Chapter 1

# The Components of a Juniper Networks Router

## JNCIA EXAM OBJECTIVES COVERED IN THIS CHAPTER:

- ✓ Describe the function of the Routing Engine
- ✓ Define the portions of the JUNOS software architecture
- ✓ Describe the boot devices available to a Juniper Networks router
- ✓ Identify the steps involved in the JUNOS software boot sequence
- ✓ Identify options for manipulating "saved" configured files
- ✓ Describe the different Juniper Networks ASICs and their functions
- ✓ Identify the flow of a packet through the Packet Forwarding Engine
- ✓ Define an exception packet

As we discussed in the Introduction, you should already have a grasp of basic networking concepts. This includes the layers of the Open Systems Interconnection (OSI) model, the format and layout of an IP packet, and the function of a router as a network device. Additionally, you should understand the operation of both the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

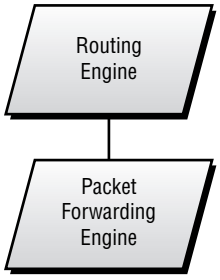
This chapter will introduce you to the basic components of the Juniper Networks family of routers. We start with a high-level examination of the two basic components of the system: the Routing Engine and the Packet Forwarding Engine. Next, we cover the specific details of the Routing Engine, including the JUNOS software modules, boot devices, and boot sequence pattern. In addition, we discuss the various modes of the software as well as some fail-over capabilities. We conclude with a discussion of the Packet Forwarding Engine ASICs and an example of a packet's flow through the router.

Let's first ensure that we have a common understanding of the terminology and an idea of how all the pieces fit together.

# Juniper Networks Router Design

The central design principle of the Juniper Networks platform centers on a separation of the control and forwarding planes within the router. The Routing Engine and the Packet Forwarding Engine, respectively, represent these planes. You can see this design concept in Figure 1.1.

**FIGURE 1.1** Juniper Networks router design




Let's examine each of these components in more detail.

# Routing Engine Overview

The *Routing Engine* in a Juniper Networks router is the central location for control of the system. This is where the intelligence of the router operates. You perform software upgrades and maintenance on the Routing Engine. In addition, you interface with the Routing Engine for monitoring and configuring the router.

## General Functions

Your experience with a Juniper Networks router begins with the Routing Engine. After connecting to the router, you supply authentication information (name and password) to the Routing Engine. After you're authenticated, you perform management and configuration operations within the Routing Engine. Troubleshooting tools like Telnet, ping, or traceroute operate from within the Routing Engine as well.

Since control of the router occurs in the Routing Engine, this is the logical location to store the JUNOS software. As such, the Routing Engine operates all routing protocols and makes all *routing table* decisions, building a master routing table with the best path to each destination selected. The router then places these best paths into the *forwarding table* on the Routing Engine and copies that same data into the forwarding table on the Packet Forwarding Engine. The forwarding table on the Packet Forwarding Engine allows the router to actually forward user data packets.

## Physical Composition

The intelligence of the Routing Engine software is not matched by equally intelligent hardware. In fact, the physical components are widely available. Each Routing Engine is based on an Intel PCI motherboard. The actual components of each Routing Engine depend on the model you are using and include the following:

**Routing Engine 2** The Routing Engine 2 is found in the Juniper Networks M-series routers (M5, M10, M20, M40, M40e, M160). It contains a 333MHz processor and 768MB of random access memory (RAM). File storage is handled by an 80MB internal flash drive and a 6.4GB traditional hard drive. When you use the Routing Engine 2 on an M40 router, it contains an LS 120 disk for external file storage; all other models use a removable PCMCIA flash card for this purpose.

**Routing Engine 3** The Routing Engine 3 is found in the Juniper Networks T-series routers (T320 and T640). Additionally, recent versions of the JUNOS software support the use of this Routing Engine in the M5, M10, M20, M40e, and M160 routers. The Routing Engine 3 contains a 600MHz processor and 2GB of RAM. File storage is handled by a 128MB internal flash drive and a 30GB traditional hard drive. The Routing Engine 3 uses a removable PCMCIA flash card for external file storage.

The RAM memory in the Routing Engine stores routing tables, forwarding tables, link-state databases, and operational memory space for the JUNOS software. The internal flash drive stores the JUNOS software and configuration files for the router. The hard drive is used to store a backup copy of the JUNOS software, log files, traceoptions output (debug), and user files.

While the differences between the Routing Engine models certainly control how much storage capacity you have in the router, they do not affect the operation of the JUNOS software. The internal flash drive is used for the same purposes and the software builds routing tables in the amount of RAM available to it. In fact, each version of the JUNOS software operates across all Routing Engine models. You never need to worry about replacing the Routing Engine hardware and then having to find the right software version to support it.

The hardware in a Juniper Networks Routing Engine is generally composed of the most common components available at its time of construction. As the cost of hardware decreases over time, you can expect that newer versions of the Routing Engine will contain more powerful hardware. Regardless, the requirements of the router design allow the Routing Engine to function quite well using the hardware described here.

## Packet Forwarding Engine Overview

The *Packet Forwarding Engine* is the central location for data packet forwarding through the router. The router's throughput speed and capacity are controlled by the specially designed hardware, which sets a Juniper Networks router apart from its competitors.

### General Functions

Simply put, the Packet Forwarding Engine provides industry-leading performance in the forwarding of data packets across any interface in the router. Achieving this type of throughput requires dividing the forwarding plane of the router into multiple segments controlled by *application-specific integrated circuits (ASICs)*. The interaction of these ASICs provides the forwarding path within a Juniper Networks router.

The function of the Juniper Networks ASICs and their role in packet forwarding is covered in the section "Packet Forwarding Engine Components" later in this chapter.

### Physical Composition

In contrast to the Routing Engine with its single motherboard and processor, the Packet Forwarding Engine contains a passive midplane as well as multiple boards and processors. Each circuit board is controlled by software that is fairly non-intelligent when compared to the JUNOS software on the Routing Engine.

The main portions of the Packet Forwarding Engine are the Physical Interface Card, the Flexible PIC Concentrator, and a switching control board. Each component contains an ASIC custom-designed by Juniper Networks engineers and manufactured by IBM. Each ASIC performs a specific function in the forwarding path of packets. (We discuss the specific functions of each ASIC in the section "Packet Flow" later in this chapter.)

### Switching Control Board

The switching control board contains a PowerPC CPU and 64MB of RAM that operates the components of the circuit board itself, but doesn't participate in packet forwarding. An additional 8MB (or 16MB in recent versions of the circuit board) of synchronized static random access memory (SSRAM) contains the forwarding table for the router. The Internet Processor ASIC is located on the control board and accesses the forwarding table for route lookups. Additionally, the control board contains an ASIC designed for packet storage memory management.

As a comparison, the 8MB of SSRAM on the switching control board holds approximately 450,000 forwarding table entries. As of this writing, the Internet has about 120,000 unique routing entries. This means that the Internet can double in size twice before you run out of storage capacity for your forwarding table.

Each router model uses a different name for the control board functionality. The possible names include:

**Forwarding Engine Board (FEB)** The *Forwarding Engine Board* is found in both the M5 and M10 platforms and integrates the circuit board with the FPC. Each router contains no more than one FEB, which is specific to either the M5 or the M10 chassis.

**System Switching Board (SSB)** The *System Switching Board* is found in the M20 platform. Each platform is configured to hold dual SSBs, but only one board is operational at any one time.

**System Control Board (SCB)** The *System Control Board* is found in the M40 platform. Each chassis contains no more than one SCB.

**Switching and Forwarding Module (SFM)** The *Switching and Forwarding Module* is found in the M40e and M160 platforms. Each M40e router can contain 2 SFMs, with only one operational at a time. The M160 router contains four SFMs working in parallel.

**Memory Mezzanine Board (MMB)** The *Memory Mezzanine Board* is found in the T320 and T640 platforms and is located on the FPC itself.

The T320 and T640 platforms are designed with a different internal architecture for the Packet Forwarding Engine. The M-series platforms are the focus of this book, and we point out differences with the T-series platforms where appropriate.

### Flexible PIC Concentrator

The *Flexible PIC Concentrator (FPC)* connects to both the switching control board and the router's interfaces within the Packet Forwarding Engine. A PowerPC CPU controls the FPC board, and it uses 64MB of RAM to operate the Embedded OS software. The PowerPC CPU doesn't participate in data packet forwarding, however. This is the function of a Juniper Networks ASIC, which is located on the FPC and interacts with the data packets as they enter and exit the router interfaces.

### Physical Interface Card

The physical media in your network connects to the *Physical Interface Card (PIC)* in your router. Up to four individual PICs are contained on an FPC. A media-specific ASIC is located on each PIC.

# Routing Engine Components

Let's now discuss the specific details and operation of the Routing Engine components. We start with the JUNOS software, examine the operation of the command-line interface (CLI), and finish with the fail-over capabilities of the Routing Engine.

## Software Architecture

The JUNOS software is based on the FreeBSD Unix operating system. The open source software is modified and hardened by Juniper Networks engineers to operate in the router's specialized environment. For example, some executables have been deleted while other utilities were de-emphasized. Additionally, certain daemons were added to enhance the routing functionality. The result of this transformation is the *kernel*, the heart of the JUNOS software.

The kernel is responsible for operating multiple daemons that perform the actual functions of the router. Each daemon operates in its own protected memory space, which is also controlled by the kernel. This separation provides isolation between the processes and resiliency in the event of a process failure. This is important in a core routing platform since a single process failure does not cause the entire router to cease functioning. Some common daemons include:

**Routing Protocol Daemon (rpd)** The router's protocols are controlled by the *Routing Protocol Daemon*. Its functionality includes all protocol messages, routing table updates, and implementation of routing policies.

**Device Control Daemon (dcd)** The router's interfaces are configured and maintained by the *Device Control Daemon*. This process controls both the physical and logical properties of the interfaces.

**Management Daemon (mgd)** The *Management Daemon* process controls all user access to the router. For example, the user's CLI is a client of mgd.

**Chassis Daemon (chassisd)** The *Chassis Daemon* process controls the properties of the router itself, including the interaction of the passive midplane, the FPCs, and the control boards.

**Packet Forwarding Engine Daemon (pfed)** The *Packet Forwarding Engine Daemon* process controls the communication between the Packet Forwarding Engine and the Routing Engine. For example, one of its functions is retrieving the interface input/output statistics from the Packet Forwarding Engine.

The kernel also generates specialized daemons as needed for additional functionality. Some examples include Simple Network Management Protocol (SNMP), Virtual Router Redundancy Protocol (VRRP), and Class of Service (CoS).

## Software Components

The JUNOS software is actually made up of multiple pieces working together to control the router's functions. Each section of the software is referred to as a *package* and contains files specific to its particular function. The current packages found in each copy of the JUNOS software are:

**`jkernel`** The *jkernel* package contains the basic components of the JUNOS software operating system.

**`jbase`** The *jbase* package contains additions to the JUNOS software since the last revision of the `jkernel` package.

**`jroute`** The *jroute* package contains the software that operates on the Routing Engine. This controls the Unicast routing protocols, the multicast routing protocols, and the Multiprotocol Label Switching (MPLS) signaling protocols. The package also contains the software for some daemons, such as mgd.

**`jpfe`** The *jpfe* package contains the Embedded OS software that controls the components of the Packet Forwarding Engine.

