Chapter 1
Configuring IP

This chapter describes how to configure Internet Protocol (IP) routing on your E-series router.

Overview

TCP/IP is a suite of data communications protocols. Two of the more important protocols in the suite are the Transmission Control Protocol (TCP) and the Internet Protocol (IP).

IP provides the basic packet delivery service for all TCP/IP networks. IP is a *connectionless* protocol, which means that it does not exchange control information to establish an end-to-end connection before transmitting data. A *connection-oriented* protocol exchanges control information with the remote computer to verify that it is ready to receive data before sending it.

IP relies on protocols in other layers to establish the connection if connection-oriented services are required and to provide error detection and error recovery. IP is sometimes called an unreliable protocol, because it contains no error detection or recovery code.

IP Packets

A *packet* is a block of data that carries with it the information necessary to deliver it to a destination address. A *packet-switching network* uses the addressing information in the packets to switch packets from one physical network to another, moving them toward their final destination. Each packet travels the network independently of any other packet. The *datagram* is the packet format defined by IP.

IP Functions

Some of the functions IP performs include:

- Moving data between the network access layer and the host-to-host transport layer
- Routing datagrams to remote hosts
- Fragmenting and reassembling datagrams

Moving Data Between Layers

When IP receives a datagram that is addressed to the local host, it must pass the data portion of the datagram to the correct host-to-host transport layer protocol. IP uses the *protocol number* in the datagram header to select the transport layer protocol. Each host-to-host transport layer protocol has a unique protocol number that identifies it to IP.

Routing Datagrams to Remote Hosts

Internet gateways are commonly referred to as IP routers because they use IP to route packets between networks. In traditional TCP/IP terms, there are only two types of network devices: gateways and hosts. Gateways forward packets between networks, and hosts do not. However, if a host is connected to more than one network (called a *multihomed host*), it can forward packets between the networks. When a multihomed host forwards packets, it acts like any other gateway and is considered to be a gateway.

Fragmenting and Reassembling Datagrams

As a datagram is routed through different networks, it may be necessary for the IP module in a gateway to divide the datagram into smaller pieces. A datagram received from one network may be too large to be transmitted in a single packet on a different network. This condition occurs only when a gateway interconnects dissimilar physical networks.

Each type of network has a maximum transmission unit (MTU) that determines the largest packet it can transfer. If the datagram received from one network is longer than the other network's MTU, it is necessary to divide the datagram into smaller fragments for transmission in a process called *fragmentation*. See *Fragmentation* on page 25.

IP Layering

TCP/IP is organized into four conceptual layers (as shown in Figure 1).

Figure 1: TCP/IP Conceptual Layers

Network Interface Layer

The network interface layer is the lowest level of the TCP/IP protocol stack. It is responsible for transmitting datagrams over the physical medium to their final destinations.

Internet Layer

The Internet layer is the second level of the TCP/IP protocol stack. It provides host-to-host communication. In this layer, packets are encapsulated into datagrams, routing algorithms are run, and the datagram is passed to the network interface layer for transmission on the attached network.

Transport Layer

The transport layer is the third level of the TCP/IP protocol stack. It is responsible for providing communication between applications residing in different hosts. By placing identifying information in the datagram (such as socket information), the transport layer enables process-to-process communication.

The transport layer provides either a reliable transport service (TCP) or an unreliable service (User Data Protocol). In a reliable delivery service, the destination station acknowledges the receipt of a datagram.

Application Layer

The application layer is the fourth and highest level of the TCP/IP protocol stack. Some applications that run in this layer are:

- Telnet
- File Transfer Protocol (FTP)
- Simple Mail Transfer Protocol (SMTP)
- Simple Network Management Protocol (SNMP)
- Domain Name System (DNS)

Platform Considerations

For information about modules that support IP on the ERX-7xx models, ERX-14xx models, and the ERX-310 router:

- See *ERX Module Guide, Table 1, Module Combinations* for detailed module specifications.
- See *ERX Module Guide, Appendix A, Module Protocol Support* for information about the modules that support IP.

For information about modules that support IP on the E120 router and the E320 router:

- See *E120 and E320 Module Guide, Table 1, Modules and IOAs* for detailed module specifications.
- See *E120 and E320 Module Guide, Appendix A, IOA Protocol Support* for information about the modules that support IP.

References

For more information about IP, consult the following resources:

- RFC 768—User Datagram Protocol (August 1980)
- RFC 791—Internet Protocol DARPA Internet Program Protocol Specification (September 1981)
- RFC 792—Internet Control Message Protocol (September 1981)
- RFC 793—Transmission Control Protocol (September 1981)
- RFC 854—Telnet Protocol Specification (May 1983)
- RFC 919—Broadcasting Internet Datagrams (October 1984)
- RFC 922—Broadcasting Internet Datagrams in the Presence of Subnets (October 1984)

- RFC 950—Internet Standard Subnetting Procedure (August 1985)
- RFC 1112—Host Extensions for IP Multicasting (August 1989)
- RFC 1122—Requirements for Internet Hosts—Communication Layers (October 1989)
- RFC 1812—Requirements for IP Version 4 Routers (June 1995)
- RFC 3419—Textual Conventions for Transport Addresses (December 2002)
- *JUNOSe Release Notes*, *Appendix A*, *System Maximums*—Refer to the Release Notes corresponding to your software release for information about maximum values.

IP Features

The E-series router supports the following IP features:

- Internet Control Message Protocol (ICMP)
- Traceroute
- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)
- Classless interdomain routing (CIDR)
- Maximum transmission unit (MTU)
- Support for simultaneous multiple logical IP stacks on the same router
- Flexible IP address assignment to support any portion of a physical interface (for example, a channel or circuit), exactly one physical interface, or multilink PPP interfaces
- Packet segmentation and reassembly
- Loose source routing to specify the IP route
- Strict source routing to specify the IP route for each hop
- Record route to track the route taken
- Internet timestamp
- Broadcast addressing, both limited broadcast and directed broadcast
- Support for 32,000 discrete, simultaneous IP interfaces per router to support thousands of logical connections
- Capability of detecting and reporting changes in the up or down state of any IP interface

- IP policy support. See *JUNOSe IP Services Configuration Guide, Chapter 1, Configuring Routing Policy*, for more information about policy configuration.
- Indirect next hops
- IP tunnel routing tables

IP Addressing

This section provides an overview of IP addressing in general and includes a discussion of CIDR, which your router fully supports.

Physical and Logical Addresses

Physical node addresses are used at the network access layer to identify physical devices in a network. For example, each Ethernet controller comes from the manufacturer with a physical address, called a MAC address.

IP implements a system of logical host addresses called IP addresses. The IP addresses are used by the internetwork and higher layers to identify devices and to perform internetwork routing. The Address Resolution Protocol (ARP) enables IP to identify the physical (MAC) address that matches a given IP address. You can use ARP only on Ethernet and bridged IP 1483 interfaces.

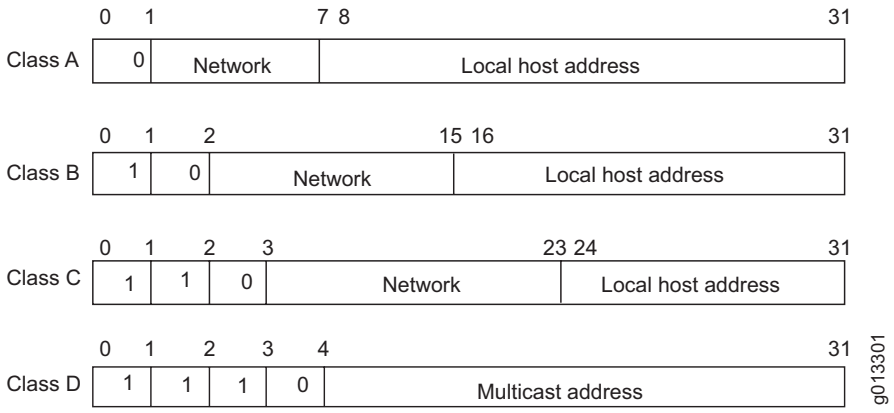
IP is used by all protocols in the layers above and below it to deliver data. This means that all TCP/IP data flows through IP when it is sent and received, regardless of its final destination.

