



The example on the graphic illustrates the **show interfaces terse** command. In this example the ***interface-name*** option is omitted, which causes all installed interfaces and their accompanying details to be 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.

Gathering Extensive Interface Information

```
user@host> show interfaces ge-0/0/0 extensive
Physical interface: ge-0/0/0, Enabled, Physical link is Up
  Interface index: 129, SNMP ifIndex: 32, Generation: 130
  Link-level type: Ethernet, MTU: 1514, Speed: 100mbps, Loopback: Disabled,
  Source filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
  Remote fault: Online
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 00:17:cb:4e:a2:80, Hardware address: 00:17:cb:4e:a2:80
  Last flapped   : 2010-10-03 20:46:59 UTC (8w6d 07:27 ago)
  Statistics last cleared: 2010-10-15 21:16:11 UTC (7w1d 06:58 ago)
  Traffic statistics:
…
```

Use the **show interface extensive** command to view detailed information for a named interface (or all interfaces when a specific interface is not identified). The example on the graphic shows a portion of the generated output when using the **extensive** option. This command is ideal when troubleshooting interfaces because it shows errors, statistics, and physical and logical interface properties. This command is also helpful when determining default settings for interfaces.

Monitoring an Interface

```
host                              Seconds: 23                 Time: 06:11:08
                                                             Delay: 0/0/2
Interface: ge-0/0/0.0, Enabled, Link is Up
Flags: SNMP-Traps
Encapsulation: ENET2
Local statistics:                                               Current delta
  Input bytes:                    146945                              [13768]
  Output bytes:                    33911                              [14327]
  Input packets:                    2383                                [185]
  Output packets:                    313                                 [70]
Remote statistics:
  Input bytes:                        48 (4824 bps)                      [0]
  Output bytes:                      240 (0 bps)                         [0]
  Input packets:                      11 (0 pps)                        [7]
  Output packets:                      4 (0 pps)                        [0]
Traffic statistics:
  Input bytes:                    146993  Output bytes:    ,              [0]


Next='n', Quit='q' or ESC, Freeze='f', Thaw='t', Clear='c', Interface='i'
```

The graphic depicts typical output from the **monitor interface** command. Your terminal session must support VT100 emulation for the screen to correctly display the output. 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 following is a sample of the output from this command:

```
user@host> monitor interface traffic
host                                Seconds: 27                 Time: 04:47:57

Interface     Link   Input packets         (pps)     Output packets         (pps)
 ge-0/0/0       Up           22763         (581)              21275         (581)
...

Bytes=b, Clear=c, Delta=d, Packets=p, Quit=q or ESC, Rate=r, Up=^U, Down=^D
```

Ping and Traceroute Utilities

- Use the `ping` and `traceroute` commands to test reachability and determine the forwarding path
  - Use Ctrl+c to stop the ping and traceroute operations

```
user@host> ping 10.210.14.173
PING 10.210.14.173 (10.210.14.173): 56 data bytes
64 bytes from 10.210.14.173: icmp_seq=0 ttl=64 time=0.345 ms
64 bytes from 10.210.14.173: icmp_seq=1 ttl=64 time=0.292 ms
^C
--- 10.210.14.173 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.218/0.281/0.345/0.046 ms

user@host> traceroute 10.210.14.173
traceroute to 10.210.14.173 (10.210.14.173), 30 hops max, 40 byte pkts
 1  10.210.14.173 (10.210.14.173)  2.872 ms  0.203 ms  0.150 ms
```

The Junos CLI provides ping and traceroute utilities. You can use these tools to determine general network reachability and the path that packets take to reach a destination. 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 illustrated on the graphic. Alternatively, you can include the **count** option with a specified number of ICMP echo requests to send out:

```
user@host> ping 10.210.11.177 count 5
PING 10.210.11.177 (10.210.11.177): 56 data bytes
64 bytes from 10.210.11.177: icmp_seq=0 ttl=64 time=0.071 ms
64 bytes from 10.210.11.177: icmp_seq=1 ttl=64 time=0.060 ms
64 bytes from 10.210.11.177: icmp_seq=2 ttl=64 time=0.125 ms
64 bytes from 10.210.11.177: icmp_seq=3 ttl=64 time=0.128 ms
64 bytes from 10.210.11.177: icmp_seq=4 ttl=64 time=0.080 ms

--- 10.210.11.177 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.060/0.093/0.128/0.028 ms
```

Monitoring Traffic

- Use `monitor traffic` to decode packets:
  - Captures traffic sourced from or destined to device
    - Use options to modify or filter captured traffic: the following are some common options:

```
user@host> monitor traffic ?
Possible completions:
  <[Enter]>            Execute this command
…
  detail               Display detailed output
  extensive            Display extensive output
  interface            Name of interface
  layer2-headers       Display link-level header on each dump line
  matching             Expression for headers of receive packets to match
…
```

The CLI's **`monitor traffic`** command provides access to the tcpdump utility. This tool monitors traffic that originates or terminates on the local RE. If you do not specify an interface, the management interface is monitored. This capability provides a way to monitor and diagnose problems at Layer 2 using the **`layer2-headers`** argument. You can match packet fields using the **`matching`** option and save packet captures for analysis from a third-party packet decoder such as Ethereal or Wireshark using the **`write-file`** option.

The **`write-file`** option is hidden and should be used with caution. If used improperly, this command option could fill the available storage space of the device.

Packet Capture Example

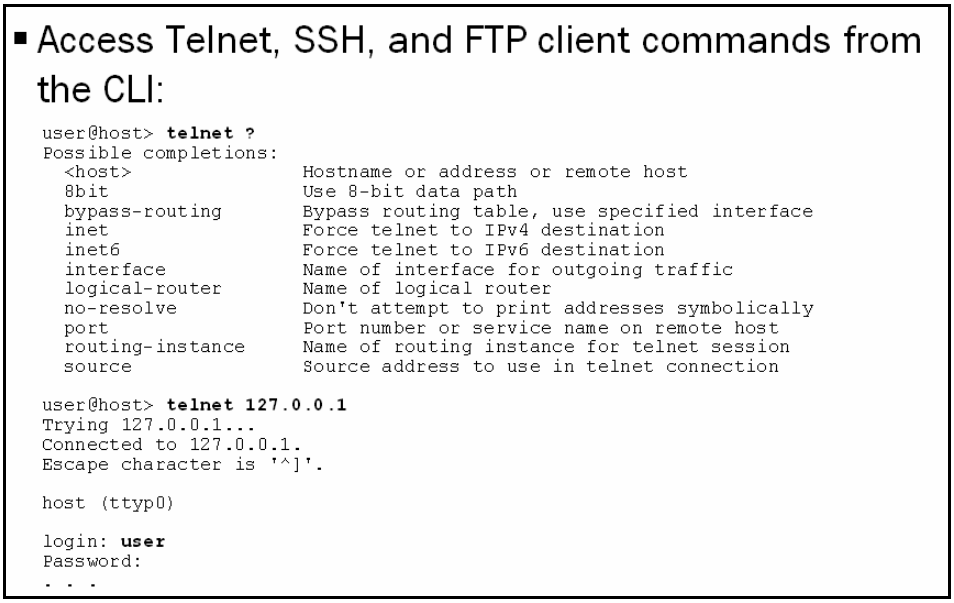
Use the `detail` or `extensive` option for complete decode

```
user@host> monitor traffic interface ge-0/0/2 layer2-headers no-resolve
verbose output suppressed, use <detail> or <extensive> for full protocol decode
Address resolution is OFF.
Listening on ge-0/0/2, capture size 96 bytes

06:19:35.121217  In 0:1b:c0:5e:53:a2 > 0:19:e2:50:3f:e3, ethertype IPv4 (0x0800),
length 98: 10.100.200.1 > 10.100.200.2: ICMP echo request, id 5153, seq 222, length 64
06:19:35.121269 Out 0:19:e2:50:3f:e3 > 0:1b:c0:5e:53:a2, ethertype IPv4 (0x0800),
length 98: 10.100.200.2 > 10.100.200.1: ICMP echo reply, id 5153, seq 222, length 64
^C
10 packets received by filter
0 packets dropped by kernel
```

Ctrl+c key sequence exits listening mode

The graphic provides an example of the CLI **monitor traffic** command. Note that to stop a packet capture, you use the Ctrl + c keyboard sequence.

Network Utilities



The CLI supports powerful Telnet, SSH, and FTP clients. These clients support various arguments that tailor their specific operations.