**`jdocs`** The *jdocs* package contains the complete JUNOS software documentation set.

**`jcrypto`** The *jcrypto* package contains software that controls various security functions, such as IP Security (IPSec) and Secure Shell (SSH). This package is available only in U.S. and Canadian versions of the software.

**`jbundle`** The *jbundle* package is a single file that contains all of the other packages we discussed previously.

### Getting Help from Your Router

The `jdocs` package is an interesting topic to discuss. It contains the entire JUNOS software documentation set on your router and is accessed through the user CLI. It is a handy tool to keep at your disposal.

You can find conceptual information on network topics by using the `help topic` command. For example, let's say you'd like to know more about setting up Open Shortest Path First (OSPF) backbone areas. Here's how you'd access this information and what the router would tell you:

```
user@Merlot> help topic ospf area-backbone
```

```
  Configure the Backbone Area


You must create a backbone area if your network consists of multiple areas. An ABR
must have at least one interface in the backbone area, or it must have a virtual
link to a router in the backbone area. The backbone comprises all area border
routers and all routers that are not included in any other area. You configure all
these routers by including the following area statement at the [edit protocols
ospf] hierarchy level (for routing instances, include the statement at the [edit
routing-instances routing-instance-name protocols ospf] hierarchy level):

   [edit protocols ospf]
       area 0.0.0.0;
```

When you are ready to configure your router to support an OSPF area, you can view specific configuration information using the `help reference` command:

```
user@Merlot> help reference ospf area


area

      Syntax

   area area-id;

      Hierarchy Level

   [edit protocols ospf],
   [edit routing-instances routing-instance-name protocols ospf]

      Description


Specify the area identifier for this router to use when participating in OSPF
routing. All routers in an area must use the same area identifier to establish
adjacencies.


Specify multiple area statements to configure the router as an area border router.
An area border router automatically summarizes routes between areas; use the area-
range statement to configure route summarization. By definition, an area border
router must be connected to the backbone area either through a physical link or
through a virtual link. To create a virtual link, use the virtual-link statement.
```

```
    To specify that the router is directly connected to the OSPF backbone, include
    the area 0.0.0.0 statement.

       Options

 area-id--Area identifier. The identifier can be up to 32 bits. It is common to
 specify the area number as a simple integer or an IP address. Area number 0.0.0.0
 is reserved for the OSPF backbone area.
```

## The JUNOS software Naming Convention

The JUNOS software follows a specific naming convention of *package-major_version <stage>released_version-type*. An example from the Merlot router shows this format:

```
user@Merlot> file list jbundle*
/var/home/user/jbundle-5.2R1.4-domestic-signed.tgz
/var/home/user/jbundle-5.2R2.3-domestic-signed.tgz
/var/home/user/jbundle-5.3R2.4-domestic-signed.tgz
```

The command output shown here contains information explained in the "Command-Line Interface" section later in this chapter.

This naming structure allows you to quickly determine if your version of the software supports a desired feature. The details of the naming convention are as follows:

**package** This represents the specific portion of the JUNOS software contained in the file. Examples include `jbundle`, `jroute`, and `jpfe`.

**major_version** This represents the major version of the JUNOS software located in the file. This is always shown in a two-integer format, as in 5.2 or 5.3.

**stage** This single capital letter represents the type of software in the file. Possible values include:

- R—Publicly released software (most common)
- A—Alpha version of the software
- B—Beta version of the software
- I—Internal or test version of the software

**released_version** Each major_version of the software may contain multiple releases. This field represents the specific release number contained in this file. The Merlot router shows several examples: 1.4, 2.3, and 2.4.

**type** Each jbundle package contains an additional field that represents whether the jcrypto package is contained. jcrypto is included in files marked with *domestic* and is omitted in files marked as *export*.

In addition, all packages may include the *signed* notation. This means that the package file is protected using the MD5 algorithm. When you apply the package to the router, the router runs the algorithm and compares the MD5 hash result to the stored value in the file. The package is used only when the values match, thereby protecting you from corrupted software files.

## Upgrading the Software

You upgrade the JUNOS software by using the request system software add ***filename*** command. This command loads a software file from some location, often the user's home directory, onto the internal flash on the Routing Engine. (Files and directories are discussed in the section "Manipulating Files on the Router" later in this chapter.) In the following example, we want to upgrade the Merlot router, which is currently running version 5.2R2.3, to version 5.3R2.4:

```
user@Merlot> show version brief
Hostname: Merlot
Model: m5
JUNOS Base OS boot [5.2R2.3]
JUNOS Base OS Software Suite [5.2R2.3]
JUNOS Kernel Software Suite [5.2R2.3]
JUNOS Packet Forwarding Engine Support [5.2R2.3]
JUNOS Routing Software Suite [5.2R2.3]
JUNOS Online Documentation [5.2R2.3]
JUNOS Crypto Software Suite [5.2R2.3]

user@Merlot> request system software add jbundle-5.3R2.4-domestic-signed.tgz
```

The command separates the jbundle package into its individual package components (jroute, jpfe, etc.), and the Routing Engine upgrades each package individually. If the new version relies on changes to the base operating system, the jbase package is also upgraded. Upon a successful completion, you must reboot the router to use the new software using the request system reboot command. You also have the option of an automatic reboot by using the *reboot* option in conjunction with the request system software add ***filename*** command. The process looks like this:

```
user@Merlot> request system software add jbundle-5.3R2.4-domestic-signed.tgz
reboot
Installing package '/var/home/lab/jbundle-5.3R2.4-domestic-signed.tgz' ...
Verified MD5 checksum of jbundle-5.3R2.4-domestic.tgz
```

```
Adding jbundle...
Verified MD5 checksum of jbase-5.3R2.4.tgz
Verified MD5 checksum of jboot-5.3R2.4
Verified MD5 checksum of jcrypto-5.3R2.4.tgz
Verified MD5 checksum of jdocs-5.3R2.4.tgz
Verified MD5 checksum of jkernel-5.3R2.4.tgz
Verified MD5 checksum of jpfe-5.3R2.4.tgz
Verified MD5 checksum of jroute-5.3R2.4.tgz
Auto-deleting old jroute...
Auto-deleting old jdocs...
Auto-deleting old jpfe...
Auto-deleting old jcrypto...
Restarting kmd ...
Auto-deleting old jkernel...
Adding jkernel...
Restarting watchdog ...
Adding jcrypto...
Adding jpfe...
Adding jdocs...
Adding jroute...
Saving package file in /var/sw/pkg/jbundle-5.3R2.4-domestic-signed.tgz ...
Saving state for rollback ...
Rebooting ...
shutdown: [pid 5584]

*** FINAL System shutdown message from root@HongKong-3 ***
System going down IMMEDIATELY
```


It is possible to upgrade a single JUNOS software package individually, but this practice is not recommended. Software packages operating at different version levels might have an interaction that causes unforeseen consequences. This type of upgrade should be completed only with the guidance of the Juniper Networks Technical Assistance Center (JTAC).

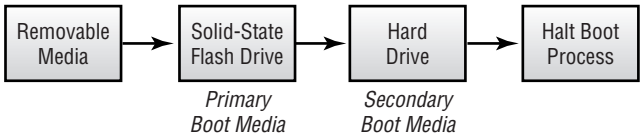
## Boot Sequence

Whether installed at the factory or upgraded in your network, the router stores bootable copies of the JUNOS software in three possible locations: the internal flash disk, the hard drive, or the removable media. Each location has the ability to load the software into memory and boot the router, but the primary boot media is the internal flash disk. The hard drive is considered the secondary boot location, while the removable media is used for disaster-recovery purposes.

A "factory fresh" router has the most recent JUNOS software version installed on the internal flash disk as well as the removable media.

As the router boots, it first runs a power-on self-test (POST) to verify that the basic system components are operating normally. The router then locates a copy of the JUNOS software and loads it into memory. The boot sequence of the router is shown in Figure 1.2.

**FIGURE 1.2** The JUNOS software boot sequence




The removable media is the first boot location examined. If the router finds a copy of the JUNOS software there, it loads the software on the router. This presents a possible hazard in your network since all existing files and file systems on the router are erased during this process. This type of boot process returns the router to a factory default-type environment and should be used only for disaster recovery. If no removable media is present, the router loads the software from the internal flash disk. This is considered the normal boot operation and should occur at each router start.

It is possible for the internal flash disk to become corrupted or otherwise unusable. In this situation, the router uses the hard drive as its boot location and displays a message to alert you of this issue. You can see this as you log into the router:

To successfully boot the router from the hard drive, you first need to copy the JUNOS software and other critical files to it with the `request system snapshot` command.

```
Merlot (ttyp0)

login: user
Password:

--- JUNOS 5.3R1.2 built 2002-04-30 01:40:52 UTC
---
--- NOTICE: System is running on alternate media device (/dev/ad1s1a).
---

user@Merlot>
```


Should your router encounter a problem and boot from the hard drive, please contact the JTAC for assistance.

## Command-Line Interface

At this point, we have the router booted and the appropriate software loaded on it. It is now time to monitor and configure the router using the *command-line interface (CLI)*.

The JUNOS software CLI contains two main modes: operational and configuration. The names adequately describe the functions permitted within each environment. Operational mode displays the current router status, and you use it for verifying and troubleshooting the router. Configuration mode, on the other hand, provides you with a method for altering the current environment.

### Operational Mode

You enter *operational mode* on the router after a successful login attempt. The router prompt displays your status graphically:

```
user@Merlot>
```

The default prompt for the JUNOS software is a combination of your username and the router hostname. In our case, the username is `user` and the hostname of the router is `Merlot`. In addition, the > character tells you that you are in operational mode.

As with most router operating systems, the JUNOS software uses a command hierarchy paradigm within operational mode. This allows you to find only the information you request in a timely manner. Here we are accessing the top level of the operational-mode hierarchy on the Merlot router:

```
user@Merlot> ?
Possible completions:
  clear                Clear information in the system
  configure            Manipulate software configuration information
  file                 Perform file operations
  help                 Provide help information
  monitor              Real-time debugging
  mtrace               Trace multicast path from a source to a receiver
  ping                 Ping a remote target
  quit                 Exit the management session
  request              Make system-level requests
  restart              Restart a software process
  set                  Set CLI properties, date, time, craft display text
  show                 Show information about the system
```

```
  ssh                 Open a secure shell to another host
  start               Start a software process
  telnet              Telnet to another host
  test                Diagnostic debugging commands
  traceroute          Trace the route to a remote host
```

You use this hierarchy level for several different purposes. For example, the `ping`, `telnet`, `traceroute`, and `ssh` commands allow the router to behave as an IP end host. The router generates these IP packets on the Routing Engine and sends them into the network through a particular interface. Commands such as `request`, `restart`, and `start` control the router's operations. In the "Upgrading the Software" section earlier in this chapter, we used the `request` command to load a new version of the JUNOS software.