Internet Addresses

Internet addressing uses a 32-bit address field. The bits in this address field are numbered 0 to 31. The 32-bit address field consists of two parts: a network number and a host number whose boundaries are defined based on the class of IP address. Hosts attached to the same network must share a common prefix designating their network number.

Four types of IP classes lend themselves to different network configurations, depending on the desired ratio of networks to hosts. Figure 2 shows the format of IP address classes.

Figure 2: IP Address Classes



- Class A—The leading bit is set to 0, a 7-bit number, and a 24-bit local host address. Up to 125 class A networks can be defined, with up to 16,777,214 hosts per network.
- Class B—The two highest-order bits are set to 1 and 0, a 14-bit network number, and a 16-bit local host address. Up to 16,382 class B networks can be defined, with up to 65,534 hosts per network.
- Class C—The three leading bits are set to 1, 1, and 0, a 21-bit network number, and an 8-bit local host address. Up to 2,097,152 class C networks can be defined, with up to 254 hosts per network.
- Class D—The four highest-order bits are set to 1, 1, 1, and 0. Class D is used as a multicast address.

Subnetwork Mask Format Options

Most commands allow you to specify IPv4 subnetwork masks in one of two ways: dotted decimal or prefix length notation.

NOTE: Protocol commands that use a reverse mask format (for example, RIP) cannot use the prefix notation format. Use the CLI help to verify if a command supports the /N prefix notation.

Dotted decimal notation expresses IP addresses and masks in dotted quads - four octets separated by dots (A.B.C.D). In this format, each octet in the address or mask is represented as a decimal number and the dots are used as octet separators.

For example, an IP address and subnetwork mask in dotted decimal notation would appear as follows:

10.10.24.6 255.255.0.0

Prefix length notation (often called network prefix format) allows for more efficient allocation of IP addresses than the old Class A, B, and C address scheme. The prefix length is the number of leftmost contiguous bits equal to 1 in the subnetwork mask. This format appears immediately following the dotted decimal IP address using a /N format.

NOTE: You can issue the network prefix with or without a space between the IP address and the network prefix (/N).

For example, the same IP address and subnetwork mask mentioned above would appear as follows using /N format:

10.10.24.6/16
or
10.10.24.6 /16

The following sections describe each subnetwork mask addressing method in more detail.

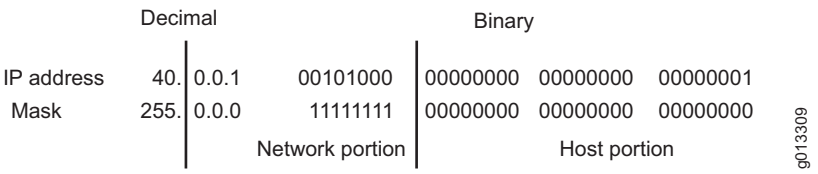
Subnet Addressing

A subnet is a subset of a class A, B, or C network. Subnets cannot be used with class D (multicast) addresses.

A network mask is used to separate the network information from the host information about an IP address. Figure 3 shows the network mask 255.0.0.0 applied to network 10.0.0.0. The mask in binary notation is a series of 1s followed by a series of contiguous 0s. The 1s represent the network number; the 0s represent the host number. The sample address splits the IP address 10.0.0.1 into a network portion of 10 and a host portion of 0.0.1.

NOTE: The router supports a 31-bit mask on point-to-point links. This means that the IP address 1.2.3.4 255.255.255.254 is valid. A point-to-point link in which only one end supports the use of 31-bit prefixes may not operate correctly.

Figure 3: Basic Network Masking



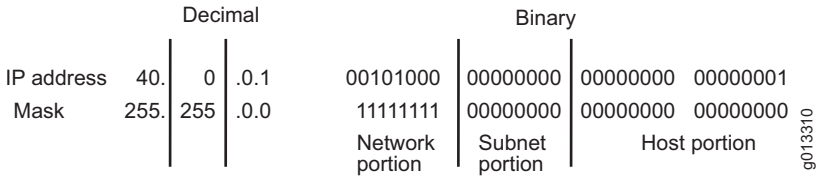
Classes A, B, and C have the following *natural masks,* which define the network and host portions of each class:

- Class A natural mask 255.0.0.
- Class B natural mask 255.255.0.0
- Class C natural mask 255.255.255.0

The use of masks can divide networks into subnetworks by extending the network portion of the address into the host portion. Subnetting increases the number of subnetworks and reduces the number of hosts.

For example, a network of the form 10.0.0.0 accommodates one physical segment with about 16 million hosts on it. Figure 4 shows how the mask 255.255.0.0. is applied to network 10.0.0.0. The mask divides the IP address 10.0.0.1 into a network portion of 10, a subnet portion of 0, and a host portion of 0.1. The mask has borrowed a portion of the host space and has applied it to the network space. The network space of the class 10 has increased from a single network 10.0.0.0 to 256 subnetworks, ranging from 10.0.0.0 to 10.255.0.0. This process decreases the number of hosts per subnet from 16,777,216 to 65,536.

Figure 4: Subnetting



Classless Addressing with CIDR

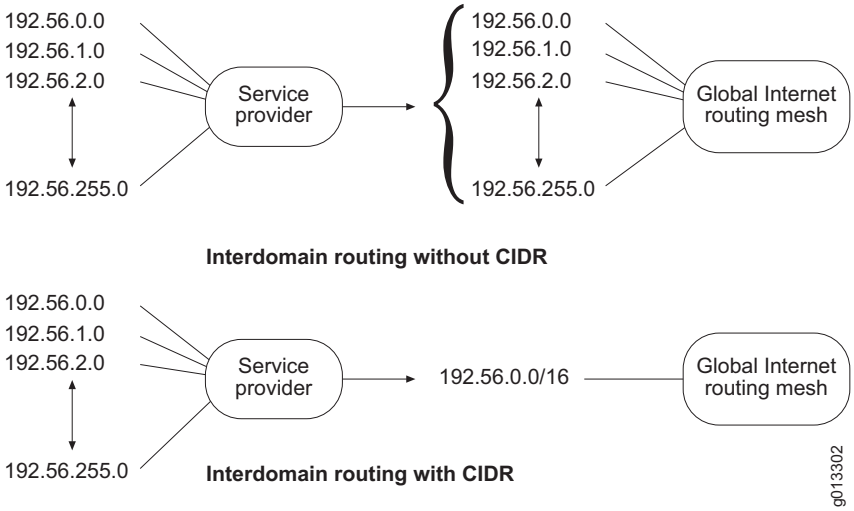
Classless interdomain routing (CIDR) is a system of addressing that improves the scaling factor of routing in the Internet. CIDR does not use an implicit mask based on the class of network. In CIDR, an IP network is represented by a prefix, which is an IP address and an indication of the leftmost contiguous significant bits within this address.

For example, without CIDR, the class C network address 192.56.0.0 would be an illegal address. With CIDR, the address becomes valid with the notation: 192.56.0.0/16. The /16 indicates that 16 bits of mask are being used (counting from the far left). This would be similar to an address 198.32.0.0. with a mask of 255.255.0.0.

A network is called a *supernet* when the prefix boundary contains fewer bits than the network's natural mask. For example, a class C network 192.56.10.0 has a natural mask of 255.255.255.0. The representation 192.56.0.0/16 has a shorter mask than the natural mask (16 is less than 24), so it is a supernet.

Figure 5 shows how CIDR can reduce the number of entries globally in Internet routing tables. A service provider has a group of customers with class C addresses that begin with 192.56. Despite this relationship, the service provider announces each of the networks individually into the global Internet routing mesh.

Figure 5: Routing With and Without CIDR



Adding and Deleting Addresses

This section provides information about adding or deleting IP addresses.

Multinetting is adding more than one IP address to an IP interface—that is, a primary address and one or more secondary addresses.

To make an interface unnumbered, see *Setting Up an Unnumbered Interface* on page 39.