You use the CLI's **file copy** command to transfer files to and from devices running the Junos OS. The following example uses the **file copy** command in conjunction with the FTP client to transfer a file from a remote FTP server to the local device running the Junos OS:

```
user@host> file copy ftp://ftp:ftp@10.210.11.189/junos-jseries-domestic.tgz /
var/tmp/junos-jseries-domestic.tgz
/var/tmp//...transferring.file.........Ri4PRe/100% of   41 MB 4071 kBps 00m00s
```

Determining the Junos OS Release

```
user@host> show version
Hostname: host
Model: mx480
JUNOS Base OS boot [10.1R1.8]
JUNOS Base OS Software Suite [10.1R1.8]
JUNOS Kernel Software Suite [10.1R1.8]
JUNOS Crypto Software Suite [10.1R1.8]
JUNOS Packet Forwarding Engine Support (M/T Common) [10.1R1.8]
JUNOS Packet Forwarding Engine Support (MX Common) [10.1R1.8]
JUNOS Online Documentation [10.1R1.8]
JUNOS Routing Software Suite [10.1R1.8]
```

You use the **show version** CLI command to determine the current Junos OS Release on a device running the Junos OS. You can include the **detail** option to view additional details about the software packages and the processes included in the Release. The following are some common Junos packages and a description of each:

- *jkernel*: The kernel and network tools package. This package contains the basic operating system files.
- *jroute*: The Routing Engine package. This package contains the Routing Engine software.
- *jpfe*: The Packet Forwarding Engine (PFE) package. This package contains the PFE software.

- *jdocs*: The documentation package. This package contains the documentation set for the software.
- *jcrypto*: The encryption package. This package contains the domestic version of the security software.

The Junos Naming Convention

The Junos naming convention format is *package-release-edition*.

- *package* is a description of the software contents. Package descriptions include `jinstall`, which is used on M Series, T Series, 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. The actual package name might vary between platforms within a Junos product family. Always ensure that you download and install the proper image for your device.
- *release* describes the Junos OS Release 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. The most commonly occurring letter is R, which stands for 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). In some situations, you might see the letter S, which is reserved for service releases. The release also includes a build and spin number for the Junos OS Release. For example, `jinstall-10.1R1.8-domestic.tgz` indicates a Junos image associated with version 10.1, build 1, spin 8.
- *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 that provides advanced network security for customers who must comply with and operate in a Federal Information Processing Standards (FIPS) 140-2 environment.

All Junos OS packages contain digital signatures, the Secure Hash Algorithm 1 (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 Release.

Downloading the Junos OS

Before upgrading the Junos OS, you must download the appropriate image for your device from the Junos download site. You can download the Junos OS using a Web browser or through an FTP client (including the device running the Junos OS itself). Regardless of the download method you choose, you must have a valid service contract and access account.

To download the Junos OS through a Web browser, you point your browser to http://www.juniper.net/support/, login, select the desired image, and accept the request to begin the download process.

To access the Junos OS through an FTP client, you open an FTP session from an FTP client to the FTP server using the ftp.juniper.net Uniform Reference Identifier (URI), login, navigate to the desired directory where the Junos image is stored, and download the desired image using the appropriate FTP commands.

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

Upgrading the Junos OS

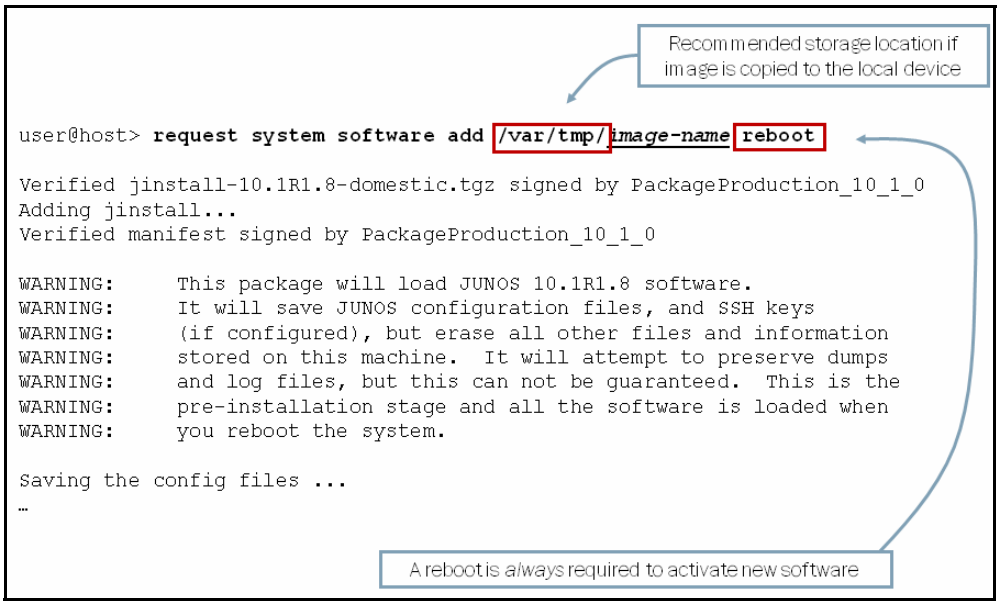
Use the **`request system software add`** ***`path/image name`*** CLI command to upgrade the Junos OS. You can specify a local path and file name or a remote FTP or SCP URI that contains the required Junos image to

download and install. To activate the new software, you must reboot the system. You can perform the system reboot as a separate step or you can initiate it by adding the **reboot** option at the end of the **request system software add** command.

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

Devices running the Junos OS execute binaries supplied only by Juniper Networks. Each Junos image includes a digitally signed manifest of executables that are registered with the system only if the signature can be validated. The Junos OS 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 device running the Junos OS.

Upgrade Example

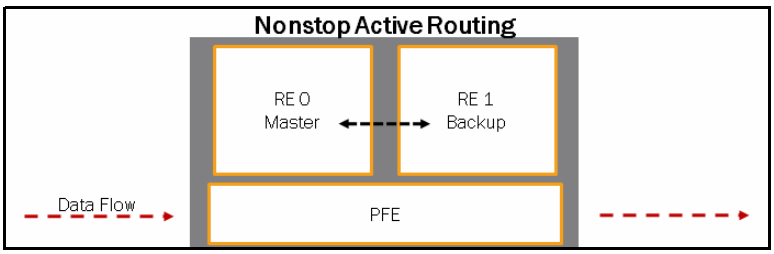


When upgrading the Junos OS, you reference the image name and a local path or a remote server within a URI. You should store the Junos OS images copied to a device running the Junos OS in preparation for an upgrade in the `/var/tmp` directory. You can delete Junos images stored in the `/var/tmp` directory when you perform the file system cleanup operation using the **request system storage cleanup** CLI command. To determine which files are cleanup candidates, you can issue the **request system storage cleanup dry-run** command.

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

As the graphic indicates, when an upgrade is performed, the system must be rebooted for the new Release to take affect. To save time and keystrokes, you can use the **reboot** option when performing the upgrade. Once the Junos OS 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, illustrated earlier in this material, to verify the Junos OS Release. You can also review the boot messages by issuing the **show system boot-messages** command.

Unified ISSU



A unified in-service software upgrade (unified ISSU) enables you to upgrade between two different Junos OS releases with no disruption on the control plane and with minimal disruption of traffic. Unified ISSU is only supported on dual Routing Engine platforms. In addition, the graceful Routing Engine switchover (GRES) and nonstop active routing (NSR) must be enabled. Note that NSR is not supported on all the Junos devices. Refer to the technical publications for platform and protocol support details.

The master RE and backup RE must be running the same software Release before you can perform a unified ISSU. You cannot take any PICs online or offline during a unified ISSU.

Perform a Unified ISSU

- To perform a unified ISSU, complete the following steps:
  1. Enable GRES and NSR, and verify that the REs and protocols are synchronized
  2. Download the new software package from the Juniper Networks Support Web site and then copy the package to the router
  3. Issue the **`request system software in-service-upgrade`** command on the master RE

The graphic lists the high-level process for performing a unified ISSU. Detailed coverage of GRES and NSR are outside the scope of this material.