You can view the router's current status by using the `show` command. The hierarchy located in this directory lets you access interface statistics, routing protocol information, and the current routing table. The `show` hierarchy directory on the Merlot router is:

```
user@Merlot> show ?
Possible completions:
  accounting          Show accounting profiles and records
  aps                 Show APS information
  arp                 Show system ARP table entries
  as-path             Show table of known AS paths
  bgp                 Show information about BGP
  chassis             Show chassis information
  class-of-service    Show information about class of service
  cli                 Show command-line interface settings
  configuration       Show configuration file contents
  connections         Show CCC connections
  dvmrp               Show information about DVMRP
  firewall            Show firewall counters and information
  helper              Show port-forwarding helper information
  host                Host name lookup service using domain name server
  igmp                Show information about IGMP
  ike                 Show IKE information
  ilmi                Show ILMI information
  interfaces          Show interface information
  ipsec               Show IPSec information
  ipv6                Show information about IPv6
  isis                Show information about IS-IS
  l2circuit           Show information about Layer 2 circuits
  l2vpn               Show information about Layer 2 VPNs
  ldp                 Show information about LDP
  log                 Show contents of a log file
```

```
  mpls                Show information about MPLS
  msdp                Show information about MSDP
  multicast           Show multicast information
  ntp                 Show Network Time Protocol information
  ospf                Show information about OSPF
  pfe                 Show Packet Forwarding Engine information
  pim                 Show information about PIM
  policer             Show interface policer counters and information
  policy              Show policy information
  rip                 Show information about RIP
  ripng               Show information about RIPng
  route               Show routing table information
  rsvp                Show information about RSVP
  sap                 Show session advertisement addresses
  snmp                Show SNMP information
  system              Show system information
  task                Show routing protocol per-task information
  ted                 Show information about TED
  version             Show software process revision levels
  vrrp                Show VRRP information
```

### Context-Sensitive Help

It is no accident that we've been utilizing the question mark (?) throughout this chapter to locate information. This character gives you *context-sensitive help* to navigate the command hierarchy. We often use the help function at a specific hierarchy level, but it also provides assistance in locating specific options within a particular level. For example, let's locate the possible commands starting with the letter `i` within the `show` hierarchy:

```
user@Merlot> show i?
Possible completions:
  igmp                Show information about IGMP
  ike                 Show IKE information
  ilmi                Show ILMI information
  interfaces          Show interface information
  ipsec               Show IPSec information
  ipv6                Show information about IPv6
  isis                Show information about IS-IS
```

To see information about the Intermediate System to Intermediate System (IS-IS) routing protocol, let's use the question mark within the next level of the command hierarchy like so:

```
user@Merlot> show isis ?
```

```
Possible completions:
  adjacency            Show the IS-IS adjacency database
  database             Show the IS-IS link-state database
  hostname             Show IS-IS hostname database
  interface            Show IS-IS interface information
  route                Show the IS-IS routing table
  spf                  Show information about IS-IS SPF calculations
  statistics           Show IS-IS performance statistics
```

The context-sensitive help system is a powerful tool when you're learning the command hierarchy and locating specific commands. The router also provides another tool to assist you in learning the CLI—the command completion function, which we discuss next.

### Command Completion

The JUNOS software CLI provides you with a *command completion* function. Each unique combination of characters at a particular hierarchy level expands into the full command when you use either the spacebar or the Tab key. For example, the characters `sh` are the most unique combination at the top of the operational hierarchy. You press the spacebar to complete the `show` command as follows:

```
user@Merlot> sh<space>ow
```

We further complete our command with the letter `c` followed by the Tab key:

```
user@Merlot> sh<space>ow c<tab>
                              ^
'c' is ambiguous.
Possible completions:
  chassis              Show chassis information
  class-of-service     Show information about class of service
  cli                  Show command-line interface settings
  configuration        Show configuration file contents
  connections          Show CCC connections
user@Merlot> show c
```

The router returns an error message telling us that there are multiple commands in the `show` hierarchy that start with the letter `c`. The output informs you that `'c' is ambiguous` and displays the possible commands that begin with the requested letter. The router maintains the command prompt at the position of the error and waits for you to enter more characters. We now complete our command:

```
user@Merlot> sh<space>ow c<tab>
                              ^
'c' is ambiguous.
```

```
Possible completions:
  chassis              Show chassis information
  class-of-service     Show information about class of service
  cli                  Show command-line interface settings
  configuration        Show configuration file contents
  connections          Show CCC connections
user@Merlot> show cli
CLI complete-on-space set to on
CLI idle-timeout disabled
CLI restart-on-upgrade set to on
CLI screen-length set to 24
CLI screen-width set to 80
CLI terminal is 'vt100'
CLI is operating in enhanced mode
```

The router's parsing of the CLI during your command typing provides you with an immediate syntax check. With the CLI, you'll never type out a long command and then after you press Enter be told that you made an error at the beginning of the command.

**Real World Scenario**

**Easing into the JUNOS software CLI**

The command completion functionality of the JUNOS software allows you to easily migrate from another vendor's software. Your mind and your fingers can still use the same keystrokes and produce the same results. For example, the command string `sh rou` completes into `show route` and displays the routing table:

```
user@Merlot> sh<space>ow rou<space>te

inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.24.2/32       *[Local/0] 10w2d 05:45:23
                      Local via so-0/3/0.0
10.0.31.0/24       *[Direct/0] 2w2d 06:42:26
                    > via fe-0/0/1.0
10.0.31.1/32       *[Local/0] 2w2d 06:42:26
                      Local via fe-0/0/1.0
172.64.0.0/16      *[Direct/0] 2w2d 06:50:26
                    > via fxp0.0
```

```
172.64.0.24/32     *[Local/0] 2w2d 06:50:26
                      Local via fxp0.0
192.168.24.1/32    *[Direct/0] 2w2d 06:42:26
                    > via lo0.0
```

The string `sh int` completes into `show interfaces` and displays the interfaces on the router:

```
user@Merlot> sh<space>ow int<space>erfaces

Physical interface: fe-0/0/0, Enabled, Physical link is Up
  Interface index: 11, SNMP ifIndex: 13
  Description: Where does this go
  Link-level type: VLAN-CCC, MTU: 1518, Speed: 100mbps, Loopback: Disabled,
  Source filtering: Disabled, Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps
  Current address: 00:90:69:6a:f0:00, Hardware address: 00:90:69:6a:f0:00
  Last flapped   : 2002-07-09 10:40:16 UTC (10w2d 06:24 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)
  Active alarms  : None
  Active defects : None

(Note: Information deleted for brevity)
```

Since the JUNOS software was built to support IPv4 as its primary protocol, the only thing you have to erase from your mind is the use of `ip` in your commands:

```
user@Merlot> sh<space>ow ip
                          ^
'ip' is ambiguous.
Possible completions:
  ipsec                Show IPSec information
  ipv6                 Show information about IPv6
user@Merlot> show ip
```

### Editing Command Lines

The router stores operational mode commands in a history buffer as you type them. This allows you to repeat a command by accessing the previous version and pressing the Enter key. When the CLI and your terminal emulator agree to use `vt100` as the character output, you can use the

left, right, up, and down arrows. These keystrokes provide you with access to the CLI history and allow you to easily edit your previous commands. The Backspace key is also enabled in `vt100` mode, and you use it to delete characters to the left of the cursor. To set your current terminal session to `vt100` mode, use the following:

```
user@Merlot> set cli terminal vt100
```

Regardless of your current terminal type, the JUNOS software CLI responds to common *editor macros (Emacs)* keystrokes to edit the command line. Some of the more useful commands are shown in Table 1.1.

**TABLE 1.1** Common Emacs Keystrokes

| Command | Effect |
|---|---|
| Ctrl+P | Displays the previous line in the CLI history buffer and is equivalent to the Up arrow key. |
| Ctrl+N | Displays the next line in the CLI history buffer and is equivalent to the Down arrow key. |
| Ctrl+B | Moves the cursor back one character and is equivalent to the Left arrow key. |
| Ctrl+F | Moves the cursor forward one character and is equivalent to the Right arrow key. |
| Esc+B | Moves the cursor back one word at a time. The Esc key must be released and re-pressed for each keystroke. |
| Esc+F | Moves the cursor forward one word at a time. The Esc key must be released and re-pressed for each keystroke. |
| Ctrl+A | Moves the cursor to the beginning of the current command line. |
| CrtI+E | Moves the cursor to the end of the current command line. |
| Ctrl+W | Deletes the word to the left of the cursor. |
| Ctrl+X | Deletes the entire current command line. |
| Ctrl+L | Redraws the current command line. |

### Operational Command Variables

Now that you are exposed to the command hierarchy and you know how to get help from the CLI, you can operate your Juniper Networks router efficiently. The router, however, provides

you with additional capabilities that you may find useful. One such option is the ability to attach variables to your commands through the use of the pipe key (|).

Each valid command in operational mode has this ability. You can see it as a final option when you use the context-sensitive help function:

```
user@Merlot> show cli ?
Possible completions:
  <[Enter]>            Execute this command
  authorization        Show authorization and authentication information
  history              Show list of previous commands
  |                    Pipe through a command
user@Merlot> show cli
```

The router's help system also works with the pipe functionality:

```
user@Merlot> show cli | ?
Possible completions:
  count                Count occurrences
  display              Display additional information
  except               Show only text that does not match a pattern
  find                 Search for the first occurrence of a pattern
  hold                 Hold text without exiting the --More-- prompt
  match                Show only text that matches a pattern
  no-more              Don't paginate output
  resolve              Resolve IP addresses
  save                 Save output text to a file
  trim                 Trim specified number of columns from start of line
user@Merlot> show cli
```

The options available to you include the following:

**count** This option prompts the router to count the lines in the output. You see only a single line returned with the total count listed; for example, `Count: 7 lines`.

**display** This option allows the router to show you additional data associated with the command. In operational mode, only the `xml` switch is accessible to view the Extensible Markup Language (XML) tags for each command.

**except** This option allows you to omit any line in the output containing the text string you provide.

**find** This option prompts the router to begin the output at the first occurrence of the text string you provide.

**hold** The router automatically paginates its output based on the current terminal screen length. When the end of the output is reached, the router returns to the command prompt. This option prevents the router from automatically ending the pagination process when the end of the output is reached.

**match** This option prompts the router to display only lines in the output containing the text string you provide.

**no-more** This option causes the router to not paginate the output.

**resolve** This option causes the router to resolve IP addresses in the output to hostnames, if possible. You must configure the router with the IP address of a domain name server to use this option effectively.

**save** This option automatically saves the output to the filename you provide. You see a single line returned with the number of lines saved to the file; for example, `Wrote 27 lines of output to 'saved-file'`.

**trim** This option prompts the router to omit the number of columns you supply from the output, beginning with the left-hand side of the output. You might use this command when your terminal width is small and you need to see data without a line wrap.