Adding a Primary Address

- The primary address must be the first address added to the interface.
- Adding a new primary address overwrites the existing primary address.
- You can change a secondary address to be the primary address on an interface only via SNMP.
- An unnumbered address is always the primary address; adding an unnumbered address, therefore, overwrites any other numbered address.

Deleting a Primary Address

- You must always remove the primary address from an interface last.
- You cannot delete the primary address if the interface still has assigned secondary addresses.

Adding a Secondary (Multinet) Address

- You cannot add a secondary address until you add the primary address.
- You cannot add a secondary address to bridged Ethernet interfaces.
- You cannot change a primary address to a secondary address.
- An interface can have multiple secondary addresses.

Deleting a Secondary Address

- You must delete secondary addresses before deleting the primary address.

ip address Command

Use the following command to add addresses to or delete addresses from an interface:

ip address

- Use to add a primary address or to add secondary addresses to an interface.
- To add multiple addresses to a single IP interface, use the **secondary** keyword. (Remember, if you add an address using the **ip address** command and do not include the **secondary** keyword, the new address becomes the primary address.)
- You can specify the subnetwork mask value in either dotted decimal or prefix length notation.
- Example—Adds a primary address (192.168.2.77) and two secondary addresses (172.31.7.22 and 10.8.7.22); the Fast Ethernet interface now has addresses in three networks.

  host1(config)#**interface fastEthernet 0/0**
  host1(config-if)#**ip address 192.168.2.77 255.255.255.0**
  host1(config-if)#**ip address 172.31.7.22 255.255.255.0 secondary**
  host1(config-if)#**ip address 10.8.7.22 255.255.255.0 secondary**

NOTE: You can use this command in Interface Configuration mode, Subinterface Configuration mode, or Profile Configuration mode.

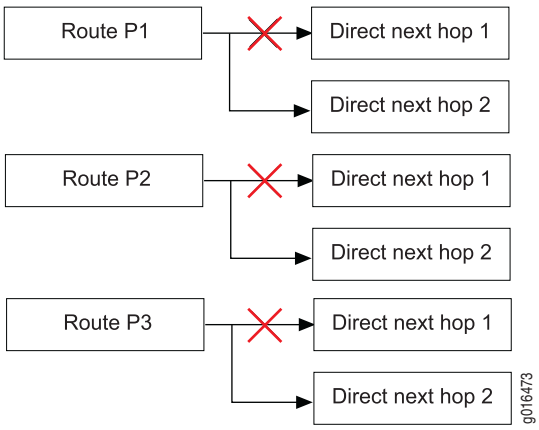
- Use the **no** version to remove an IP address. If you remove a primary IP address, IP processing is disabled on the interface.

Indirect Next-Hop Support

The router uses indirect next hops to promote faster network convergence (for example, in BGP networks) by decreasing the number of routing table changes required when a change in the network topology occurs.

Direct next-hops point routes in the routing table toward individual, direct next-hop connections. (See Figure 6.)

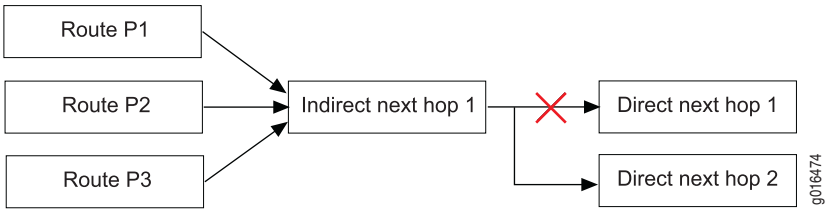
Figure 6: Direct Next Hops



Indirect next hops enable multiple routes in the routing table to point to a single next hop, thereby accelerating convergence. (See Figure 7.)

NOTE: Indirect next hops are not limited to any number of levels. In other words, an indirect next hop can point to a direct next hop or another indirect next hop.

Figure 7: Indirect Next Hops



By using indirect next hops, if a topology change occurs in the network, only the indirect next hop is modified in the routing table, decreasing the number of state changes required to achieve convergence.

Before You Configure IP

Before you configure IP, created lower-layer interfaces over which IP traffic flows.

For example, to configure an ATM interface:

```
host1(config)#interface atm 1/0
host1(config-if)#atm sonet stm-1
host1(config-if)#no loopback
host1(config-if)#atm clock internal chassis
host1(config-if)#interface atm 1/0.10
host1(config-if)#atm pvc 10 0 20 aal5snap
```

Refer to the appropriate chapters for information about configuring a specific type of interface.

NOTE: If you choose to configure VRRP, we recommend that you complete all IP address configurations before you configure VRRP. See *JUNOSe IP Services Configuration Guide, Chapter 14, Configuring VRRP*, for additional information.

Creating a Profile

You can configure an IP interface dynamically by creating a profile. A profile is a set of characteristics that acts as a pattern that can be dynamically assigned to an IP interface. You can manage a large number of IP interfaces efficiently by creating a profile with a specific set of characteristics. In addition, you can create a profile to assign an IP interface to a virtual router.

A profile can contain one or more of the following characteristics:

- access-route—Enables the creation of host access routes on an interface
- address—Configures an IP address on an interface
- auto-configure—Configures the interface for auto-configure mode
- auto-detect—Configures the interface for auto-detect mode
- directed-broadcast—Enables directed broadcast forwarding
- filter-options-all—Enables filtering of packets with IP options on an interface
- igmp—Configures an IGMP interface
- ignore-df-bit—Specifies that the don't-fragment bit is ignored
- inactivity-timer—Configures inactivity time for IP interfaces
- inspection—Associates an inspection list to the interface for firewalling
- mtu—Configures the maximum transmission unit for a network

- nat—Configures the interface as inside or outside for Network Address Translation (NAT)
- policy—Assigns a policy to the ingress or egress of an interface
- redirects—Enables transmission of ICMP redirect messages
- route-maps—Configures the interface for route-map processing
- source address validation—Verifies that a packet has been sent from a valid source address
- tcp adjust-mss—Adjusts maximum packet sizes on TCP connections when path MTU detection is not sufficient
- unnumbered—Configures IP on this interface without a specific address
- virtual-router—Specifies a virtual router to which interfaces created by this profile will be attached

Use the **profile** command from Global Configuration mode to create or edit a profile. See *JUNOSe Link Layer Configuration Guide, Chapter 15, Configuring Dynamic Interfaces* for information about creating profiles and on other characteristics that can be applied to the profile.

```
host1(config)#profile acton
host1(config-profile)#ip virtual-router warf
host1(config-profile)#ip unnumbered atm 3/0
```

ip access-routes

- Use to enable an access route in a profile.
- Example

```
host1(config)#profile foo
host1(config-profile)#ip access-routes
```

- Use the **no** version to remove the access route.

ip address

- Use to assign an IP address to a profile.
- You must first specify the layer 2 encapsulation before you can set the IP address for serial interfaces.
- Example

```
host1(config-if)#ip address 192.56.32.2 255.255.255.0
```

- Use the **no** version to remove the IP address assigned to the profile.

ip directed-broadcast

- Use to enable a directed broadcast address in a profile.
- Example

  host1(config-if)#**ip directed-broadcast**

- Use the **no** version to remove the directed broadcast address from the profile.

ip inspection

- Use to associate an inspection list to the inbound or outbound side of the IP interface.
- Example

  host1(config-profile)#**ip inspection list1**

- Use the **no** version to remove the inspection list association to this interface.

ip mtu

- Use to assign the MTU size sent on an IP interface.
- Example

  host1(config-if)#**ip mtu 5000**

- Use the **no** version to remove the assignment from the profile.

ip redirects

- Use to enable the sending of redirect messages if the software is forced to resend a packet through the same interface on which it was received.
- Example

  host1(config-if)#**ip redirects**

- Use the **no** version to remove the assignment from the profile.

ip tcp adjust-mss

- Use to modify the maximum segment size (MSS) for TCP SYN packets traveling through the interface. The router compares the MSS value of incoming or outgoing packets against the MSS adjustment value. For any packet that contains an MSS value larger than the MSS adjustment value, the router replaces the MSS option with the configured adjustment value. If the packet does not contain an MSS value, the router assumes a value of 536 for the packet MSS on which to base the comparison.