Password Recovery Requires Console Connection

1. Reboot the system
   - Press the Spacebar when prompted
   - Enter **`boot -s`** to access single user mode
2. Enter **`recovery`** when prompted to go into recovery mode
3. Set root password
4. Commit the change and exit configuration mode—reboot when prompted

If you become locked out of a device running the Junos OS, you can recover the root password. As a security precaution, you can perform the recovery only by using the console connection. You can disable the password recovery option by setting the console port to the insecure mode as shown in the following screen capture:

```
[edit system ports]
user@host# show
console insecure;
```

Password Recovery Steps

The following steps list the process for recovering the root password.

1. Obtain 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:

```
...TRIMMED...
Hit [Enter] to boot immediately, or space bar for command prompt.
<user presses Spacebar>
Type '?' for a list of commands, 'help' for more detailed help.
loader> boot -s
```

2. 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.

```
...TRIMMED...
Enter full pathname of shell or 'recovery' for root password recovery or RETURN
for /bin/sh: recovery
```

3. 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. Do not forget to commit your configuration.

```
...TRIMMED...
Starting CLI ...
root> configure
Entering configuration mode

[edit]
root# set system root-authentication plain-text-password
New password:
Retype new password:

[edit]
root# commit
commit complete
```

4. To complete the recovery, exit configuration mode. You are then prompted to reboot the system. Choose `y` (`yes`) to reboot the system. Once the reboot is complete, you can log in with the new root password.

```
[edit]
root# exit
Exiting configuration mode

root> exit

Reboot the system? [y/n] y
```

Review Questions

1. List two methods for monitoring devices running the Junos OS.
2. Which command do you use to view interface usage details in real time?
3. Which command do you use to perform packet captures?
4. Describe the upgrade procedure.

Answers

1.

The primary method for monitoring devices running the Junos OS is the CLI, which includes operational **show** and **monitor** commands. Some secondary methods include J-Web, SNMP, hardware LEDs, and front-panel displays or LCDs.

2.

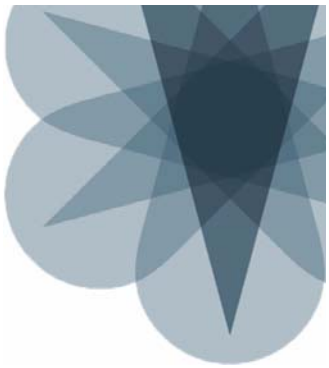
Use the CLI **monitor interface** or **monitor interface traffic** commands to view interface usage in real time.

3.

Use the CLI **monitor traffic interface** command to perform a packet capture.

4.

You must first download the Junos image for your respective platform from the Juniper Networks download site. The install package can be copied to the device running the Junos OS directly (recommended directory is /var/tmp) or you can copy the image to a server that is reachable through FTP or SCP from the device being upgraded. You then perform the upgrade using the CLI **request system software add** command. You can monitor the upgrade process through a console connection and verify the Junos OS Release using the CLI **show version** command.



JNCIA-Junos Study Guide—Part 1

Appendix A: Interface Configuration Examples

This Appendix Discusses:

- The interface configuration hierarchy;
- Configuration examples for various interface types; and
- Configuration groups.

Physical Properties

The following list provides details for some physical interface properties:

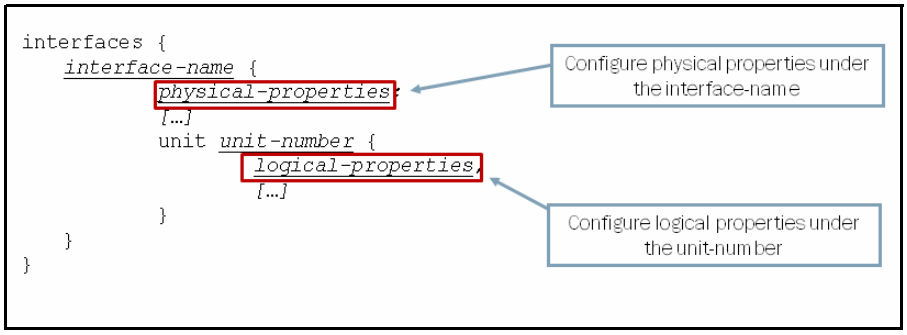
- *Data Link Layer protocol* and *keepalives*: You can change the Data Link Layer protocol for the particular media type (for example, PPP to Cisco HDLC), and you can turn keepalives on or off;
- *Link mode*: On Ethernet interfaces you can hardcode the duplex setting to either half-duplex or full-duplex;
- *Speed*: You can specify the link speed on certain interface types;
- *Maximum transmission unit (MTU)*: You can vary the size from 256 to 9192 bytes;
- *Clocking*: Refers to the interface clock source—either internal or external;
- *Scrambling*: Refers to payload scrambling, which can be on or off;
- *Frame check sequence (FCS)*: You can modify to 32-bit mode (the default is 16-bit mode); and
- *Diagnostic characteristics*: You can enable local or remote loopbacks or set up a BERT test.

Logical Properties

The following list provides details for some logical interface properties:

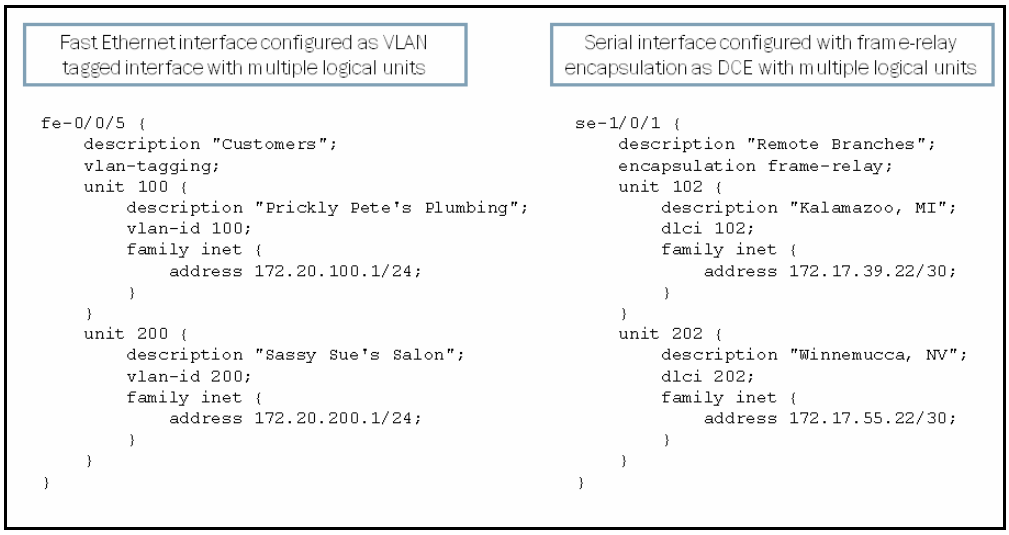
- *Protocol family*: Refers to the protocol family you want to use, such as family inet, inet6, iso, mpls, or ethernet-switching;
- *Addresses*: Refers to the address associated with the particular family (for example, IP address using family inet);
- *Virtual circuits*: Refers to the virtual circuit identifier, such as a data-link connection identifier (DLCI), virtual path identifier (VPI), virtual channel identifier (VCI), or virtual LAN (VLAN) tag; and
- *Other characteristics*: Some other configurable options include Inverse ARP, traps, and accounting profiles.

Configuration Hierarchy



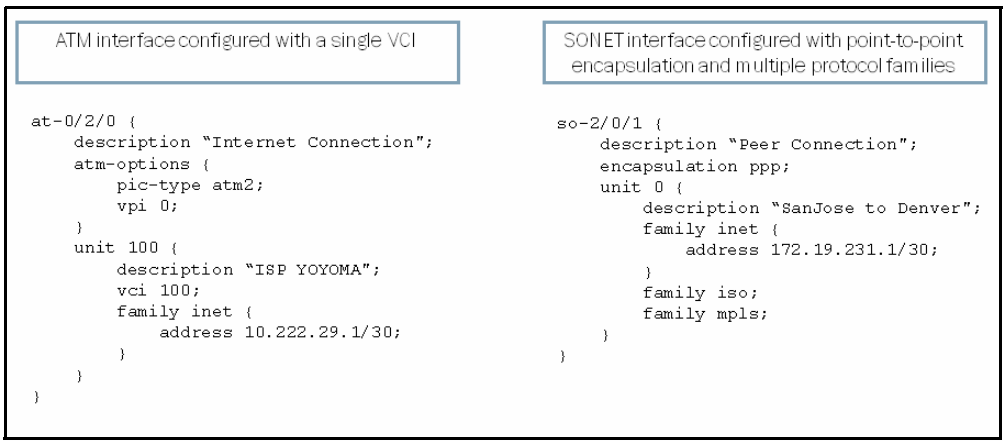
All interfaces have the same configuration hierarchy organization. The Junos operating system considers all properties defined directly under the interface name to be the physical properties of that interface. The unit number represents a particular logical interface or subinterface. The Junos OS considers all properties defined directly under the unit number to be the logical properties of each particular subinterface.