The router also gives you the ability to combine multiple pipe options together for maximum flexibility. Suppose you want to know how many logical interfaces on your router have an IP address configured. The `show interfaces terse` command supplies this information:

```
user@Merlot> show interfaces terse
Interface       Admin Link Proto Local                 Remote
fe-0/0/0        up    up
fe-0/0/0.100    up    up   ccc
fe-0/0/0.200    up    up   ccc
fe-0/0/1        up    up
fe-0/0/1.0      up    up   inet  10.0.31.1/24
fe-0/0/2        up    down
fe-0/0/3        up    down
so-0/3/0        up    up
so-0/3/0.0      up    up   inet  10.0.24.2           --> 0/0
                           mpls
so-0/3/1        up    down
so-0/3/2        up    down
so-0/3/3        up    down
fxp0            up    up
fxp0.0          up    up   inet  172.64.0.24/16
fxp1            up    up
fxp1.0          up    up   tnp   4
gre             up    up
ipip            up    up
lo0             up    up
lo0.0           up    up   inet  192.168.24.1        --> 0/0
lsi             up    up
mtun            up    up
```

```
pimd            up    up
pime            up    up
tap             up    up
```

You can find the IP addresses by displaying only lines with the `inet` string:

```
user@Merlot> show interfaces terse | match inet
fe-0/0/1.0      up    up   inet  10.0.31.1/24
so-0/3/0.0      up    up   inet  10.0.24.2          --> 0/0
fxp0.0          up    up   inet  172.64.0.24/16
lo0.0           up    up   inet  192.168.24.1       --> 0/0
```

Finally, you can allow the router to count the output lines for you:

```
user@Merlot> show interfaces terse | match inet | count
Count: 4 lines
```

We discuss the assignment of IP addresses to interfaces and the meaning of `inet` in Chapter 2, "Interfaces."

### Modifying the Command Output

Each time you enter a command, the router generates the entire output before displaying any characters on your screen and stores that information in a buffer. When the display output is longer than your terminal length, the router paginates the output by displaying a prompt of `---(more 18%)---`. Not only does this tell you that more information is to follow, but it also reveals how much of the output buffer you have seen. In our case, we've viewed 18 percent of the total output.

Each time the router stops at a page break, you have the option of modifying and manipulating the output. You access these features by pressing the **h** key at the `(more)` prompt. The list of options looks like this:

```
---(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
    Move down one page:                    Space, f, ^F, or Right-Arrow
```

```
    Move to bottom of output:                              G, ^E, or End
    Move to top of output:                                g, ^A, or Home
    Move up half display:                                        u or ^U
    Move up one line:             k, Delete, Backspace, ^P, or Up-Arrow
    Move up one page:                               b, ^B, or Left-Arrow
    Quit automore:                                    q, Q, ^K, or Clear
    Redraw display:                                             ^L or ^R
    Repeat a keystroke command 1 to 9 times:                  Meta-1..9
    Repeat last search:                                               n
    Save output to a file:                          s or S <filename/url>
    Search backwards thru the output:                          ?<string>
    Search forwards thru the output:                           /<string>
---(End of Help)---
```

While the number of possible options and keystrokes is too numerous to detail here, we can point out some commonly used ones. You access the bottom (or end) of the output buffer with the Ctrl+E keystroke. This is useful when examining log files where new information is placed at the end of the file. You can move backward through any router output with the Ctrl+B sequence. This is handy for viewing information earlier in the output without retyping the command over again. You can exit from the output and return to the command line at any time by using the q key. Finally, you can search for a particular string in the output with the forward slash (/) key. This moves your prompt to the first occurrence of the supplied string and paginates the output at that point. This is similar to the `find` pipe variable.

## Configuration Mode

At some point, you're going to want to configure your router. After all, a router without configured interfaces and routing protocols is a large hunk of steel and circuitry that doesn't accomplish much. You access the router's *configuration mode* hierarchy with either the `configure` or `edit` command:

```
user@Merlot> configure
Entering configuration mode

[edit]
user@Merlot#
```

As with operational mode, the router uses the prompt to visually show you that you are in configuration mode. The > is changed into the pound character (#), and your current level in the hierarchy is displayed above the router's hostname. The `[edit]` portion of the output on Merlot tells us that we are at the top of the configuration hierarchy. We can view the command options at this level with the context-sensitive help system:

```
[edit]
user@Merlot# ?
```

```
Possible completions:
  <[Enter]>           Execute this command
  activate            Remove the inactive tag from a statement
  annotate            Annotate the statement with a comment
  commit              Commit current set of changes
  copy                Copy a statement
  deactivate          Add the inactive tag to a statement
  delete              Delete a data element
  edit                Edit a sub-element
  exit                Exit from this level
  help                Provide help information
  insert              Insert a new ordered data element
  load                Load configuration from an ASCII file
  quit                Quit from this level
  rename              Rename a statement
  rollback            Roll back database to last committed version
  run                 Run an operational-mode command
  save                Save configuration to an ASCII file
  set                 Set a parameter
  show                Show a parameter
  status              Display users currently editing the configuration
  top                 Exit to top level of configuration
  up                  Exit one level of configuration
  update              Update private database
```

**Real World Scenario**

### Using the *run* Command

One very useful command that exists in configuration mode is `run`. When you use this command, the router allows you access to operational mode commands from within the configuration. This flexibility enables you to easily verify information on the router. Let's look at an example.

Suppose that you connect to a router using Telnet and enter configuration mode to enable the OSPF routing protocol. After navigating to the `[edit protocols ospf]` hierarchy directory, you can't recall the interface names on this particular router. You could look at a network map for this information, but this option is not always available. You are now left to ask the router for the information. A router from another vendor may require you to exit the configuration to use the `show interfaces` command. The JUNOS software, however, provides this ability from within the configuration:

```
[edit protocols ospf]
```

```
user@Merlot# run show interfaces
Physical interface: so-0/0/0, Enabled, Physical link is Up
  Interface index: 11, SNMP ifIndex: 13
  Description: Sydney to Sao Paulo
  Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode,
  Speed: OC3, Loopback: None, FCS: 16, Payload scrambler: Enabled
  Device flags   : Present Running
  Interface flags: Point-To-Point SNMP-Traps
  Link flags     : Keepalives
  Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
  Keepalive: Input: 27244 (00:00:06 ago), Output: 27293 (00:00:06 ago)
  LCP state: Opened
  NCP state: inet: Opened, inet6: Not-configured, iso: Opened, mpls: Opened
  CHAP state: Not-configured
  Last flapped   : 2002-09-30 12:12:42 UTC (3d 03:21 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)
  SONET alarms   : None
  SONET defects  : None
(Note: Information deleted for brevity)
```

Another requirement that network engineers often encounter is the desire to examine the current routing table. Again, the router provides this ability from within configuration mode:

```
[edit protocols ospf]
user@Merlot# run show route

inet.0: 23 destinations, 24 routes (23 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1.1.1.0/24         *[OSPF/10] 00:40:41, metric 4
                    > via so-0/0/0.0
10.200.200.0/24    *[Direct/0] 1d 17:31:30
                    > via fe-0/3/0.0
10.200.200.1/32    *[Local/0] 3d 03:24:34
                       Local via fe-0/3/0.0
10.222.3.0/24      *[OSPF/10] 00:40:41, metric 3
                    > via so-0/0/0.0
10.222.5.2/32      *[Local/0] 3d 03:24:35
                       Reject
```

```
10.222.6.0/24      *[OSPF/10] 00:40:41, metric 2
                    > via so-0/0/0.0
10.222.44.0/24     *[OSPF/10] 00:40:41, metric 3
                    > via so-0/0/0.0
10.222.45.0/24     *[OSPF/10] 00:40:41, metric 3
                    > via so-0/0/0.0
(Note: Information deleted for brevity)
```

These are only two examples of using the `run` command to your advantage. Just keep in mind that this capability is usable for any operational-mode command within the JUNOS software.

Just because you've entered configuration mode, the router doesn't stop assisting you as it did in operational mode. The pipe variables are available with each command, the output paginates with the `(more)` prompt, and the Emacs editor strings are a usable feature in configuration mode. The CLI stills uses the command-completion function:

```
[edit]
user@Merlot# st<space>atus
Users currently editing the configuration:
  user terminal d0 (pid 23892) on since 2002-09-25 14:30:27 UTC
      [edit]
```

If you examine the command options for configuration mode, you'll notice that the `st` characters are the most significant way to access the `status` command. The output from the Merlot router shows the users who are currently in configuration mode, how long they have been in that mode, and what their current configuration hierarchy level is.

### Navigating within the Hierarchy

One method for conceptually viewing the configuration mode hierarchy is in a vertical fashion, with the top of the directory structure at the top of a tree. Each branch of the tree below the root forms a subdirectory below it. As is common with a directory system, each top-level subdirectory can branch out into its own set of subdirectories, as shown in Figure 1.3.

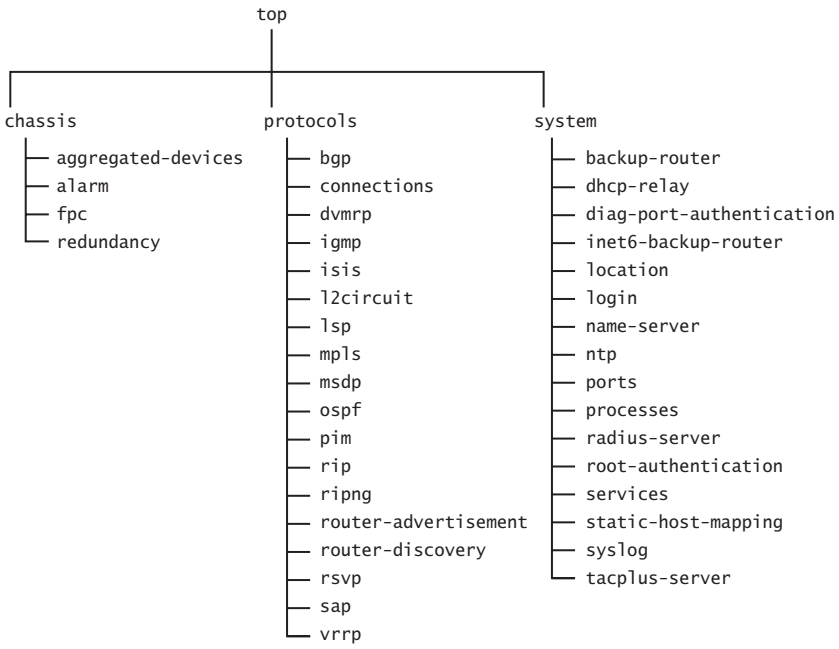
You navigate downward through this structure to the next lower directory by using the `edit` command:

```
[edit]
user@Merlot# edit protocols

[edit protocols]
user@Merlot#
```

We continue into one of the next lower directories:

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

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

**FIGURE 1.3** Configuration mode hierarchy directories




After reaching your desired directory, you can change the current configuration by using either the `set` or `delete` command as discussed in the next section, "Altering the Configuration."