NOTE: The purpose behind using MSS is to alleviate problems with Path MTU Discovery (PMTUD) and resulting "black hole" detection issues. (See RFC 2923, "TCP Problems with Path MTU Discovery," for additional information about the "black hole" scenario.)

- Example

  host1(config-if)#**ip tcp adjust-mss 5000**

- Use the **no** version to remove the MSS assignment from the profile.

ip unnumbered

- Use to specify the numbered interface with which dynamic unnumbered interfaces created with the profile are associated.
- You can specify an unnumbered interface using RADIUS instead of using the **ip unnumbered** command in a profile.
- Example

  host1(config-profile)#**ip unnumbered fastEthernet 0/0**

- Use the **no** version to remove the assignment from the profile.

ip virtual-router

- Use to assign a virtual router to a profile.
- You can configure a virtual router using RADIUS instead of adding one to the profile by using the **ip virtual-router** command.
- Example

  host1(config-profile)#**ip virtual-router VR1**

- Use the **no** version to remove the virtual router assignment.

profile

- Use to create a profile.
- You specify a profile name with up to 80 characters.
- Example

  host1(config)#**profile foo**

- Use the **no** version to remove a profile.

Assigning a Profile

To assign a profile to an interface, use the **profile** command from Interface mode.

profile

- Use to assign a profile to a PPP interface. The profile configuration is used to dynamically create an upper IP interface.
- Example

  host1(config-if)#**interface serial 2/1**
  host1(config-if)#**encapsulation ppp**
  host1(config-if)#**profile acton**

- Use the **no** version to remove the assignment from the interface.

Address Resolution Protocol

Sending IP packets on a multiaccess network requires mapping from an IP address to a MAC address (the physical or hardware address).

In an Ethernet environment, Address Resolution Protocol (ARP) is used to map a MAC address to an IP address. ARP dynamically binds the IP address (the logical address) to the correct MAC address. Before IP unicast packets can be sent, ARP discovers the MAC address used by the Ethernet interface where the IP address is configured.

Hosts that use ARP maintain a cache of discovered Internet-to-Ethernet address mappings to minimize the number of ARP broadcast messages. To keep the cache from growing too large, an entry is removed if it is not used within a certain period of time. Before sending a packet, the host searches its cache for Internet-to-Ethernet address mapping. If the mapping is not found, the host sends an ARP request.

NOTE: For information about MAC address validation, see *MAC Address Validation* on page 22.

How ARP Works

Figure 8 and Figure 9 show how ARP works where host 1 sends an IP packet to host 2 on a different subnet. To complete this transmission, host 1 needs the MAC address of router 1, to be used as the forwarding gateway.

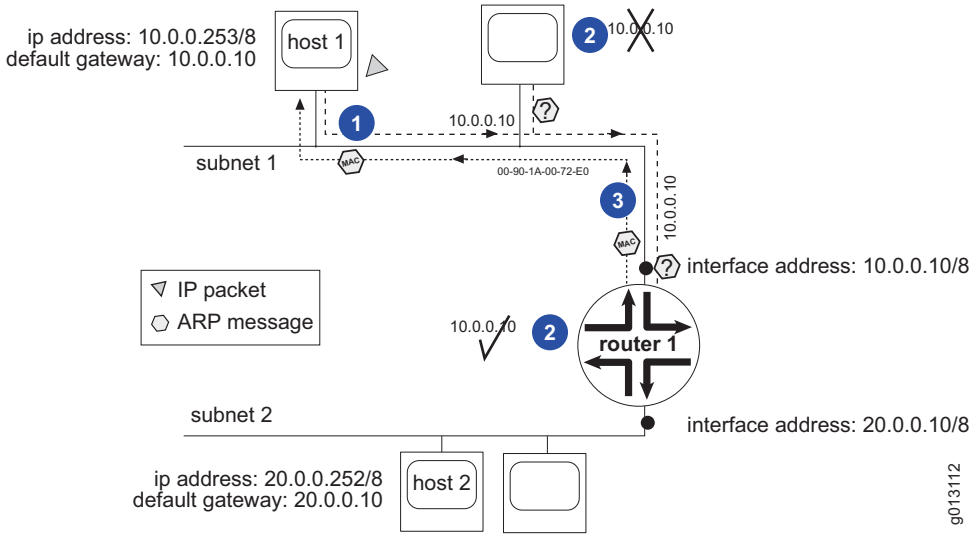
A typical scenario is:

1. Host 1 broadcasts an ARP request to all devices on subnet 1, composed by a query with the IP address of router 1. The IP address of router 1 is needed because host 2 is on a different subnet.

2. All devices on subnet 1 compare their IP address with the enclosed IP address sent by host 1.

3. Having the matching IP address, router 1 sends an ARP response, which includes its MAC address, to host 1.

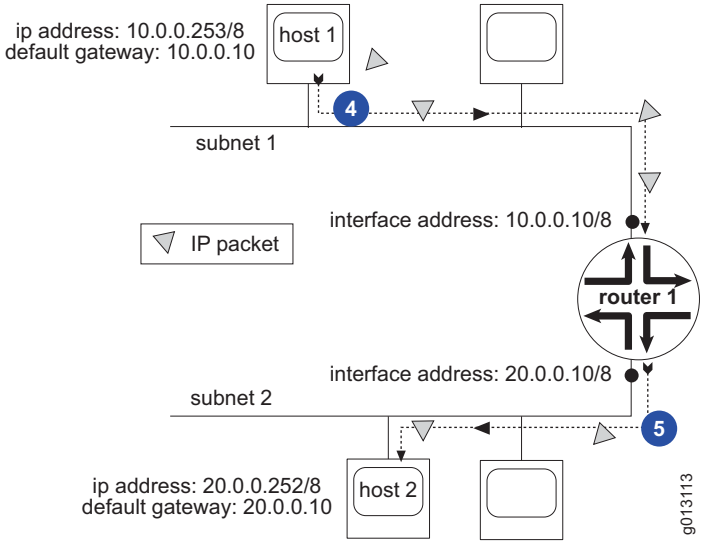
Figure 8: How ARP Works, Steps 1, 2, and 3



4. Host 1 transmits the IP packet to layer 3 DA (host 2) using router 1's MAC address.

5. Router 1 forwards IP packet to host 2. Router 1 might send an ARP request to identify the MAC of host 2. (See Figure 9.)

Figure 9: How ARP Works, Steps 4 and 5



ARP forces all receiving hosts to compare their IP addresses with the IP address of the ARP request. So if host 1 sends another IP packet to host 2, host 1 searches its ARP table for the router 1 MAC address.

If the default router/gateway becomes unavailable, then all the routing/packet forwarding to remote destinations ceases. Usually, manual intervention is required to restore connectivity, even though alternative paths may be available. Alternatively, Virtual Router Redundancy Protocol (VRRP) may be used to prevent loss of connectivity. See *JUNOSe IP Services Configuration Guide, Chapter 14, Configuring VRRP*.

arp

- Use to add a static (permanent) entry in the ARP cache.
- To add a static entry in the ARP cache, specify the *ipAddress*, *interfaceType* and *interfaceSpecifier (*as indicated in *Interface Types and Specifiers* in *JUNOSe Command Reference Guide, About This Guide)*, and an optional MAC address
- You can issue this command only for Fast Ethernet interfaces, Gigabit Ethernet interfaces, 10-Gigabit Ethernet interfaces, and bridged Ethernet interfaces configured over ATM 1483.
- Example

  host1(config)#**arp 192.56.20.1 gig 2/0 0090.1a00.0170**

- Use the **no** version to remove an entry from the ARP cache.

arp timeout

- Use to specify how long an entry remains in the ARP cache.
- You can issue this command only for Fast Ethernet interfaces, Gigabit Ethernet interfaces, 10-Gigabit Ethernet interfaces, and bridged Ethernet interfaces configured over ATM 1483.
- The default value is 21,600 seconds (6 hours). Use the **show config** command to display the current value.
- If you specify a timeout of 0 seconds, entries are never cleared from the ARP cache.
- Example

  host1(config-if)#**arp timeout 8000**

- Use the **no** version to restore the default value.

clear arp

- Use to clear dynamic entries from the ARP cache.
- To clear a particular entry, specify all of the following:
  - *ipAddress*—IP address in four-part dotted-decimal format corresponding to the local data link address
  - *interfaceType*—Interface type; see *Interface Types and Specifiers* in *JUNOSe Command Reference Guide, About This Guide*
  - *interfaceSpecifier*—Particular interface; format varies according to interface type; see *Interface Types and Specifiers* in *JUNOSe Command Reference Guide, About This Guide*
- To clear all dynamic ARP entries, specify an asterisk (*****).
- Example

  host1#**clear arp**

- There is no **no** version.

ip proxy-arp

- Use to enable proxy ARP on an Ethernet or bridge1483 interface.
- Proxy ARP is enabled by default.
- Example

  host1(config-if)#**ip proxy-arp unrestricted**

- Use the **no** version to disable proxy ARP on the interface.