Configuration Examples: Part 1



The graphic shows two configuration examples. The first configuration example displays a tagged Ethernet interface with multiple logical interfaces; each logical unit is assigned its respective VLAN ID. The second configuration example shows a serial interface configured with the frame-relay encapsulation. Each logical interface assigned to the serial interface has a corresponding data-link connection identifier (DLCI). Both configuration examples are configured for IPv4 routing, which uses the `inet` protocol family.

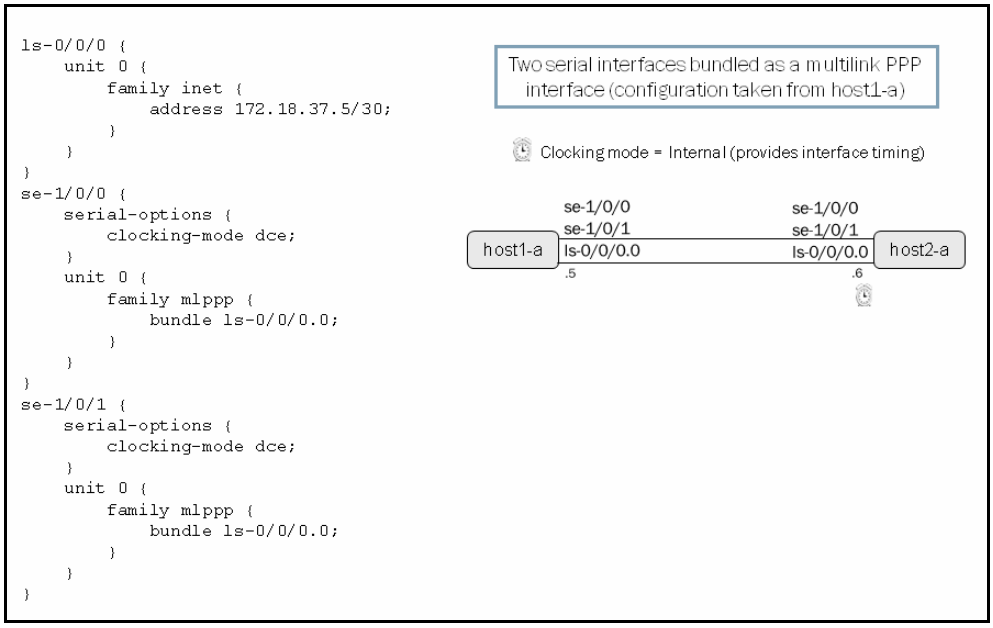
Configuration Examples: Part 2



The graphic shows two configuration examples. The first configuration example displays an Asynchronous Transfer Mode (ATM) interface with a single logical unit and corresponding VCI. Note that this ATM interface configuration example is based on the ATM2 IQ interface. A second ATM interface configuration example is shared in the Using Configuration Groups section, which is based on the ATM1 interface.

The second configuration example on the graphic shows a SONET interface configured with Point-to-Point Protocol (PPP) encapsulation and multiple protocol families. We used the `iso` protocol family for the IS-IS routing protocol, and we used the `mpls` protocol family for traffic engineering. Both configuration examples are for IPv4 routing, which uses the `inet` protocol family.

Configuration Examples: Part 3



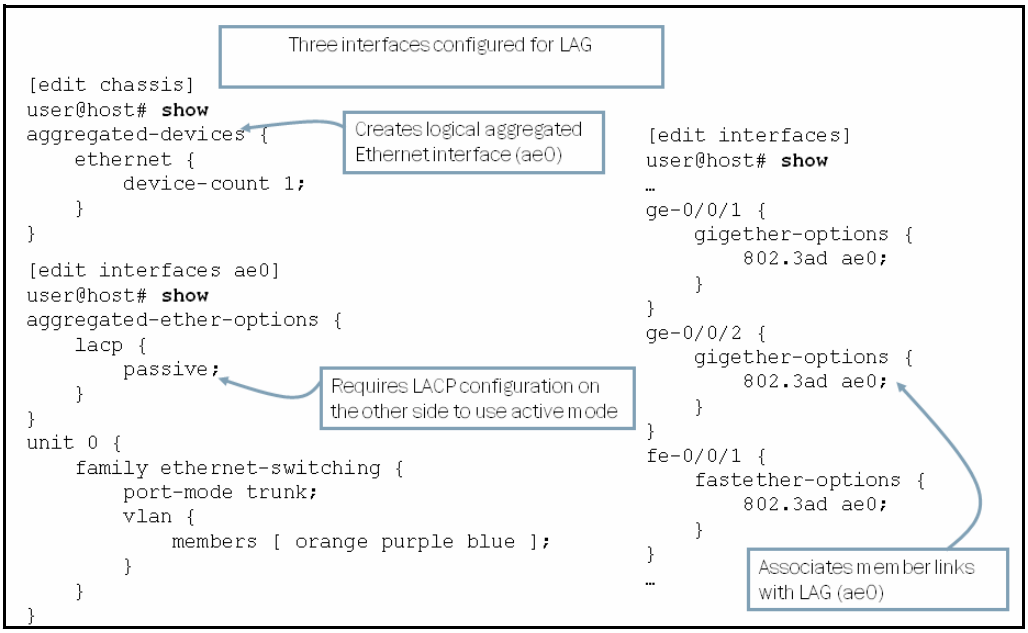
The graphic highlights a basic Multilink Point-to-Point Protocol (MLPPP) configuration. In this example, two serial interfaces function as member links for the configured bundle. The sample configuration is from the *host1-a* device.

The following is the configuration for the *host2-a* device:

```
interfaces {
    ls-0/0/0 {
        unit 0 {
```

```
            family inet {
                address 172.18.37.6/30;
            }
        }
    }
    se-1/0/0 {
        serial-options {
            clocking-mode internal;
        }
        unit 0 {
            family mlppp {
                bundle ls-0/0/0.0;
            }
        }
    }
    se-1/0/1 {
        serial-options {
            clocking-mode internal;
        }
        unit 0 {
            family mlppp {
                bundle ls-0/0/0.0;
            }
        }
    }
}
```

Configuration Examples: Part 4



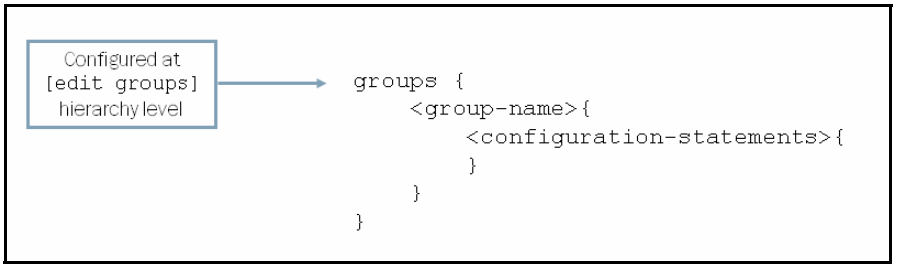
The graphic illustrates the steps used to configure a link aggregation group (LAG). The first step creates a logical aggregated Ethernet interface. In this example, we created a single aggregated interface, `ae0`. By default, no aggregated interfaces exist. To create an aggregated interface, simply add an aggregated device under the `[edit chassis]` hierarchy, as shown in the example on the graphic. Once this portion of the configuration is committed, the device creates the `ae0` interface. The following is an example that illustrates this behavior:

```
[edit]
user@host# run show interfaces terse |match ae

[edit]
user@host# edit chassis
[edit chassis]
user@host# set aggregated-devices ethernet device-count 1
[edit chassis]
user@host# commit
commit complete
[edit chassis]
user@host# run show interfaces terse |match ae
ae0                     up    down
```

The next step is to define the parameters associated with the `ae0` interface. As shown on the graphic, the `ae0` interface configuration includes at least one logical unit along with the desired logical interface properties. The example shows the `ae0` interface configured as an ethernet switch with three VLANs. The example on the graphic also includes the Link Aggregation Control Protocol (LACP) under the `aggregated-ether-options` hierarchy level. As previously indicated, if LACP is used, at least one side must be configured in active mode to successfully establish the connection.