The vertical nature of the configuration hierarchy requires you to always move in an up/down direction. We looked at how to move down to a directory, so let's see how to move up a directory. Quite simply, you use the `up` command to move up a directory level:

```
[edit protocols ospf]
user@Merlot# up

[edit protocols]
user@Merlot# up
```

```
[edit]
user@Merlot#
```

The JUNOS software allows you to reach any lower directory in the hierarchy by entering multiple directories with the `edit` command:

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

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

Conversely, the `top` command takes you to the top of the configuration hierarchy in a single step:

```
[edit protocols ospf]
user@Merlot# top

[edit]
user@Merlot#
```

Finally, if you are currently in a lower configuration directory, such as `[edit protocols ospf]`, and you wish to move to a different directory, such as `[edit routing-options static]`, you can do so by combining the `top` and `edit` commands:

```
[edit protocols ospf]
user@Merlot# top edit routing-options static

[edit routing-options static]
user@Merlot#
```

### Altering the Configuration

While moving around the configuration hierarchy is a valuable skill, at some point you are going to want to actually configure the router. Each directory in the hierarchy may contain variables that you can add or remove from the configuration. Continuing our tree analogy results in these variables becoming the leaves on each branch of the tree. You enter new information into the configuration with the `set` command:

```
[edit]
user@router# edit system

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

The router now has a hostname of `Merlot` instead of `router`. Notice that the *`host-name`* variable is actually in the `[edit system]` hierarchy directory. We used the `edit` command to move into that directory and then configured the hostname. In the previous section, "Navigating within the Hierarchy," we described the ability to add multiple directories to the `edit` command. The `set` command works in a similar manner. You can enter multiple directory names between the variable and the `set` command as long as the directories are in a direct downward line. Let's move back to the top of the hierarchy and change the hostname to `Shiraz`:

```
[edit system]
user@router# top

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

You can view the changes you've made to the configuration by issuing the `show` command. This command displays any configuration in your current directory and all subdirectories below your current location. Using this command at the top of the hierarchy displays the entire configuration:

```
[edit]
user@router# show
version 5.3R1.2;
system {
    host-name Shiraz;
    root-authentication {
        encrypted-password "$1$ZwtQb$cDpgAVcfDl/MLhTC1ZqQ4/"; # SECRET-DATA
    }
    login {
        user user {
            class super-user;
            authentication {
                encrypted-password "$1$/7NAOjwP$iwCrhoAqH38Kqh91AQFuY."; #
SECRET-DATA
            }
        }
    }
    radius-server {
        172.30.10.1;
    }
    services {
        telnet;
    }
    syslog {
        user * {
```

```
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
    }
}
```

To view the configuration just within the `[edit system]` directory, you may either move to that level with the `edit` command or add the hierarchy name to the `show` command from the top of the configuration:

```
[edit]
user@router# show system
host-name Shiraz;
root-authentication {
    encrypted-password "$1$ZwtQb$cDpgAVcfDl/MLhTC1ZqQ4/"; # SECRET-DATA
}
login {
    user user {
        class super-user;
        authentication {
            encrypted-password "$1$/7NAOjwP$iwCrhoAqH38Kqh91AQFuY."; # SECRET-DATA
        }
    }
}
radius-server {
    172.30.10.1;
}
services {
    telnet;
}
syslog {
    user * {
        any emergency;
    }
    file messages {
        any notice;
        authorization info;
    }
}
```

## Real World Scenario

### Using the *set* Command

We discussed how you have the ability to use the set command from the top of the configuration to change variables in the configuration. As you become more proficient with using the JUNOS software CLI, you might start taking advantage of this capability. To assist you in this making this transition, let's examine some details of the router's output.

Suppose we begin configuring the router in the [edit system] directory. The possible options at that hierarchy level are:

```
[edit system]
user@router# set ?
Possible completions:
+ apply-groups         Groups from which to inherit configuration data
+ authentication-order  Order in which authentication methods are invoked
> backup-router        IPv4 router to use while booting
  compress-configuration-files  Compress the router configuration files
  default-address-selection  Use system address for locally originated traffic
> dhcp-relay           BOOTP/DHCP relay configuration
> diag-port-authentication  Authentication for the diagnostic port
  domain-name          Domain name for this router
+ domain-search        List of domain names to search
  host-name            Host name for this router
> inet6-backup-router  IPv6 router to use while booting
> location             Location of the system, in various forms
> login                Users, their classes and passwords
  mirror-flash-on-disk  Mirror contents of the flash drive onto hard drive
> name-server          DNS name servers
  no-redirects         Disable ICMP redirects
  no-saved-core-context  Don't save context information for core files
> ntp                  Network Time Protocol services
> ports                Craft interface RS-232 ports
> processes            Process control
> radius-server        RADIUS server configuration
> root-authentication  Authentication information for the root login
  saved-core-context   Save context information for core files
> services             System services
> static-host-mapping  Static host name database mapping
```

```
> syslog              System logging facility
> tacplus-server      TACACS+ server configuration
  time-zone           Time zone definition name (<continent>/<major-city>)
[edit system]
user@router# set
```

When you examine the output closely, you might notice that some command options are preceded with a character—either an angle bracket (>) or a plus sign (+). These characters, as well as their absence, carry a special meaning when you use the `set` command.

The angle bracket is used to designate lower-level directories. In our case, the `name-server` option is really a subdirectory of `[edit system]`. The plus sign shows command variables you can configure that may have multiple values assigned. For example, the `authentication-order` option tells the router how to authenticate users who log in. You can assign a single authentication method or multiple methods. Finally, some options do not have any characters preceding them. These are configurable variables, such as `host-name`, that may contain only a single possible value.

You remove variables from the configuration with the `delete` command. Examining the earlier output shows that the router is currently configured to communicate with a remote authentication (RADIUS) server at address 172.30.10.1. This requirement is no longer valid, so we remove it from the router's configuration and verify that it is deleted:

```
[edit]
user@router# delete system radius-server 172.30.10.1

[edit]
user@router# show system
host-name Shiraz;
root-authentication {
    encrypted-password "$1$ZwtQb$cDpgAVcfDl/MLhTC1ZqQ4/"; # SECRET-DATA
}
login {
    user user {
        class super-user;
        authentication {
            encrypted-password "$1$/7NAOjwP$iwCrhoAqH38Kqh91AQFuY."; # SECRET-DATA
        }
    }
}
```

```
services {
    telnet;
}
syslog {
    user * {
        any emergency;
    }
    file messages {
        any notice;
        authorization info;
    }
}
```

### The Candidate Configuration

You may have noticed that we've been changing the hostname of the router but that the router's prompt hasn't changed. This brings us to a very important point concerning how a Juniper Networks router behaves. When you enter configuration mode, you are actually viewing (and changing) a file called the *candidate configuration*. The candidate configuration allows you to make configuration changes without causing operational changes to the current operating configuration, called the *active configuration*. The router implements the changes in the candidate configuration when you use the `commit` command. (We discuss this function in the "Using the `commit` Command" section later in this chapter.) This abstraction allows you the flexibility to alter your configuration without causing damage to your current network operations.

You may enter or exit configuration mode as many times as you wish without implementing your changes. If you do this several times, you may find that you've forgotten the exact changes you've made. In this situation, you can utilize a pipe command called `compare` in conjunction with the `show` command. This prompts the router to compare the current candidate configuration to the active configuration running on the router. Differences between the two files are displayed with either a plus (+) or a minus (-) sign. The plus sign represents variables in the candidate configuration that are not present in the active configuration; you've added them to the file. The minus sign shows the opposite; you've deleted variables from the file. In other words, the candidate configuration doesn't have items found in the active configuration.

Let's use this command on our router to see the difference between the candidate and active configurations:

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

We see that `host-name Shiraz` was added to the candidate configuration and that `host-name router` has been removed. This follows the configuration changes we implemented in the previous section, "Altering the Configuration."

The `show | compare` command displays only the differences between the two files. All other portions of the configuration files are not shown.

### Saving and Loading Configuration Files

The fact that the candidate configuration is a file that you edit also provides other advantages to you. You can save the current candidate configuration to a file on the router. Alternatively, you can load existing files into the router. Let's examine one example of how to use these options.

Suppose you are burning in (initially configuring) a number of routers in your network. You might want to have the common configuration components from the first router saved to more easily configure the remaining routers. Let's configure the first router with the common elements and use the `save` command from the top of the configuration hierarchy:

```
[edit]
user@router# show
version 5.3R1.2;
system {
    host-name Shiraz;
    root-authentication {
        encrypted-password "$1$ZwtQb$cDpgAVcfDl/MLhTC1ZqQ4/"; # SECRET-DATA
    }
    login {
        user user {
            class super-user;
            authentication {
                encrypted-password "$1$/7NAOjwP$iwCrhoAqH38Kqh91AQFuY."; #
SECRET-DATA
            }
        }
    }
    services {
        telnet;
    }
    syslog {
        file messages {
            any notice;
            authorization info;
```

```
        }
    }
}

[edit]
user@router# save common
Wrote 24 lines of configuration to 'common'
```

The router creates (or overwrites) the file called ***common*** and places the candidate configuration in it. We place these configuration elements on other routers with the `load` command. You have two main options for loading the files—`override` and `merge`. As you might guess from their names, the `override` option completely erases the current candidate configuration and replaces it with the contents of the file you specify. The `merge` function combines the file with the candidate configuration. Elements in the file that are not in the candidate are added. Variables in the candidate configuration that are not in the merging file are left unchanged. When an item is in both the merging file and the candidate configuration, the router uses the value specified in the file.

On the next router to be configured, we use the `load override` command to enter the common configuration elements:

```
[edit]
root# show
system {
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
    }
}

[edit]
root# load override common
load complete

[edit]
root# show
version 5.3R1.2;
system {
```

```
    host-name Shiraz;
    root-authentication {
        encrypted-password "$1$ZwtQb$cDpgAVcfDl/MLhTC1ZqQ4/"; # SECRET-DATA
    }
    login {
        user user {
            class super-user;
            authentication {
                encrypted-password "$1$/7NAOjwP$iwCrhoAqH38Kqh91AQFuY."; #
SECRET-DATA
            }
        }
    }
    services {
        telnet;
    }
    syslog {
        file messages {
            any notice;
            authorization info;
        }
    }
}
```

The previous output displays no hostname for the router. This is expected with a new router because no configuration has yet taken place.

Comparing the results of the `load override` command with the ***common*** file we saved earlier shows that only the elements detailed in the file are now in the candidate configuration. Specifically, the `user *` portion of the `syslog` directory is not in the ***common*** file and is removed. The `load merge` command provides you with different results:

```
[edit]
root# show
system {
    syslog {
        user * {
            any emergency;
        }
        file messages {
```

```
            any notice;
            authorization info;
        }
    }
}

[edit]
root# load merge common
load complete

[edit]
root# show
version 5.3R1.2;
system {
    host-name Shiraz;
    root-authentication {
        encrypted-password "$1$ZwtQb$cDpgAVcfDl/MLhTC1ZqQ4/"; # SECRET-DATA
    }
    login {
        user user {
            class super-user;
            authentication {
                encrypted-password "$1$/7NAOjwP$iwCrhoAqH38Kqh91AQFuY."; #
SECRET-DATA
            }
        }
    }
    services {
        telnet;
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
    }
}
```

The existing `user *` configuration remains as a result of the `load merge` command.

## Real World Scenario

### Cutting and Pasting Configuration Files

The ability to cut and paste portions of configuration files between routers is valuable when operating a network. Within the JUNOS software, you accomplish this with the `load merge terminal` command. In place of a file, the router expects you to enter keystrokes from the terminal directly. You may actually type portions of the configuration yourself, or more often paste text into the terminal window.

Suppose you have the following configuration within `[edit protocols]` on one of your routers:

```
[edit]
user@Shiraz# show | find protocols
protocols {
    bgp {
        group internal {
            type internal;
            local-address 192.168.16.1;
            neighbor 192.168.24.1;
            neighbor 192.168.12.1;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
}
```

You would like to copy the OSPF portion of the configuration to other routers in your network. To accomplish this, copy the output shown previously and place it into a text editor. Edit the text to look like the following:

```
protocols {
    ospf {
        area 0.0.0.0 {
            interface all;
```

```
            interface fxp0.0 {
                disable;
            }
        }
    }
}
```

You should ensure that all of the configuration hierarchy is accounted for—the router returns an error message if it does not receive the proper information. Type the **`load merge terminal`** command on your router and paste the text from your text editor into the router. After all of the text is entered, press Ctrl+D to close the paste window. You should see a `load complete` message if you are successful:

```
[edit]
user@Shiraz# load merge terminal
[Type ^D to end input]
protocols {
    ospf {
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
}
load complete

[edit]
user@Shiraz#
```

After some practice to fully understand the procedure, you'll find this to be a valuable tool for operating and configuring your network.