MAC Address Validation

MAC address validation is a verification process performed on each incoming packet to prevent spoofing on IP Ethernet-based interfaces, including bridged Ethernet interfaces. When an incoming packet arrives on a layer 2 interface, the validation table is used to compare the packet's source IP address with its MAC address. If the MAC address and IP address match, the packet is forwarded; if it does not match, the packet is dropped.

NOTE: MAC address validation for bridged Ethernet interfaces is supported only on OC12a line modules.

MAC address validation on the E-series router can be accomplished in two ways:

- You can statically configure it on a physical interface via the **arp validate** command
- You can enable DHCP to perform the function independently and dynamically. See *DHCP* in *JUNOSe Link Layer Configuration Guide, Chapter 11, Configuring Bridged IP*.

The **arp validate** command adds the IP-MAC address pair to the validation table maintained on the physical interface.

If the validation is added statically via the CLI, the IP address–MAC address pairs are stored in NVS. The entries are used for MAC validation only if MAC validation is enabled on the interface via the **ip mac-validate** command.

CAUTION: When you configure an interface using the **arp validate** command, you cannot overwrite the ARP values that were added by DHCP.

You can enable or disable MAC address validation on a per interface basis by issuing the **ip mac-validate** command. See *JUNOSe Physical Layer Configuration Guide, Chapter 5, Configuring Ethernet Interfaces* or *JUNOSe Link Layer Configuration Guide, Chapter 12, Configuring Bridged Ethernet* for information.

A dynamic IP subscriber interface inherits the MAC address validation state (enabled or disabled) configured for its parent static primary IP interface. See *Inheritance of MAC Address Validation State for Dynamic Subscriber Interfaces* in *JUNOSe Broadband Access Configuration Guide, Chapter 25, Configuring Subscriber Interfaces* for information.

arp validate

- Use to add IP address–MAC address validation pairs. When validation is enabled, all packets with the source IP address received on this IP interface are validated against the IP-MAC entries.
- To add a validation pair, specify one of the following:
  - *ipAddress* and *macAddress* of the interface
  - *ipAddress*, *interfaceType* and *interfaceSpecifier (*as indicated in *Interface Types and Specifiers* in *JUNOSe Command Reference Guide, About This Guide)*, and an optional MAC address
- You can issue this command only for an IP Ethernet-based interface.
- For subscriber interface configurations, the IP address–MAC address pair must have a matching source prefix that already exists on the subscriber interface. If the matching source prefix does not exist, the IP–MAC address pair is rejected. See *JUNOSe Broadband Access Configuration Guide, Chapter 25, Configuring Subscriber Interfaces* for information about using subscriber interfaces.
- Example 1—Packets originating from host 192.56.20.1 and validated at Gigabit Ethernet interface with the MAC address 0090.1a00.0170

  host1(config)#**arp 192.56.20.1 gig 2/0 0090.1a00.0170 validate**

- Example 2—Subscriber interface MAC address validation enabled

  host1(config)#**arp 192.168.32.0 ip subsc1 000.0001.8100**

- Use the **no** version to remove an entry from the ARP cache.

Broadcast Addressing

A broadcast is a data packet destined for all hosts on a particular physical network. Network hosts recognize broadcasts by special addresses.

The router supports the following kinds of broadcast types:

- Limited broadcast—A packet is sent to a specific network or series of networks. A limited broadcast address includes the network or subnet fields. In a limited broadcast packet destined for a local network, the network identifier portion and host identifier portion of the destination address is either all ones (255.255.255.255) or all zeros (0.0.0.0).
- Flooded broadcast—A packet is sent to every network.
- Directed broadcast—A packet is sent to a specific destination address where only the host portion of the IP address is either all ones or all zeros (such as 192.20.255.255 or 190.20.0.0).

Several early IP implementations do not use the current broadcast address standard. Instead, they use the old standard, which calls for all zeros instead of all ones to indicate broadcast addresses. Many of these implementations do not recognize a broadcast address of all ones and fail to respond to the broadcast correctly. Others forward broadcasts of all ones, which causes a serious network overload known as a *broadcast storm*. Implementations that exhibit these problems include systems based on versions of BSD UNIX before version 4.3.

Routers provide some protection from broadcast storms by limiting their extent to the local cable. Bridges (including intelligent bridges), because they are layer 2 devices, forward broadcasts to all network segments, thus propagating all broadcast storms.

The best solution to the broadcast storm is to use a single broadcast address scheme on a network. Most IP implementations allow the network manager to set the address to be used as the broadcast address. Many implementations of IP, including the one on your router, can accept and interpret all possible forms of broadcast addresses.

Broadcast Tasks

You can use two broadcast-related IP commands to perform broadcast-related tasks.

ip broadcast-address

- Use to broadcast to all addresses in the host portion of an IP address.
- You specify an IP address to set the broadcast address.
- Example

  host1(config-if)#**ip broadcast-address 255.255.255.255**

- Use the **no** version to restore the default IP broadcast address.

ip directed-broadcast

- Use to enable translation of directed broadcasts to physical broadcasts.
- Example

  host1(config-if)#**ip directed-broadcast**

- Use the **no** version to disable the function.

Fragmentation

Fragmentation is the process of segmenting a large IP datagram into several smaller pieces. Fragmentation is required when IP must transmit a large packet through a network that transmits smaller packets, or when the MTU size of the other network is smaller.

By default, the router does not fragment the packet if the don't-fragment bit (DF bit) is set in the IP header. You can specify that the router not consider the DF bit before determining whether to fragment a packet.

NOTE: Lower-layer protocols can also set the MTU value. If MTU values set in lower layers differ from the one set at the IP layer, the router always uses the MTU lower-layer value.

ip ignore-df-bit

- Use to force the router to ignore the DF bit if it is set in the IP packet header for packets on an interface.
- Example

  host1(config-if)#**ip ignore-df-bit**

- Use the **no** version to restore the default behavior, which is to consider the DF bit before fragmentation.

ip mtu

- Use to set the MTU size of IP packets sent on an interface.
- The range is 128–10240.
- Do not configure both MLPPP fragmentation (with the **ppp fragmentation** command) and IP fragmentation of L2TP packets (with the **ip mtu** command) on the same interface. Instead, you must choose only one of the fragmentation configurations by setting it to the necessary value and set the other fragmentation configuration to the maximum allowable value.
- Example

  host1(config-if)#**ip mtu 1000**

- Use the **no** version to restore the default MTU size.

IP Routing

The Internet is a large collection of hosts that communicate with each other and use routers as intermediate packet switches.

Routers forward a packet through the interconnected system of networks and routers until the packet reaches a router that is attached to the same network as the destination host. The router delivers the packet to the specified host on its local network.

Routing Information Tables

A router makes forwarding decisions based on the information that is contained in its routing table. Routers announce and receive route information to and from other routers. They build tables of routes based on the collected information about all the best paths to all the destinations they know how to reach.

Each configured protocol has one or more local routing tables, sometimes referred to as a routing information base (RIB). This table is a database local to the protocol that contains all the routes known by that protocol to the prefixes in the table. For example, OSPF might have four different routes to 10.23.40.5.32. Only one of these routes is the best route to that prefix known to OSPF, but all four routes are in the OSPF local routing table.