Note that LAG support and configuration varies between the different Junos devices. For support information, always refer to the technical documentation for your specific product.

Configuration Groups



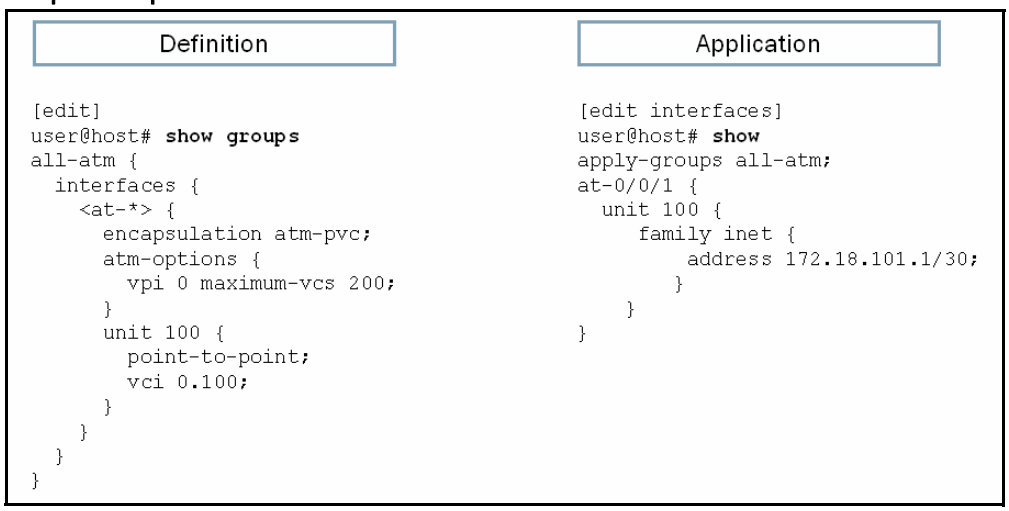
Configuration groups allow you to create a group containing configuration statements and to direct the inheritance of that group's statements in the rest of the configuration. You can apply the same group to different sections of the configuration, and different sections of one group's configuration statements can be inherited in different places in the configuration.

Configuration groups allow you to create smaller, more logically constructed configuration files, making it easier to configure and maintain the Junos OS. For example, you can group statements that repeat in many places in the configuration, such as when configuring interfaces, and thereby limit updates to just the group.

You can also use wildcards in a configuration group to allow configuration data to be inherited by any object that matches a wildcard expression.

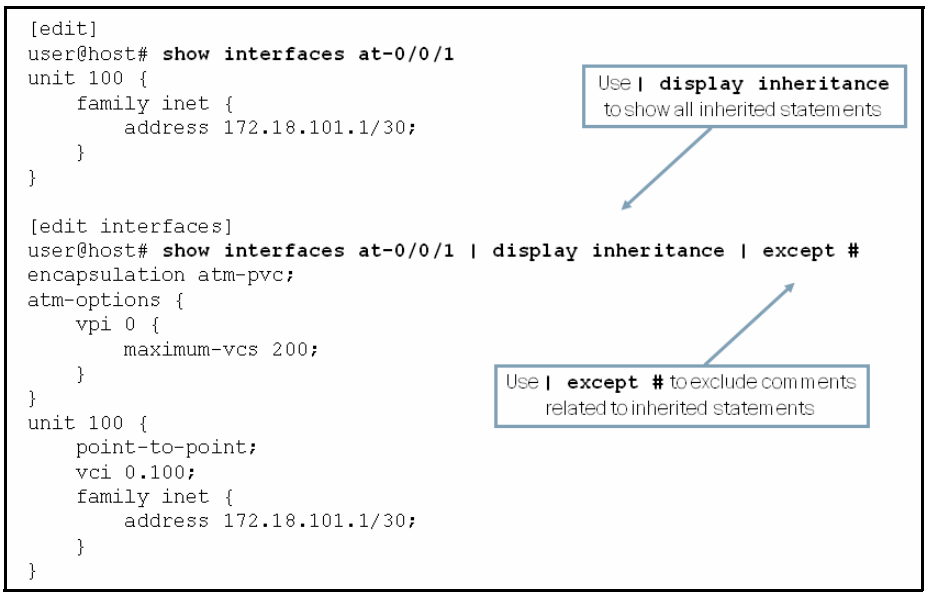
The configuration group mechanism is separate from the grouping mechanisms used elsewhere in the configuration, such as BGP groups. Configuration groups provide a generic mechanism that you can use throughout the configuration but that only the Junos command-line interface (CLI) recognizes. The individual software processes that perform the actions directed by the configuration receive the expanded form of the configuration; they have no knowledge of configuration groups.

Interface Group Example



You can use configuration groups to separate the common interface media parameters from the interface-specific addressing information. The example on the graphic places configuration data for ATM interfaces into a group called *all-atm*, which is applied at the `[edit interfaces]` hierarchy. In this example, all configuration parameters defined within the *all-atm* configuration group apply to the at-0/0/1 interface. If competing statements existed, the software would use the statements configured directly under the ATM interface.

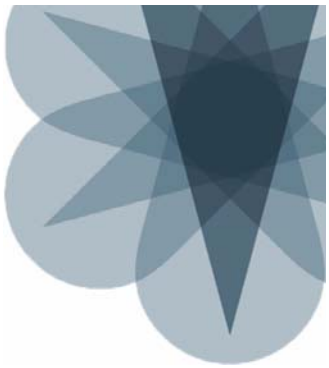
Displaying Inherited Configuration



Configuration groups can make determining the actual values used by a device running the Junos OS difficult, because configuration data can be inherited from configuration groups. To view the actual values used by a device running the Junos OS, use the **| display inheritance** option after the **show** command. This command displays the inherited statements at the level at which they are inherited and the group from which they have been inherited. As mentioned on the graphic, you can also add the **| except #** option to exclude the inheritance notes.

The following is the command illustrated on the graphic without the **| except #** command:

```
[edit]
user@host# show interfaces at-0/0/1 | display inheritance
##
## 'atm-pvc' was inherited from group 'all-atm'
##
encapsulation atm-pvc;
##
## 'atm-options' was inherited from group 'all-atm'
##
atm-options {
    ##
    ## '0' was inherited from group 'all-atm'
    ##
    vpi 0 {
        ##
        ## '200' was inherited from group 'all-atm'
        ##
        maximum-vcs 200;
    }
}
unit 100 {
    ##
    ## 'point-to-point' was inherited from group 'all-atm'
    ##
    point-to-point;
    ##
    ## '0.100' was inherited from group 'all-atm'
    ##
    vci 0.100;
    family inet {
        address 172.18.101.1/30;
    }
}
```



JNCIA-Junos Study Guide—Part 1

Appendix B: The J-Web Interface

This Appendix Discusses:

- The J-Web graphical user interface (GUI) and its tabs, key screens, and functions;
- How to add a new user;
- Basic interface configuration;
- Basic network monitoring; and
- Upgrading the Junos operating system.

The J-Web User Interface

The J-Web makes initial deployment very easy. No client software is necessary other than a standard Web browser. After initial configuration, you can return to J-Web for system monitoring and maintenance.

When you log in to J-Web, you always start by viewing the J-Web `Dashboard`. The `Dashboard` provides a quick glance at system status, ports, alarms, and utilization information.

The `Configure` tab allows you to configure the system in a point-and-click fashion or by a direct edit of the configuration in text format. Help is available by clicking the question mark (`?`) next to the various configuration options.

You can also view the results of configuration changes, such as routing table entries. You can view most details related to the **`show`** commands of the command-line interface (CLI) in J-Web using a point-and-click approach.

The `Troubleshoot` tab provides common network tools such as ping and traceroute to quickly assess network issues. You can use the `Maintain` tab to easily perform software upgrades and file system maintenance.