## Using the *commit* Command

We mentioned the `commit` command in the "The Candidate Configuration" section earlier in this chapter. Because no changes you make to the router become effective until you use this command, let's spend some time exploring its functionality.

Each time you commit your configuration, the router performs several tasks. The candidate configuration is examined for syntax and semantic problems and if any single problem exists, the candidate is not implemented. One example of a possible problem is referencing a routing

policy without creating that policy. (We discuss routing policies in Chapter 4, "Routing Policy.") If the candidate configuration possesses no errors, the router then implements the new configuration and makes changes to the operating environment as needed. Finally, the existing active configuration is saved on the router for future use.

You now decide to implement the changes to the router's configuration:

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

[edit]
user@Shiraz#
```

The `commit complete` message tells us that the process was successful. Notice a change of the router's hostname from `router` to `Shiraz`. We used the `commit` command from the top of the configuration hierarchy, but you can issue it from any level you wish. Unlike many other configuration mode commands that affect only the current configuration level and lower branches, the commit process always implements the entire configuration at once. Any errors encountered during a commit procedure result in no portion of the configuration changing.

Suppose that there was an error in the configuration we just committed. In that case, the router does not implement the changes we made and supplies an error message informing us of the problem:

```
[edit]
user@router# commit
Policy error: Policy Advertise-Routes referenced but not defined
error: configuration check-out failed

[edit]
user@router#
```

In addition to the `configuration check-out failed` message, we see that the router's hostname did not change. It appears that a policy called `Advertise-Routes` was referenced in the configuration without ever being created in the first place. We remove the offending policy and successfully commit the configuration.

```
[edit]
user@router# delete protocols ospf export Advertise-Routes

[edit]
user@router# commit
commit complete

[edit]
user@Shiraz#
```

### Command Options

The `commit` command has several options you may use to alter its operation. Let's view them on the Shiraz router:

```
[edit]
user@Shiraz# commit ?
Possible completions:
  <[Enter]>            Execute this command
  and-quit             Quit configuration mode if commit succeeds
  at                   Time at which to activate the configuration changes
  check                Check only, do not apply changes
  confirmed            Automatically rollback if not confirmed
  synchronize          Synchronize commit on both routing engines
  |                    Pipe through a command
[edit]
user@Shiraz# commit
```

The router always remains in configuration mode, by default, after committing the configuration. You may exit back to operational mode with the addition of the `and-quit` option:

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

user@Shiraz>
```


The router exits configuration mode only after a successful commit process. If any errors are encountered, they are reported and the router remains in configuration mode.

You can have the router verify the validity of the configuration without implementing the changes by using the `check` option. You might use this option after making a number of changes to the router and you want to be sure you have all of the required portions of the configuration in place. After running the syntax and semantic checks, the router does not implement the changes. You're either notified of a successful check or your errors are reported to you:

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

[edit]
user@Shiraz#
```

The syntax and semantic checks the router performs verify only that information is present in the configuration that allows the router to implement the candidate file. No verification is ever completed to see if the configuration actually does what you wanted it to do in the network; that is your job. If you are concerned that changes you made will either lock you out of your router or cause harm to the operation of the network, you should use the `confirmed` option. This option provides a safety net to the user in case of operational problems with your new configuration and is designed to allow the router to return to a working configuration automatically. After you issue the `commit confirmed` command, the router implements the changes you requested and starts a 10-minute timer. If you are happy with the new configuration, you must issue a normal `commit` command to stop the timer and end the operation of the `confirmed` option. If you don't stop the timer, the router automatically returns to the last operational configuration and implements those changes.

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

[edit]
user@Shiraz#
```

The output of the `commit confirmed` command is no different from that of a normal commit operation. The router either reports an error or displays the `commit complete` message. Additionally, you have the option of altering the timer value used with the `confirmed` option. The possible values range from 1 minute to 65,535 minutes (45 days, 12 hours, and 15 minutes).

The last option you may use with the `commit` command is `synchronize`. When you have a router with two Routing Engines installed, you can have the router apply the candidate configuration to both Routing Engines.

```
[edit]
user@router# commit synchronize
re0: configuration check succeeds
re1: configuration check succeeds
re0: commit complete
re1: commit complete

[edit]
user@Shiraz#
```

This option is helpful in the event of a Routing Engine failure; the backup Routing Engine now has the latest operational parameters in the network.

We discuss the fail-over operation of the Routing Engine in the section "Routing Engine Redundancy" later in this chapter.

### Restoring an Old Configuration

When the router commits a configuration, it also saves the existing configuration to a file. It is this saved file that the router uses during the `commit confirmed` process. This single file is not the only old configuration file saved, however. The JUNOS software saves up to nine previous configuration files for your use. The current active configuration is named `junper.conf` and is file number 0. The most recent active configuration is called `juniper.conf.1.gz` and is file number 1. This naming convention continues with each older file incrementing by 1 until the `juniper.conf.9.gz` file is reached.

You place one of these files into the candidate configuration with the `rollback` command. This command functions exactly like the `load override` command in that the existing candidate configuration is removed and the new file is put into its place. To actually implement the old configuration file, you must still issue the `commit` command to make the candidate configuration the new active configuration.

Suppose that we've altered the properties of the configuration on the Shiraz router. After committing the configuration, we realize that the new configuration is not performing as we wanted it to. So we load the most recent configuration and commit that change:

```
[edit]
user@Shiraz# rollback 1
load complete

[edit]
user@Shiraz# commit
commit complete

[edit]
user@Shiraz#
```

The router never automatically commits a rollback file for you. The only exception is a `commit confirmed` operation where the router issues both a `rollback 1` and a `commit` command.

## Manipulating Files on the Router

The JUNOS software stores multitudes of information in files on the router. Thus far, we've discussed configuration and rollback files, files we stored using the `save` command, and new versions of the JUNOS software itself. The router stores these files in various directories, including:

**`/config`** This directory is located on the router's internal flash drive. It contains the active configuration (`juniper.conf`) and rollback files 1, 2, and 3.

**`/var/db/config`** This directory is located on the router's hard drive and contains rollback files 4 through 9.

**`/var/tmp`** This directory is located on the router's hard drive. It holds core files from the various daemons on the Routing Engines. Core files are generated when a particular daemon crashes and are used by Juniper Networks engineers to diagnose the reason for failure.

**`/var/log`** This directory is located on the router's hard drive. It contains files generated by both the router's logging function as well as the `traceoptions` command.

**`/var/home`** This directory is located on the router's hard drive. It contains a subdirectory for each configured user on the router. These individual user directories are the default file location for many JUNOS software commands.

**`/altroot`** This directory is located on the router's hard drive and contains a copy of the root file structure from the internal flash drive. This directory is used in certain disaster-recovery modes where the internal flash drive is not operational.

**`/altconfig`** This directory is located on the router's hard drive and contains a copy of the `/config` file structure from the internal flash drive. This directory is also used in certain disaster-recovery modes where the internal flash drive is not operational.

You can view the router's directory structure as well as individual files by issuing the `file` command in operational mode:

```
user@Shiraz> file ?
Possible completions:
  compare              Compare files (local)
  copy                 Copy files (local or remote)
  delete               Delete files from the system (local)
  list                 List file information (local)
  rename               Rename files (local)
  show                 Display file contents (local)
```

As you can see, the `file` command gives you several options for manipulating files, but we'll focus on the `list` option here to see the directory structure of the router. The default directory for the `file list` command is the home directory of the user logged into the router. In fact, the user's home directory is the default directory for the majority of the JUNOS software commands requiring a filename. We currently have the following files in our home directory on the Shiraz router.

```
user@Shiraz> file list
.ssh/
common
```

You have the ability to view the contents of other file directories by specifying the directory structure:

```
user@Shiraz> file list /config
juniper.conf
juniper.conf.1.gz
juniper.conf.2.gz
juniper.conf.3.gz
```

The router's context-sensitive help system is also available to assist you in locating the desired directory:

```
user@Shiraz> file list /?
Possible completions:
  <[Enter]>            Execute this command
  <path>               Path to list
  /COPYRIGHT           Size: 4735, Last changed: Mar 31 2001
  /altconfig/          Last changed: Dec 11 2001
  /altroot/            Last changed: Dec 11 2001
  /bin/                Last changed: Aug 26 08:49:25
  /boot/               Last changed: Oct 03 16:27:55
  /config/             Last changed: Oct 03 16:27:56
  /dev/                Last changed: Sep 30 12:10:56
  /etc/                Last changed: Oct 03 16:27:56
  /kernel              Size: 9302545, Last changed: Apr 30 02:00:21
  /mnt/                Last changed: Dec 11 2001
  /modules/            Last changed: Aug 26 08:43:17
  /packages/           Last changed: Aug 26 08:49:45
  /proc/               Last changed: Oct 04 10:20:32
  /root/               Last changed: Aug 26 08:47:33
  /sbin/               Last changed: Aug 26 08:49:45
  /tmp/                Last changed: Oct 03 16:27:55
  /usr/                Last changed: Dec 11 2001
  /var/                Last changed: Dec 27 2001
user@Shiraz> file list /var/?
Possible completions:
  <[Enter]>            Execute this command
  <path>               Path to list
```

```
  /var/crash/          Last changed: Sep 16 09:03:30
  /var/cron/           Last changed: Dec 27 2001
  /var/db/             Last changed: Oct 03 16:27:56
  /var/etc/            Last changed: Oct 03 16:27:56
  /var/home/           Last changed: Oct 03 15:07:40
  /var/log/            Last changed: Oct 03 16:27:56
  /var/run/            Last changed: Oct 04 10:07:53
  /var/sw/             Last changed: Dec 27 2001
  /var/tmp/            Last changed: Sep 30 12:11:28
user@Shiraz> file list /var/log
messages

(Note: Information deleted for brevity)
```

## Routing Engine Redundancy

Certain Juniper Networks routers have the ability to contain two Routing Engines in the physical chassis. For the M-series platforms, the M20, M40e, and M160 support this configuration. Only one of the Routing Engines is considered the *master* at any point in time, and it controls the router's operations. The other Routing Engine, the *backup*, is available in the chassis to provide fail-over capability should the master cease to function.

By default, the router does not automatically enable the backup Routing Engine to assume the master role. You have to enable this functionality:

```
[edit chassis]
user@Shiraz# set redundancy failover on-loss-of-keepalives

[edit chassis]
lab@SanJose# show
redundancy {
    failover on-loss-of-keepalives;
}
```

The master and backup Routing Engines begin generating keepalive signals to each other. If the backup Routing Engine fails to receive keepalives for 20 seconds (a non-configurable timer), it enters a message in the `messages` log file. After 300 seconds, the default fail-over timer, the backup Routing Engine attempts to assume the master role for the router. When it succeeds, an alarm is generated to notify you that the master Routing Engine failed.