The global routing table is a database maintained by IP on the SRP module. It contains at most one route per protocol to each prefix in the table. Each of these routes is the best route known by a given protocol to get to that prefix. The IP routing table does not, for example, have two OSPF routes to 10.5.11.0/24; it will have only one (if any) OSPF route to that prefix. It might also have a BGP route to the prefix, and a RIP route to the prefix, but no more than one route to a prefix per protocol.

IP compares the administrative distances for the routes to each prefix and selects the overall best route regardless of protocol. The best route to 10.5.11.0/24 might be via IS-IS. The best route to 192.168.0.0/16 might be via EBGP, and so on. These selected overall best routes to each prefix are used to create the forwarding table. The forwarding table is pushed to each line module. The line modules use their local instance of the forwarding table to forward the packets that they receive. When the global IP routing table is updated, so are the forwarding tables on the line modules.

Figure 10 illustrates a very simple network composed of three networks and two routers. The hosts that are attached to each network are not shown, because each router makes its forwarding decisions based on the network number and not on the address of each individual host. The router uses ARP to find the physical address that corresponds to the Internet address for any host or router on networks directly connected to it.

Figure 10: Routers in a Small Network

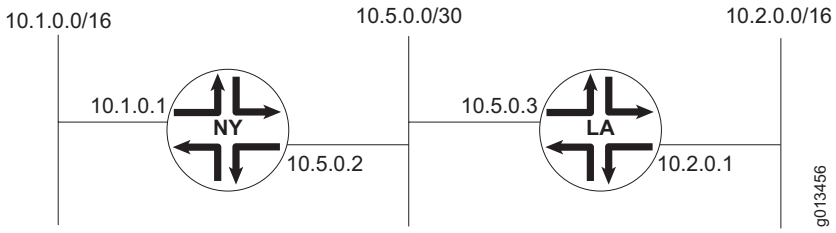


Table 4 and Table 5 represent information from the routing tables for routers NY and LA. Each routing table contains one entry for each route for each protocol or route type. Each routing table entry includes the following information:

- The destination IP network address.
- The IP address of the next-hop router.
- The type of network, such as static, directly connected, or the particular protocol.
- An administrative distance that is used to select the least-cost route among multiple routes to the same destination network. The least-cost (best) route is placed in the forwarding table. The administrative distance is not included in the forwarding table.
- A metric that is used by protocols to which the route is redistributed to select the least-cost route among multiple routes to the same destination network. The metric is not used to determine the best route to be placed in the forwarding table. The metric is also not listed in the forwarding table.

Table 4: Routing Table for Router NY

Destination Network	Next-Hop Router	Route Type	Administrative Distance	Metric
10.1.0.0/16	10.1.0.1	connected	0	0
10.2.0.0/16	10.5.0.3	OSPF	110	10
10.2.0.0/16	10.5.0.3	IS-IS	115	10
10.2.0.0/16	10.5.0.3	EBGP	20	15
10.2.0.0/16	10.5.0.3	RIP	120	5
10.5.0.0/30	10.5.0.2	connected	0	0

Table 5: Routing Table for Router LA

Destination Network	Next-Hop Router	Route Type	Administrative Distance	Metric
10.1.0.0/16	10.5.0.2	static	1	0
10.1.0.0/16	10.5.0.2	OSPF	110	10
10.1.0.0/16	10.5.0.2	RIP	120	4
10.2.0.0/16	10.2.0.1	connected	0	0
10.5.0.0/30	10.5.0.3	connected	0	0

Setting the Administrative Distance for a Route

The administrative distance is an integer that is associated with each route known to a router. The distance represents how reliable the source of the route is considered to be. A lower value is preferred over a higher value. An administrative distance of 255 indicates no confidence in the source; routes with this distance are not installed in the routing table.

Table 6 lists the default distance for each type of source from which a route can be learned.

Table 6: Default Administrative Distances for Route Sources

Route Source	Default Distance
Connected interface	0
Static route	1
Internal access route	2
Access route	3
External BGP	20
OSPF	110
IS-IS	115
RIP	120
Internal BGP	200
Unknown	255

If the IP routing table contains several routes to the same prefix—for example, an OSPF route and a RIP route—the route with the lowest administrative distance is used for forwarding.

To set the administrative distance for BGP routes, see *Setting the Administrative Distance for a Route* in *JUNOSe BGP and MPLS Configuration Guide, Chapter 1, Configuring BGP Routing*.

To set the administrative distance for RIP, IS-IS, and OSPF, use the following **distance** commands in Router Configuration mode.

distance

- Use to set an administrative distance for RIP or OSPF routes in the range 0–255.
- For RIP routes, the default value is 120.
- For OSPF routes, the default value is 110.
- Example

  host1(config)#**router rip**
  host1(config-router)#**distance 100**

- Use the **no** version to restore the default value.

distance ip

- Use to set the administrative distance for IS-IS routes in the range 1–255.
- Example

  host1(config)#**router isis**
  host1(config-router)#**distance 80 ip**

- Use the **no** version to restore the default value of 115.

Setting the Metric for a Route

For information about how to set a metric for a route, see *JUNOSe IP Services Configuration Guide, Chapter 1, Configuring Routing Policy* as well as the individual routing protocol chapters in the *JUNOSe BGP and MPLS Configuration Guide,* and in this guide.

Routing Operations

Routers keep track of next-hop information that enables a data packet to reach its destination through the network. A router that does not have a direct physical connection to the destination checks its routing table and forwards packets to another next-hop router that is closer to that destination. This process continues until the packet reaches its final destination.

Identifying a Router Within an Autonomous System

The router ID is commonly one of the router's defined IP addresses. Although the router ID is, by convention, formatted as an IP address, it is not required to be a configured address of the router. If you do not use the **ip router-id** command to assign a router ID, the router uses one of its configured IP addresses as the router ID.

ip router-id

- Use to assign a router ID—a unique identifier that IP routing protocols use to identify the router within an autonomous system.
- Example

  host1(config)#**ip router-id 192.32.15.23**

- Use the **no** version to remove the router ID assignment.

Establishing a Static Route

You can set a destination to receive and send traffic by a specific route through the network.

ip route

- Use to establish a static route.
- Example

  host1(config)#**ip route 192.56.15.23 255.255.255.0 192.66.0.1**

- Use the **no** version to remove a static route from the routing table.

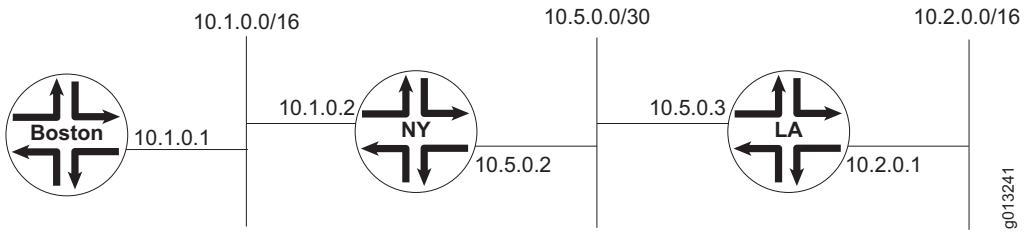
Configuring Static Routes with Indirect Next Hops

You can configure static routes where next hops are not on directly connected interfaces. Such a route is usable, and appears in the route table, only if another route in the route table can resolve the next hop.

The resolving route can be either statically created or dynamically learned by a routing protocol (like RIP, BGP, OSPF, and so on).

NOTE: When configuring this type of static route, the route that resolves the next hop must have an administrative distance value that is better (lower) than the distance of the static route you want to resolve.

Figure 11: Static Routes with Indirect Next Hops



On the Boston router in the network shown in Figure 11:

1. Configure a static route to 10.2.0.0/16 with a next hop of 10.5.0.2 (which is not directly connected) and an administrative distance of 254 (which is worse [higher] than the default administrative distance for static routes [1]).

   host1(config)#**ip route 10.2.0.0 255.255.0.0 10.5.0.2 254**

2. Configure another static route that resolves 10.5.0.2 and uses the default administrative distance.

   host1(config)#**ip route 10.5.0.0 255.255.255.252 10.1.0.2** [**1**]

NOTE: The previous example shows the default administrative distance value, 1, to illustrate the difference between the two static route commands. However, you do not have to enter this value when issuing the command.