Logging In to J-Web

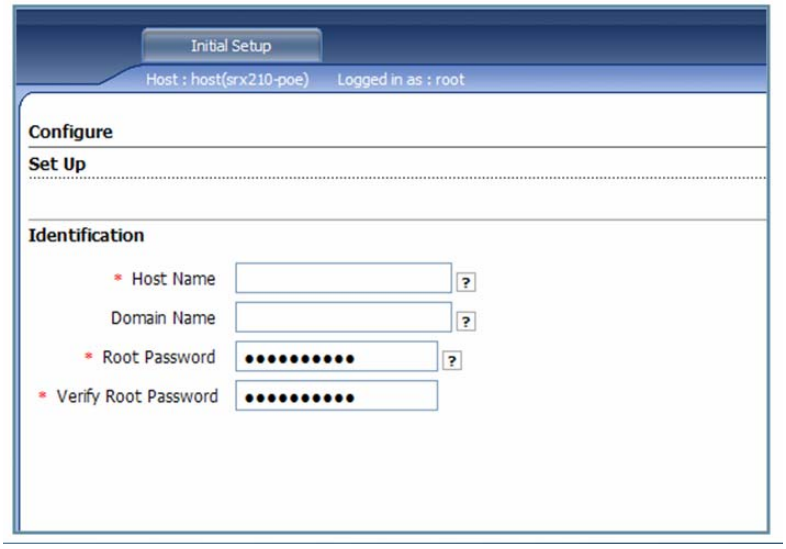
If you want remote access using J-Web, you must enable the HTTP or HTTPS service under the `[edit system services]` hierarchy level as shown in the following capture:

```
[edit system]
user@host# show services
ssh;
telnet;
web-management {
    http;
}
```

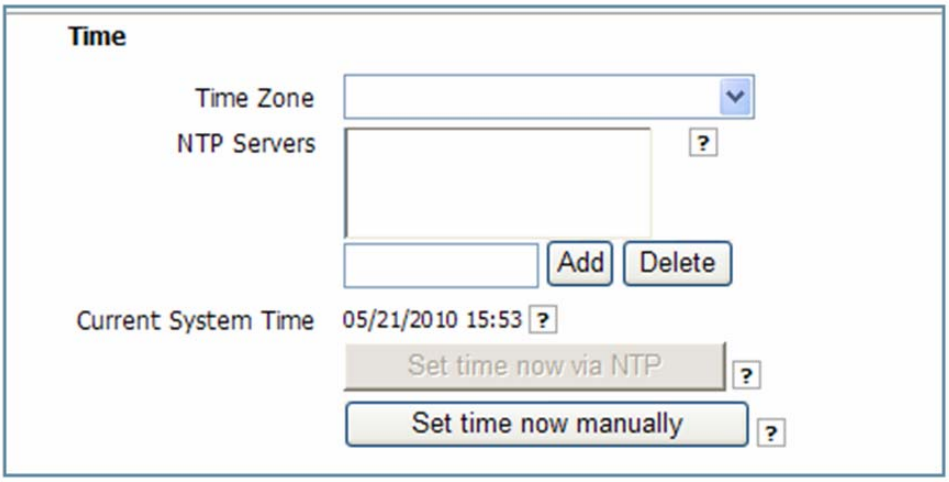
If you configure HTTPS, a local certificate for secure Web management is created automatically.

Once you configure a device running the Junos OS for access, you can log in using your Web browser. If you configured the system to use an external authentication mechanism such as a RADIUS server, J-Web will also use that mechanism for authentication. Otherwise, it uses the username and password configured on the local system.

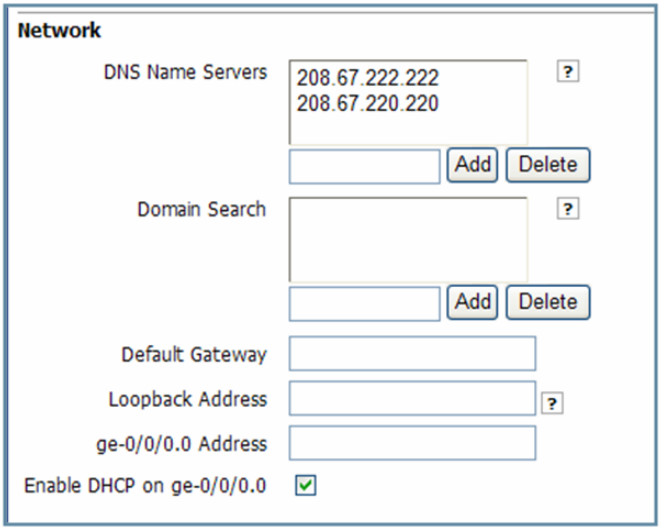
Initial Setup



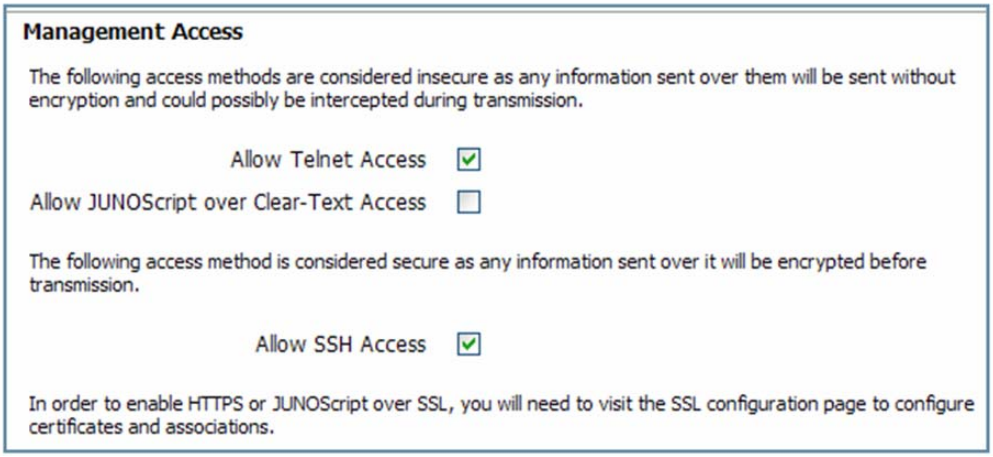
The initial setup screen appears the first time you log in to an unconfigured Junos device. You enter basic configuration information on this one-page setup screen. Any field with a red asterisk is a required field. Remember to click `Apply` (not shown on the graphic) when you finish with this screen.



The second section of the initial configuration screen allows you to configure the time. None of these fields are required. We recommend that you configure the time on the device.



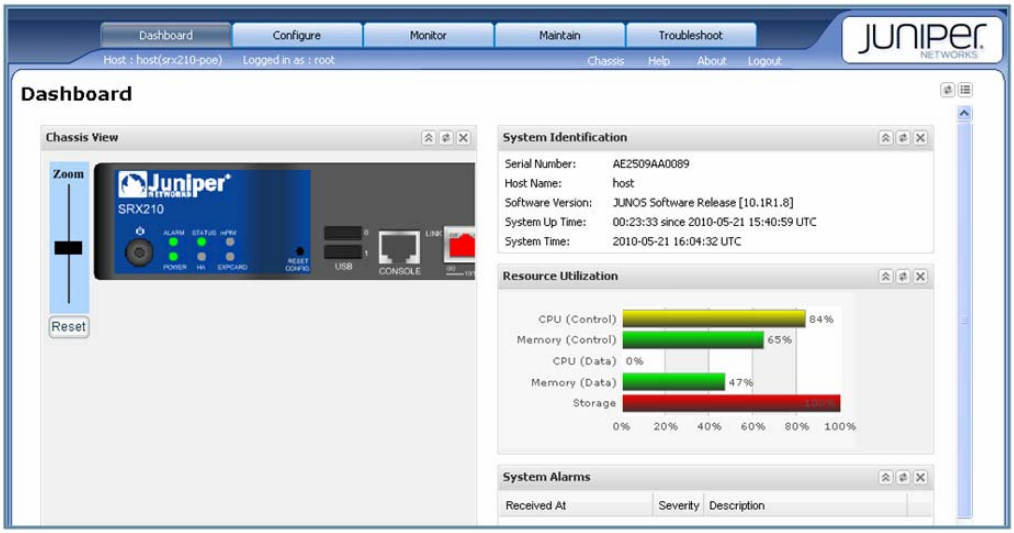
The next part of the initial setup involves a basic network configuration.



The last part of the initial setup screen involves configuring management access.