You can adjust the fail-over timer to between 2 and 10,000 seconds by using the `keepalive-time` command. Here, we've decided that the Shiraz router should use a 30-second timer value:

```
[edit chassis]
user@Shiraz# set redundancy keepalive-time 30
```

```
[edit chassis]
lab@SanJose# show
redundancy {
    failover on-loss-of-keepalives;
    keepalive-time 30;
}
```


Both the master and backup Routing Engines must be operating the same version of the JUNOS software for the redundancy process to function correctly.

# Packet Forwarding Engine Components

We now investigate the details of the Packet Forwarding Engine. This is a shorter discussion than the components of the Routing Engine, since your interaction with the Packet Forwarding Engine is through the CLI and the JUNOS software. The components of the Packet Forwarding Engine fall into two main subsets: the Embedded OS software operating the circuit boards themselves and the ASICs actually participating in packet forwarding. After covering the details of these two components, we discuss examples of the data packet flow through the forwarding plane.

## Embedded OS Software

As the router boots, the *Embedded OS software* (microcode) is downloaded from the Routing Engine to the CPUs on the circuit boards. Built by Juniper Networks engineers, the Embedded OS software contains a microkernel and individual threads that operate like the daemons on the Routing Engine. In stark contrast to the JUNOS software on the Routing Engine, the Embedded OS software on the Packet Forwarding Engine is fairly non-intelligent. It contains only enough capabilities to operate the control board, the FPCs, and the PICs. Perhaps most important, the Embedded OS software also begins the operation of the ASICs in the Packet Forwarding Engine.

## Application-Specific Integrated Circuits

Each circuit board in the Packet Forwarding Engine contains at least one ASIC, with some boards containing multiple chips. It is the interaction of these ASICs that provide the forwarding path through the router and supply the industry-leading forwarding performance of Juniper Networks routers.


For the remainder of this book, we focus only on the ASICs found within the M-series family of routers.

## PIC I/O Manager ASIC

Each PIC in the router contains an individual *PIC I/O Manager ASIC* that is unique to the specific media type on the PIC. For example, a PIC with Asynchronous Transfer Mode (ATM) interfaces has a different ASIC than a PIC with Synchronous Optical Network (SONET)/Synchronous Digital Hierarchy (SDH) interfaces. The requirement for this individuality arises from the tasks of the ASIC.

The PIC I/O Manager ASIC handles media-specific tasks such as verifying data-link framing, detecting link-level errors, and generating link-level alarms. Specialized functions such as ATM segmentation and reassembly (SAR) takes place on the PIC ASIC as well.

Generally speaking, the PIC I/O Manager ASIC is responsible for removing data packets from the physical media and placing data packets back on the physical media. It connects directly to the I/O Manager ASIC on the FPC containing the PIC.

## I/O Manager ASIC

Each FPC contains a single *I/O Manager ASIC* that connects to both the PIC I/O Manager ASIC and the Distributed Buffer Manager ASIC (which we discuss next) on the control board. The I/O Manager ASIC performs multiple functions on each data packet.

As a data packet enters the router, the I/O Manager ASIC verifies the integrity of both the Layer 2 and Layer 3 headers. Provided the data packet is valid, the ASIC removes the Layer 2 header and segments the packet into 64-byte units called a *J-cell*. The I/O Manager ASIC sends these J-cells to the Distributed Buffer Manager ASIC for storage in the shared memory pool.

Each I/O Manager ASIC in the router contributes memory to the shared memory packet storage on the router, controlled by the ASICs on the router's control board. The Distributed Buffer Manager ASIC instructs the I/O Manager ASIC to place and retrieve individual J-cells in the memory on its FPC.

On the outgoing side of the router, the I/O manager queues a special J-cell called the *result cell*. The result cell contains the next-hop information for the packet as well as other information about which queue to store the packet in. When the router is ready to send the data packet out an interface, the I/O manager ASIC receives all of the packet's J-cells from the packet buffer storage via the Distributed Buffer Manager ASIC. The I/O Manager ASIC re-forms the data packet and adjusts any protocol time-to-live (TTL) values before encapsulating the packet into the appropriate Layer 2 format. Finally, the packet is sent to the PIC I/O Manager ASIC for placement on the physical media.

## Distributed Buffer Manager ASIC

Each control board in the router contains two *Distributed Buffer Manager ASICs*. The ASICs connect to the I/O Manager ASIC on the FPC and to the Internet Processor ASIC, which is also on the control board. The ASIC is logically split into two components, each with an important function. One of the ASICs, which we refer to as the Inbound Distributed Buffer Manager ASIC, is responsible for handling inbound J-cells. Its partner, the Outbound Distributed Buffer Manager ASIC, handles outbound J-cells.

The two ASICs work in conjunction with each other to store and retrieve J-cells in the shared packet buffer pool. In addition, the Inbound Distributed Buffer Manager ASIC also generates a special J-cell called the *notification cell*. The notification cell contains information from the data packet, such as source and destination IP addresses, source and destination port numbers, the incoming interface on the router, Quality of Service (QoS) settings, and the existing protocol TTL value of the packet. The ASIC then sends the notification cell to the Internet Processor ASIC.

### Internet Processor ASIC

Every Juniper Networks router contains a single *Internet Processor ASIC* on the control board in the Packet Forwarding Engine. In many respects, the Internet Processor ASIC is the heart of the Packet Forwarding Engine. It is the only ASIC in the forwarding path that accesses the forwarding table, performs route lookups, and makes forwarding decisions. It receives notification cells from the Inbound Distributed Buffer Manager ASIC and transforms them into result cells after performing a route lookup. Additionally, the Internet Processor ASIC performs firewall packet filtering, enforces policy controls on data packets, and collects exception packets for transmission to the routing engine.


We discuss exception packets in the section "Exception Packets" later in this chapter.

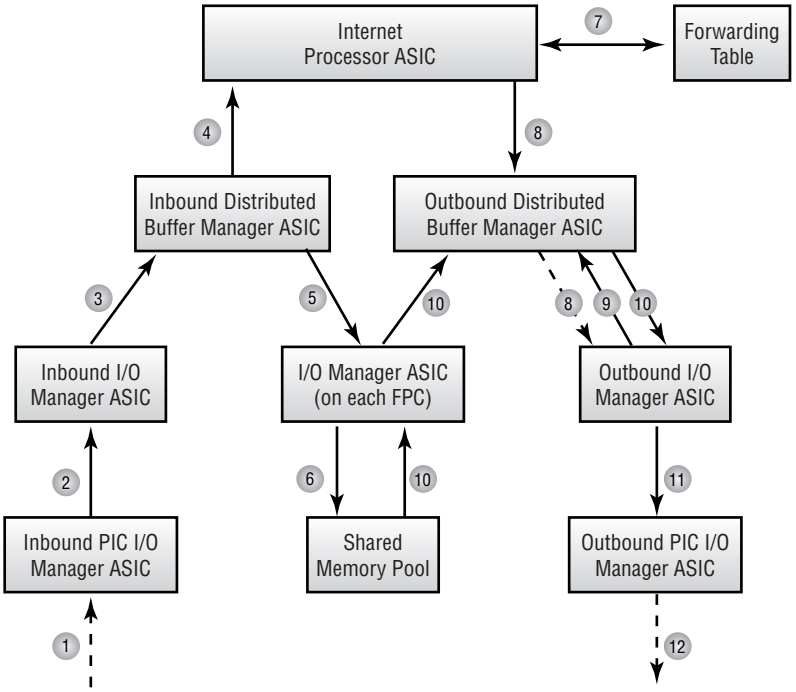
## Packet Flow

By understanding the functionality of the different ASICs in the Packet Forwarding Engine, you may already have a good idea of how a data packet flows through the forwarding path of the router. To provide a complete picture to use as a concise guide, we follow a unicast packet as it enters and then leaves the Packet Forwarding Engine. We then examine the differences in the forwarding path for multicast packets and discuss what exception packets are and how they are handled.

### Unicast Packets

Each unicast packet received on a router's interface is treated in a similar fashion. At a high level, the packet is stored in the shared memory pool, a route lookup is performed, and the packet is transmitted out one of the router's interfaces. Figure 1.4 displays a simplified view of the Packet Forwarding Engine ASICs and their representation to each other.

**FIGURE 1.4** Unicast packet flow




Let's use Figure 1.4 as a reference for the detailed steps of the unicast packet flow:

1. A data packet arrives on one of the router's interfaces. The PIC I/O Manager ASIC formulates the packet and performs link-layer error checking, if appropriate.
2. The PIC I/O Manager ASIC transmits the data packet, complete with Layer 2 and Layer 3 headers, to the I/O Manager ASIC on its FPC.
3. The I/O Manager ASIC verifies the integrity of the Layer 2 and Layer 3 headers. Provided a valid protocol packet remains, the I/O Manager ASIC removes the Layer 2 header and segments the data packet into 64-byte J-cells. It then sends those J-cells to the Inbound Distributed Buffer Manager ASIC.
4. The Inbound Distributed Buffer Manager ASIC begins to receive J-cells from the I/O Manager ASIC. The notification cell is built and is transmitted to the Internet Processor ASIC.
5. The J-cells that make up the data packet are stored in the shared memory pool. Each FPC supplies the physical components of the shared memory, and the Inbound Distributed Buffer Manager ASIC sends the packet's J-cells to all FPCs in the router on a round-robin basis.
6. The I/O Manager ASIC on each FPC receives the J-cells and stores them in its physical memory as instructed by the Inbound Distributed Buffer Manager ASIC.

7. While the J-cells are being stored in memory, the Internet Processor ASIC receives the notification cell and performs a route lookup in the forwarding table. The next-hop router along the path of the route and the outgoing interface on the router is determined. This next-hop information is stored in the notification cell, which now becomes the result cell.
8. The Internet Processor ASIC sends the result cell to the Outbound Distributed Buffer Manager ASIC, which examines the cell to locate the outgoing interface. The result cell is then sent to the appropriate FPC for queuing and transmission.
9. The I/O Manager ASIC queues the result cell and applies appropriate queuing mechanisms. When the result cell reaches the head of the queue, the I/O Manager ASIC requests the packet's J-cells from the Outbound Distributed Buffer Manager ASIC.
10. The Outbound Distributed Buffer Manager ASIC copies the J-cells from the packet storage buffer and sends them to the I/O Manager ASIC on the outgoing FPC.
11. The I/O Manager ASIC re-forms the data packet into a single unit and alters any protocol TTL values. The ASIC then appends the appropriate Layer 2 header information and sends the packet to the PIC I/O Manager ASIC.
12. The PIC I/O Manager ASIC performs any link-layer duties, if required, and transmits the data packet out the router's interface.

## Multicast Packets

A Juniper Networks router handles multicast data packets in a very similar fashion to unicast packets. There is only one major difference between the two, so we won't repeat the packet flow steps in detail here.

Refer back to Figure 1.4 and the steps outlined in the previous section, and focus on steps 7 and 8. When the Internet Processor ASIC performs its route lookup on a multicast packet, it often finds multiple next-hop interfaces in the forwarding table. Information about all the outgoing interfaces is placed in the result cell and sent to the Distributed Buffer Manager ASIC. This ASIC examines the result cell and finds several outgoing interfaces. It generates a copy of the result cell for each interface and sends those cells to the appropriate I/O Manager ASICs on the FPCs. The queuing and transmission of the multicast packets at this point then follows the unicast packet steps outlined previously.