A static route to 10.2.0.0 is added to the route table with a next hop of 10.1.0.2 (on the directly connected Ethernet interface).

NOTE: A dynamically learned route can also resolve indirect next hops, as long as the administrative distance value of the learned route is better (lower) than the static route whose next hop is being resolved.

Verifying Next Hops for Static Routes

You can configure either Bidirectional Forwarding Detection (BFD) or Response Time Reporter (RTR) probes to further control when a static route is installed in the routing table. Using either BFD or RTR, static route installation is based on two factors: whether the next hop to the specified destination is resolved, and whether an IP service running above the static route application is currently available.

Next-hop verification is useful for static route configurations in which the layer 2 and layer 3 interfaces are up and the next hop to the specified destination is available, but the IP services that run above the static route are currently unavailable. When the upper-layer IP services are unavailable, the router does not install the static route in its routing table.

How BFD Next-Hop Verification Works

Static routes on E-series routers can use Bidirectional Forwarding Detection (BFD) to verify the availability of the next hop and obtain the state of the IP service. For additional information about BFD, see *JUNOSe IP Services Configuration Guide, Chapter 5, Configuring BFD*.

If you specify the **bfd-liveness-detection** keywords with a minimum receive interval, minimum transmit interval, or multiplier when you issue the **ip route** command to establish a static route, the router verifies the next-hop status and installs the static route in the routing table under the following conditions:

- You configure the static routes with the actual next hop address and not just interface details.
- The BFD protocol is operational on both ends of the verification.
- The next hop is adjacent to the router (that is, only one hop away).

NOTE: BFD next-hop verification does not currently function in a multi-hop configuration.

- The next hop to the specified IP destination address is resolved.

You can further control the installation of static routes by specifying the **last-resort** keyword, which is valid when you use the **bfd-liveness-detection** keywords in the **ip route** command. The **last-resort** keyword instructs the router to install the static route in the routing table even if the specified BFD operation is unreachable, provided that no other static route to the same network prefix is available.

The static route is removed from the routing table if the next hop is no longer reachable and reinstalled when the route becomes reachable again.

BFD Next Hop Verification Configuration Example

To enable BFD next hop verification between two adjacent peers, you configure each peer as follows:

1. Configure peer A with the next hop address of peer B along with the desired intervals and keyword options.

   host1(config)#**ip route 192.1.1.0 255.255.255.0 192.1.2.1 verify bfd-liveness-detection minimum-interval 500 multiplier 3 last-resort**

2. Configure peer B with the next hop address of peer A along with the desired intervals and keyword options.

   host1(config)#**ip route 192.1.2.1 255.255.255.0 192.1.1.0 verify bfd-liveness-detection minimum-interval 300 multiplier 3**

ip route verify bfd-liveness-detection

- Use to enable BFD on a static route.
- Use the **minimum-interval** keyword to specify a value in the range 100–65535 milliseconds. This keyword defines both the minimum receive interval and minimum transmit interval using the same value.
- Use the **minimum-receive-interval** keyword to specify a minimum receive interval value in the range 100–65535 milliseconds.
- Use the **minimum-transmit-interval** keyword to specify a minimum transmit interval value in the range 100–65535 milliseconds.
- Use the **multiplier** keyword to specify a multiplier number in the range 1–255.
- Optionally, you can include the **last-resort** keyword when you use the **verify bfd-liveness-detection** keywords to instruct the router to install the static route in the routing table even if the specified BFD operation is currently unreachable, provided that no other static route to the same network prefix is available.
- Change parameters at any time without stopping or restarting the existing session; BFD automatically adjusts to the new parameter value. However, no changes to BFD parameters take place until the values resynchronize with each BFD peer.
- Example 1—Next hop address and last resort

  host1(config)#**ip route 192.56.15.23 255.255.255.0 192.66.0.1 verify bfd-liveness-detection minimum-interval 800 multiplier 2 last-resort**

- Example 2—Next hop address and interface

  host1(config)#**ip route 192.56.15.24 255.255.255.0 192.66.0.2 fast 6/0 verify bfd-liveness-detection**

- Example 3—Next hop address with different receive and transmit intervals

  host1(config)#**ip route 192.56.15.23 255.255.255.0 192.66.0.1 verify bfd-liveness-detection minimum-receive-interval 800 minimum-transmit-interval 300 multiplier 2 last-resort**

- Use the **no** version to remove the static route from the routing table and thereby remove BFD from that static route.

How RTR Next-Hop Verification Works

Static routes on E-series routers can use Response Time Reporter (RTR) probes configured as echo (ping) types to verify the availability of the next hop and obtain the state of the IP service. For more information about using RTR, see *Response Time Reporter* on page 63.

If you specify the **verify rtr** keywords with an RTR operation number when you issue the **ip route** command to establish a static route, the router verifies the next-hop status and installs the static route in the routing table only if *both* of the following conditions are met:

- The next hop to the specified IP destination address is resolved.
- The specified RTR operation is currently reachable.

You can further control the installation of static routes by specifying the **last-resort** keyword, which is valid only when you use the **verify rtr** keywords in the **ip route** command. The **last-resort** keyword instructs the router to install the static route in the routing table even if the specified RTR operation is unreachable, provided that no other static route to the same network prefix is available.

Although the configuration example in the next section uses Fast Ethernet interfaces, E-series routers support next-hop verification on any type of lower-layer interface.

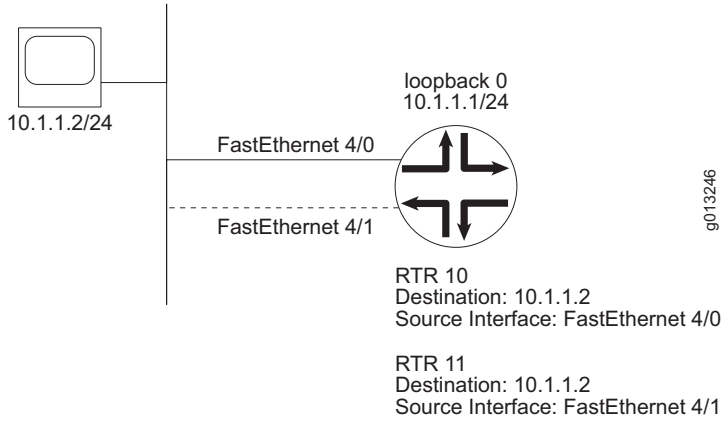
RTR Configuration Example

Figure 12 shows a sample configuration that illustrates the next-hop verification feature. In this example, two Fast Ethernet interfaces are configured between a remote system and an E-series router: Fast Ethernet interface 4/0 and Fast Ethernet interface 4/1. At any given time, only one of these interfaces forwards IP traffic, even though the associated layer 2 interfaces may be up concurrently.

On the E-series router, Fast Ethernet interfaces 4/0 and 4/1 are configured as unnumbered IP interfaces. In addition, each interface has an RTR probe configured as an echo type that sends requests over the interface to determine its availability. RTR 10 sends requests over Fast Ethernet interface 4/0, and RTR 11 sends requests over Fast Ethernet interface 4/1.

In this example, both RTR 10 and RTR 11 use the IP address of the remote system (10.1.1.2) as the target address. When you configure multiple RTR entries to use the same target address, you must set the **receive-interface** attribute to specify the interface on which the probe expects to receive responses. (See Step 4c in the next section, *Configuring RTR Next-Hop Verification*.) This action enables the router to map incoming responses to the proper RTR entry, even when multiple RTR entries have the same target address.

Figure 12: Sample Configuration for Next-Hop Verification



The **ip route** command is issued for each interface with the **verify rtr** and **last-resort** keywords to establish the necessary static routes. (See Steps 6 and 7 in the next section, *Configuring RTR Next-Hop Verification*.) This command causes the results described in Table 7, based on the status of the associated RTR operations.