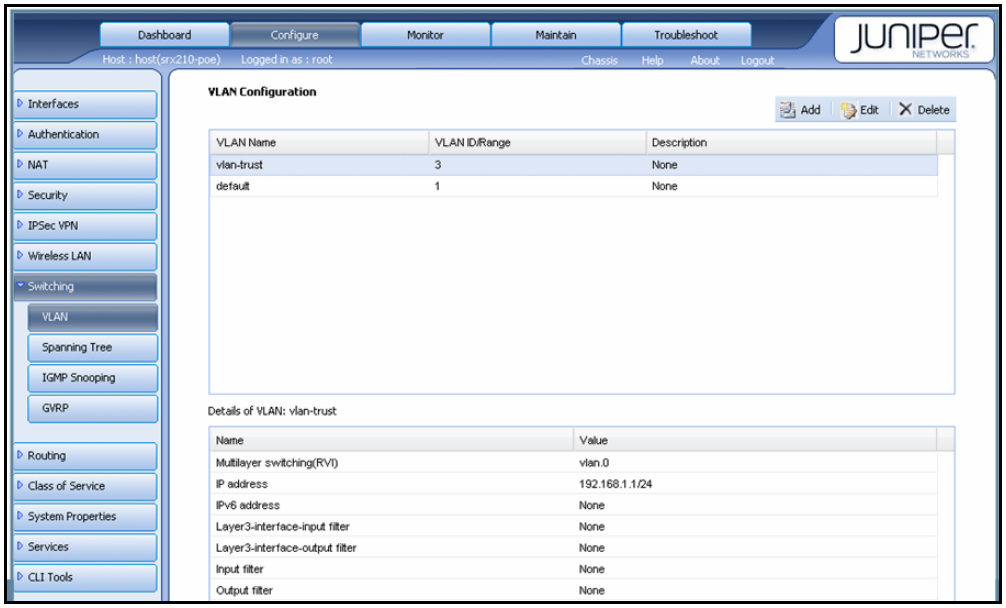
Ensure that you click `Apply` at the bottom of the window (not shown on the graphic), when you are done with the initial setup.

Quick Verification



The default J-Web tab is the `Dashboard` tab. The `Dashboard` provides a quick view of the system's current status along with other system-specific details.

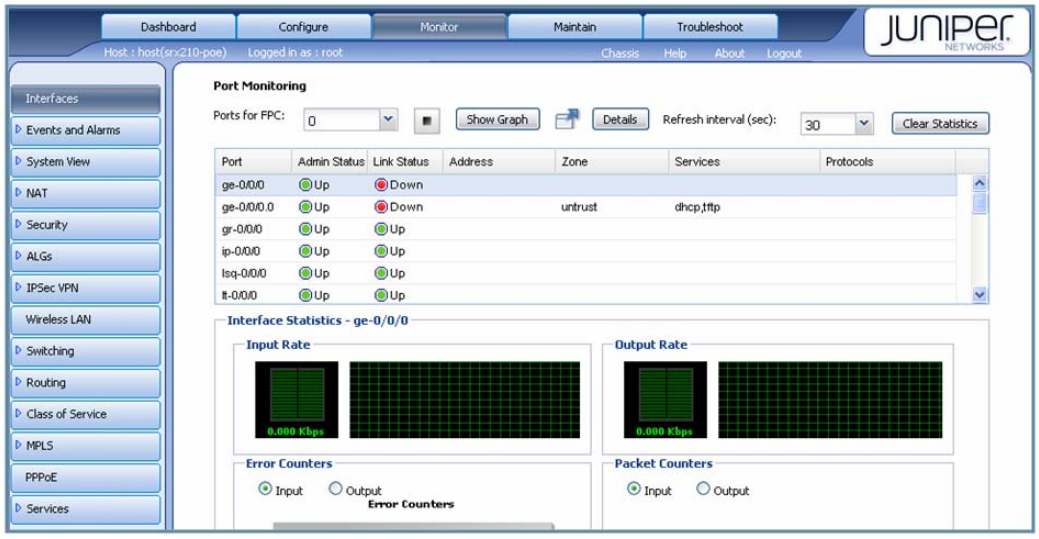
Performing Configuration Tasks with J-Web



J-Web offers an easy-to-use interface for configuring your Junos device. Choose which configuration hierarchy you want to view or edit in the left navigation menu. Information about that hierarchy appears on the main portion of the screen. You can select various options for viewing or editing. You can add new configuration options with the `Add` button or edit existing configuration options with the `Edit` button. These buttons and a `Delete` button are located near the top right of the screen.

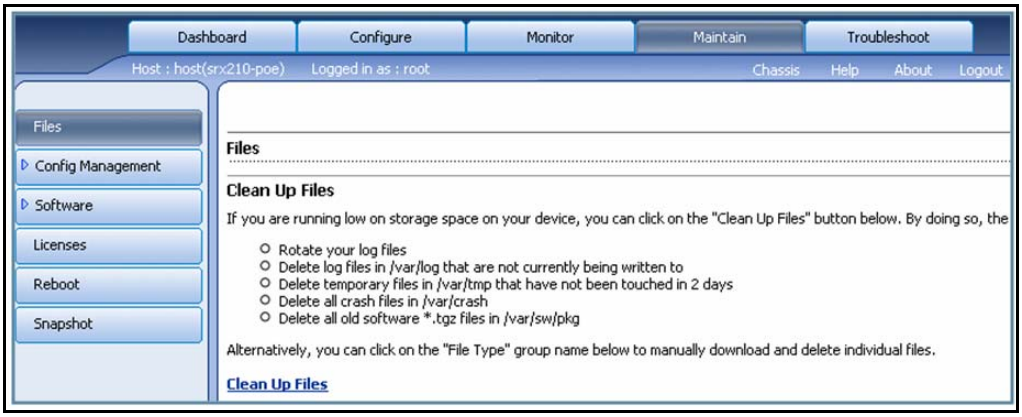
If you prefer to manipulate your configuration with a text-based approach, choose the `CLI Tools` option at the bottom of the navigation menu.

Performance Monitoring



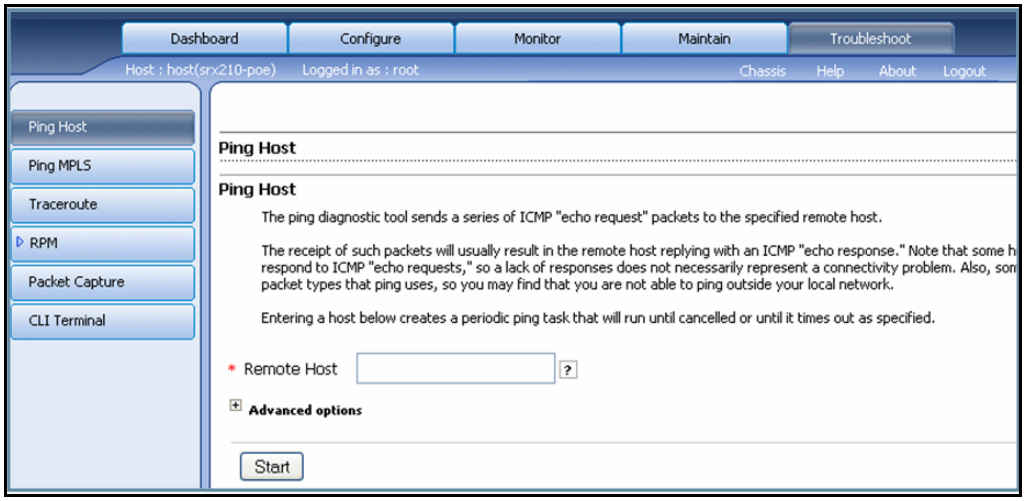
On the `Monitor` tab, you can view detailed real-time statistics and the results of configuration-related activity. As seen on the graphic, the `Interfaces` hierarchy provides statistics in a graphical fashion using colorful charts and graphs. Use the drop-down menus to customize your view. Hovering the mouse pointer over various parts of the screen presents you with more detailed information. Most of the hierarchies on the left side of the screen are carry-overs from the `Configure` tab. Selecting these options provides a point-and-click alternative over CLI **`show`** commands.