## Exception Packets

The Packet Forwarding Engine can't process some data packets in your network in its normal fashion. A prime example of these packets is routing protocol updates addressed to the router itself. There is no outgoing interface for these packets; they should be sent to the Routing Engine instead. The CPU on the router's control board handles this type of traffic, called an *exception packet*.

Other forms of exception packets include:

- Packets addressed to the router, such as ICMP pings, Telnet, and SSH traffic
- Packets requiring the generation of an ICMP error message, including traceroute responses and destination unreachable messages
- Packets containing an IP Options field

The control board CPU handles different types of exception traffic differently. For example, routing protocol updates are sent to the Routing Engine over the `fxp1` interface. Local delivery packets (Telnet, for example) and IP Options packets are sent to the Routing Engine as well. The control board CPU itself generates any ICMP error messages and sends them to the appropriate IP source address.

# Summary

This chapter discussed the basic router design of a Juniper Networks router. We examined the basic functionality and components of both the Routing Engine and the Packet Forwarding Engine.

We further explored the Routing Engine with a look at the JUNOS software architecture, its naming convention, and operational parameters. Next, we discussed the router's CLI by examining the operational and configuration modes of the router. We saw how to navigate through the CLI, use the context-sensitive help system, understand the command completion process, and modify the output of commands. We then discussed the differences between the candidate and active configurations, including a look at the router's rollback functionality. We also examined the various ways to use the `commit` command and where the router stores configuration and user files.

We concluded the chapter by looking at the Packet Forwarding Engine. This discussion centered on how the ASICs and control boards forward user data packets through the router. We examined the handling of both unicast and multicast packets. Finally, we defined an exception packet and explained how the router handles them.

# Exam Essentials

**Understand the basic functions of the Routing Engine and the Packet Forwarding Engine.** The Routing Engine is the intelligence of the router. It operates the routing protocols and builds a routing and forwarding table. The forwarding table is copied to the Packet Forwarding Engine, where the actual transmission of user data packets is handled.

**Be able to identify the JUNOS software boot locations and the default boot sequence.** The JUNOS software is stored on the internal flash drive, the internal hard drive, and the removable flash media. When the router begins to boot, the removable media is checked first, followed by the internal flash drive, and finally the internal hard drive.

**Understand the JUNOS software commands associated with configuration files.** You may save the router's configuration to the hard drive with the `save` command. The `load` command restores files to the candidate configuration. The candidate configuration becomes the active configuration with the `commit` command. You can easily return to a previous configuration with the `rollback` command.

**Be able to identify the ASICs used in the Packet Forwarding Engine.** There are four main ASICs used in the Packet Forwarding Engine: the Internet Processor ASIC, the Distributed Buffer Manager ASIC, the I/O Manager ASIC, and the PIC I/O Manager ASIC.

**Be able to describe the flow of a packet through the Packet Forwarding Engine.** A packet is received on an interface and is segmented into J-cells by the I/O Manager ASIC. The Distributed Buffer Manager ASIC stores the packet in the shared memory pool. The Internet Processor ASIC performs a route lookup and sends the result to the Distributed Buffer Manager ASIC, which forwards it to the outgoing I/O Manager ASIC. After queuing the packet, the I/O Manager ASIC receives the J-cells from the memory pool and re-forms the packet. It is sent to the outgoing PIC I/O Manager ASIC for transmission into the network.

**Understand what an exception packet is and how the router handles those packets.** An exception packet could be a routing protocol update, a locally addressed packet, or a packet requiring the generation of an ICMP error message. The CPU on the router's control board handles these exception packets and performs the appropriate action.

# Key Terms

Before you take the exam, be certain you are familiar with the following terms:

active configuration
application-specific integrated circuits (ASICs)
backup
candidate configuration
Chassis Daemon (chassid)
command completion
command-line interface (CLI)
configuration mode
context-sensitive help
Device Control Daemon (dcd)
Distributed Buffer Manager ASIC
editor macros (Emacs)
Embedded OS software
exception packet
Flexible PIC Concentrator (FPC)
Forwarding Engine Board (FEB)
forwarding table
I/O Manager ASIC
Internet Processor ASIC
`jbase`
`jbundle`
J-cell
`jcrypto`
`jdocs`
`jkernel`
`jpfe`
`jroute`
kernel
Management Daemon (mgd)
master
Memory Mezzanine Board (MMB)
notification cell
operational mode
package
Packet Forwarding Engine
Packet Forwarding Engine Daemon (pfed)
Physical Interface Card (PIC)
PIC I/O Manager ASIC
result cell
Routing Engine
Routing Protocol Daemon (rpd)
routing table
Switching and Forwarding Module (SFM)
System Control Board (SCB)
System Switching Board (SSB)

# Review Questions

1. What are the functions of the Routing Engine? (Choose three.)
   - **A.** Operates routing protocols
   - **B.** Segments data packets into J-cells
   - **C.** Loads the JUNOS software
   - **D.** Controls the CLI
2. Which router component is responsible for creating the forwarding table?
   - **A.** Packet Forwarding Engine
   - **B.** Routing Engine
   - **C.** Flexible PIC Controller
   - **D.** Physical Interface Card
3. The PIC I/O Manager ASIC is responsible for what function?
   - **A.** Creating J-cells
   - **B.** Performing route lookups
   - **C.** Transmitting packets
   - **D.** Storing packets in memory
4. The Internet Processor ASIC is responsible for what function?
   - **A.** Creating J-cells
   - **B.** Performing route lookups
   - **C.** Transmitting packets
   - **D.** Storing packets in memory
5. The I/O Manager ASIC is responsible for what function?
   - **A.** Creating J-cells
   - **B.** Performing route lookups
   - **C.** Transmitting packets
   - **D.** Creating notification cells
6. The Distributed Buffer Manager ASIC is responsible for what function?
   - **A.** Creating J-cells
   - **B.** Performing route lookups
   - **C.** Transmitting packets
   - **D.** Storing packets in memory

7. A unicast packet is flowing through the Packet Forwarding Engine. Which ASIC receives the packet after the incoming PIC I/O Manager ASIC performs its functions?
   - **A.** Incoming I/O Manager ASIC
   - **B.** Outgoing I/O Manager ASIC
   - **C.** Incoming Distributed Buffer Manager ASIC
   - **D.** Outgoing Distributed Buffer Manager ASIC

8. What component of the router is responsible for handling exception packets?
   - **A.** Internet Processor ASIC
   - **B.** Switching control board CPU
   - **C.** Routing Engine
   - **D.** Flexible PIC Controller CPU

9. Which types of packets are considered exception packets? (Choose two.)
   - **A.** IP packets with TTL=1
   - **B.** HTTP packets
   - **C.** SMTP packets
   - **D.** Routing protocol updates

10. Which JUNOS software daemon is responsible for operating the CLI?
    - **A.** chassisd
    - **B.** rpd
    - **C.** mgd
    - **D.** dcd

11. Which JUNOS software daemon is responsible for controlling the routing protocols?
    - **A.** chassisd
    - **B.** rpd
    - **C.** mgd
    - **D.** dcd

12. When issued from the top of the configuration hierarchy, which command creates a file called ***saved-file*** that contains the entire candidate configuration?
    - **A.** `file save saved-file`
    - **B.** `save saved-file`
    - **C.** `run file save saved-file`
    - **D.** `run save saved-file`

13. Which command places the `juniper.conf.5.gz` file in the candidate configuration?
    A. `rollback 5`
    B. `run rollback 5`
    C. `load override juniper.conf.5.gz`
    D. `load merge juniper.conf.5.gz`

14. Where does the router store each user's home directory?
    A. `/var/db/config`
    B. `/var/log`
    C. `/var/home`
    D. `/var/usr`

15. What is the primary boot media for the JUNOS software?
    A. Removable media
    B. Internal flash drive
    C. External flash drive
    D. Internal hard drive

16. What is the secondary boot media for the JUNOS software?
    A. Removable media
    B. Internal flash drive
    C. External flash drive
    D. Internal hard drive

17. Which command loads a new version of the JUNOS software into the internal flash drive?
    A. `load upgrade` ***`filename`***
    B. `request system software add` ***`filename`***
    C. `load software` ***`filename`***
    D. `request system load` ***`filename`***

18. Which Emacs keystroke takes the cursor to the beginning of the command line?
    A. Ctrl+A
    B. Ctrl+D
    C. Ctrl+E
    D. Ctrl+W

19. Which command allows you to paste text directly into the candidate configuration?
    A. `load override`
    B. `load override filename`
    C. `load merge`
    D. `load merge terminal`

20. When you're committing your configuration, what command allows the router to automatically return to a previous configuration?
    A. `commit`
    B. `commit and-quit`
    C. `commit check`
    D. `commit confirmed`

# Answers to Review Questions

1. A, C, D. The Routing Engine performs multiple functions, including operating the routing protocols on the router, loading the JUNOS software, and controlling the CLI. The Packet Forwarding Engine controls packet forwarding.

2. B. The Routing Engine builds the master routing table, selects the best path to each route, and places those next hops into the forwarding table.

3. C. The PIC I/O Manager ASIC is responsible for receiving and transmitting data packets from the physical media connected to the PIC.

4. B. The Internet Processor ASIC consults the forwarding table on the control board to determine the next-hop router along the path to the destination.

5. A. The I/O manager ASIC is responsible for multiple functions in the router. One of those is the creation of J-cells from the original data packet.

6. D. The primary role of the Distributed Buffer Manager ASIC is storing and retrieving J-cells from the packet storage buffer.

7. A. After receiving the packet from the physical media and performing any link-layer functions, the incoming PIC I/O Manager ASIC sends the packet to the incoming I/O Manager ASIC on its FPC.

8. B. The CPU on the router's control board is responsible for handing exception packets. Some of those exception packets might reach the Routing Engine.

9. A, D. Routing protocol updates and packets requiring an ICMP error message (TTL = 1) are considered exception packets. A Juniper Networks router does not communicate using the HTTP or SMTP protocols. Therefore, these packets must be transiting the router and are handled by the Packet Forwarding Engine.

10. C. The Management Daemon (mgd) is responsible for controlling the CLI process.

11. B. The Routing Protocol Daemon (rpd) is responsible for all routing protocol activity on the router.

12. B. The `save` command takes portions of the candidate configuration and places them in a file you specify. When used from the top of the hierarchy, this process saves the entire candidate configuration.

13. A. Only the `rollback 5` command places the fifth rollback file in the candidate configuration. Options C and D will look for the `juniper.conf.5.gz` file in the user's home directory, where it is not stored by default.

14. C. Each user configured on the router receives his or her own home directory in the `/var/home` section of the hard drive.

15. B. The router's internal flash drive is the primary boot location for the JUNOS software.

**16.** D. The router's internal hard drive is the secondary boot location for the JUNOS software.

**17.** B. The `request system software add` ***`filename`*** command loads a copy of the JUNOS software onto the router's flash drive.

**18.** A. To reach the beginning of the command line, use the Ctrl+A keystroke. Ctrl+E takes you to the end and Ctrl+W deletes the previous word. Ctrl+D closes your terminal during a `load merge terminal` operation.

**19.** D. The `load merge terminal` command allows you to cut and paste configuration directly into the router.

**20.** D. The `commit confirmed` command allows the router to return to the previous configuration automatically if you don't issue a regular `commit` within the default 10-minute timer.