System Maintenance



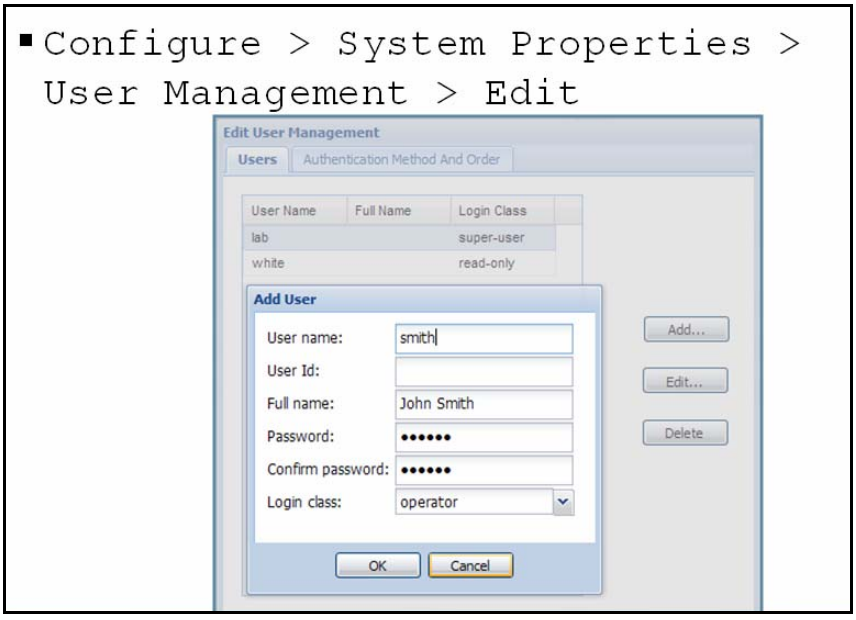
The `Maintain` tab provides an interface to manage file systems, the Junos OS, and configuration files. Under the `Files` section, you can download and delete log files, memory dump files, and other temporary files to keep your flash memory device from becoming too full. `Config Management` allows you to retrieve historical configuration files and to compare differences between configurations. Choosing `Software` provides methods for upgrading and downgrading the Junos OS. You can automate the upgrade process by specifying a remote FTP server to retrieve the Junos OS. The system then upgrades with the retrieved software and issues a reboot of the system to complete the upgrade process. The `Licenses` section provides the details on installed licenses on the system, allowing you to add licenses. The `Reboot` section allows you to schedule reboots and provides other options for rebooting the system. `Customer Support` provides a quick method for registering your device and retrieving support information required by Juniper Networks Technical Assistance Center (JTAC).

Troubleshooting Tools



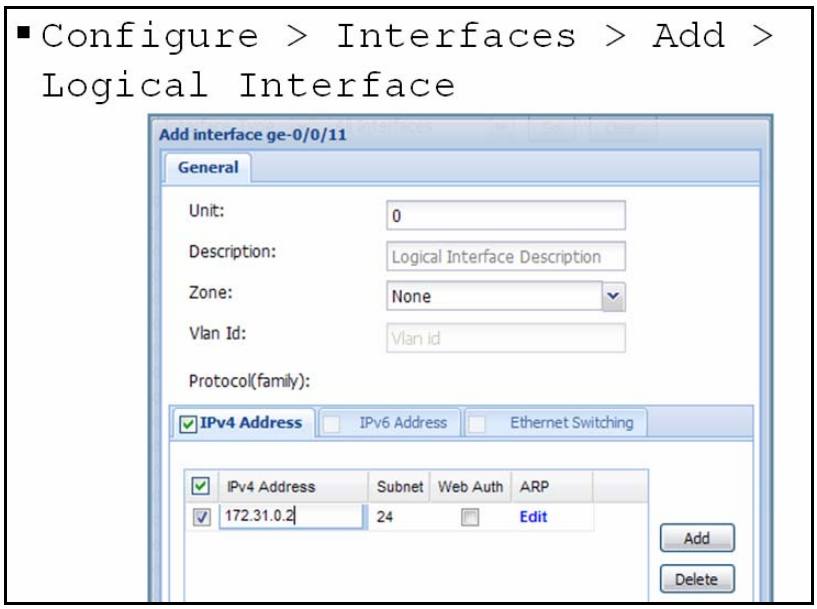
The `Troubleshoot` tab offers several useful utilities that can ease your troubleshooting efforts. You can troubleshoot individual ports, ping a remote host, perform a traceroute, capture packet dumps, and even open an embedded Java-based terminal session to your system.

Creating a New User



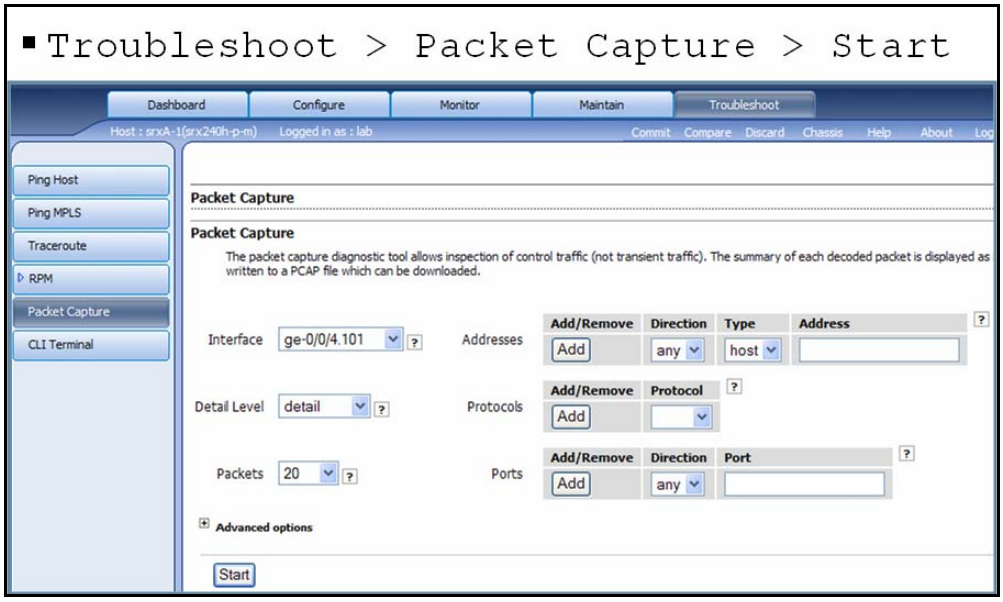
You can use the `Edit User Management` page to add new users to the device's local database. For each account, you define a login name and password for the user and specify a login class for access privileges.

Interface Configuration



You can use J-Web to configure logical interfaces on your Fast Ethernet or Gigabit Ethernet interfaces. You must have at least one logical interface configured on your physical Ethernet interface.

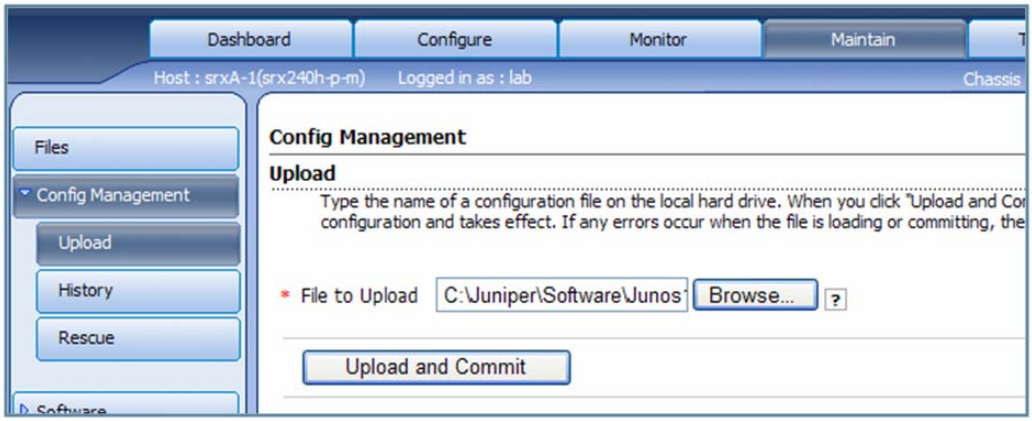
Network Monitoring



You can use the J-Web packet capture diagnostic tool when you need to quickly capture and analyze router control traffic on a device. Packet capture on the J-Web interface allows you to capture traffic destined for or originating from the Routing Engine. You can use the J-Web packet capture tool to compose expressions with various matching criteria to specify the packets that you want to capture. You can either choose to decode and view the captured packets in the J-Web interface as they are captured, or save the captured packets to a file and analyze them offline using packet analyzers such as Ethereal. The J-Web packet capture tool does not capture transient traffic.

Alternatively you can use the CLI **`monitor traffic`** command to capture and display packets matching a specific criteria.

The Junos OS Upgrade



You can use the J-Web interface to install software packages uploaded from your computer.