Table 7: Next-Hop Verification Results for Sample Configuration

RTR 10 Status	RTR 11 Status	Results
Up	Up	The router installs an equal-cost multipath (ECMP) route to 10.1.1.2 in the routing table, using Fast Ethernet interfaces 4/0 and 4/1 as the next hops.
Up	Down	The router installs a route to 10.1.1.2, using Fast Ethernet interface 4/0 as the next hop.
Down	Up	The router installs a route to 10.1.1.2, using Fast Ethernet interface 4/1 as the next hop.
Down	Down	Although both RTR operations are down, the **last-resort** keyword instructs the router to install an ECMP route to 10.1.1.2, using Fast Ethernet interfaces 4/0 and 4/1 as the next hops. When all of the RTR operations associated with your static routes are down, you can control which route is installed in the routing table by including the **last-resort** keyword in the **ip route verify rtr** command only for the route that you want to install.

The next section, *Configuring RTR Next-Hop Verification*, provides instructions for configuring the example shown in Figure 12.

Configuring RTR Next-Hop Verification

To configure the next-hop verification example shown in Figure 12:

1. Configure a loopback interface, and assign an IP address and mask to the interface.

   ```
   host1(config)#interface loopback 0
   host1(config-if)#ip address 10.1.1.1 255.255.255.0
   host1(config-if)#exit
   ```

2. Configure Fast Ethernet port 4/0 with an unnumbered primary IP interface associated with the loopback interface configured in Step 1.

   ```
   host1(config)#interface fastEthernet 4/0
   host1(config-if)#ip unnumbered loopback 0
   host1(config-if)#exit
   ```

3. Repeat Step 2 for Fast Ethernet port 4/1.

   ```
   host1(config)#interface fastEthernet 4/1
   host1(config-if)#ip unnumbered loopback 0
   host1(config-if)#exit
   ```

4. Define probe RTR 10 for Fast Ethernet interface 4/0.

   a. Assign an operation number to the RTR probe, and access RTR Configuration mode. For information, see *Configuring the Probe Type* on page 64.

      ```
      host1(config)#rtr 10
      host1(config-rtr)#
      ```

   b. Configure the RTR probe as an echo type, and set the IP destination address and source interface.

      You must configure the RTR probe as an echo type to use next-hop verification. For information, see *Configuring the Probe Type* on page 64.

      ```
      host1(config-rtr)#type echo protocol ipIcmpEcho 10.1.1.2
      source fastEthernet 4/0
      ```

   c. Specify the interface on which the RTR probe expects to receive responses.

      You must set the **receive-interface** attribute when multiple RTR operations use the same target address. For information, see *Setting the Receiving Interface* on page 68.

      ```
      host1(config-rtr)#receive-interface fastEthernet 4/0
      ```

   d. (Optional) Configure optional probe characteristics, such as the frequency and samples-of-history kept. For information, see *Configuring Optional Characteristics* on page 65.

      ```
      host1(config-rtr)#frequency 1
      host1(config-rtr)#samples-of-history-kept 0
      ```

e. Exit RTR Configuration mode.

```
host1(config-rtr)#exit
```

f. Enable the probe to react to the test-failure event and the test-completion event.

You must configure both the test-failure and test-completion reaction conditions to use next-hop verification. For information, see *Setting Reaction Conditions* on page 69.

```
host1(config)#rtr reaction-configuration 10   test-failure 3
host1(config)#rtr reaction-configuration 10   test-completion
```

g. Schedule the probe operation. For information, see *Scheduling the Probe* on page 70.

```
host1(config)#rtr schedule 10 life 3
host1(config)#rtr schedule 10 restart-time 1
host1(config)#rtr schedule 10 start now
```

5. Repeat Step 4 to define RTR 11 for Fast Ethernet interface 4/1.

```
host1(config)#rtr 11
host1(config-rtr)#type echo protocol ipIcmpEcho 10.1.1.2
source fastEthernet 4/1
host1(config-rtr)#receive-interface fastEthernet 4/1
host1(config-rtr)#frequency 1
host1(config-rtr)#samples-of-history-kept 0
host1(config-rtr)#exit
host1(config)#rtr reaction-configuration 11 test-failure 3
host1(config)#rtr reaction-configuration 11 test-completion
host1(config)#rtr schedule 11 life 3
host1(config)#rtr schedule 11 restart-time 1
host1(config)#rtr schedule 11 start now
```

6. Establish a static route associated with RTR 10.

This command creates a static route and installs it in the routing table only if RTR 10 is currently reachable *or* if no other static route to IP destination address 10.1.1.2 is usable.

```
host1(config)#ip route 10.1.1.2 255.255.255.255 10.1.1.2 fastEthernet 4/0
verify rtr 10 last-resort
```

7. Establish a static route associated with RTR 11.

This command creates a static route and installs it in the routing table only if RTR 11 is currently reachable *or* if no other static route to IP destination address 10.1.1.2 is usable.

```
host1(config)#ip route 10.1.1.2 255.255.255.255 10.1.1.2 fastEthernet 4/1
verify rtr 11 last-resort
```

When both RTR 10 and RTR 11 are unreachable, you can control which static route is installed in the routing table by including the **last-resort** keyword in the **ip route verify rtr** command only for the route that you want to install.

NOTE: For detailed information about the commands for configuring RTR probes, see *Response Time Reporter* on page 63.

interface fastEthernet

- Use to select a Fast Ethernet (FE) interface on a line module or an SRP module.
- Example

  host1(config)#**interface fastEthernet 1/0**

- Use the **no** version to remove IP from an interface or subinterface. You must issue the **no** version from the highest level down; you cannot remove an interface or a subinterface if the one above it still exists.

NOTE: For more details on use of this command, see the syntax description in the *JUNOSe Command Reference Guide A to M*.

interface loopback

- Use to access and configure a loopback interface.
- You can use a loopback interface to provide a stable IP address that can minimize the impact if a physical interface goes down.
- Example

  host1(config)#**interface loopback 10**
  host1(config-if)#**ip address 100.20.32.1 255.255.255.0**

- Use the **no** version to delete the loopback interface.

ip address

- Use to set an IP address for an interface or a subinterface.
- Specify the layer 2 encapsulation before you set the IP address.
- Example

  host1(config-subif)#**ip address 192.0.2.50 255.255.255.0**

- Use the **no** version to remove the IP address or to disable IP processing on the interface.

ip route verify rtr

- Use to establish a static route and associate it with a configured RTR operation.
- Use the **verify rtr** keywords to instruct the router to install the static route in the routing table only if the next hop to the specified destination address is resolved and if the specified RTR operation is currently reachable. When you use the **verify rtr** keywords, you must also specify the number of the associated RTR operation.
- Optionally, you can include the **last-resort** keyword when you use the **verify rtr** keywords to instruct the router to install the static route in the routing table even if the specified RTR operation is currently unreachable, provided that no other static route to the same network prefix is available.
- Example

  host1(config)#**ip route 10.1.1.5 255.255.255.0 10.1.1.5 fastEthernet 1/0 verify rtr 5 last-resort**

- Use the **no** version to remove a static route from the routing table.

ip unnumbered

- Use to configure an unnumbered IP interface.
- This command enables IP processing on an interface without assigning an explicit IP address to the interface.
- You must specify an interface location, which is the identifier of another interface on which the router has an assigned IP address. This interface cannot be another unnumbered interface.
- Example 1

  host1(config-if)#**ip unnumbered fastEthernet 3/0**

- Example 2

  host1(config-if)#**ip unnumbered loopback 10**

- Use the **no** version to disable IP processing on the interface.

Setting Up Default Routes

A router examines its routing table to find a path for each packet. If the router cannot locate a route, it must discard the packet. You can set up a default route using the special address: 0.0.0.0. If the router cannot locate a path to a destination network and a default route is defined, the router forwards the packet to the default router. For example:

host1(config)#**ip route 0.0.0.0 0.0.0.0 192.56.21.3**

Default routes are typically used to reduce the size of the routing table. Routing is simplified because the router can test for a few local networks or use the default route. However, a disadvantage of default routes is the possible creation of multiple paths and routing loops.

Setting Up an Unnumbered Interface

An unnumbered interface does not have an IP address assigned to it. Unnumbered interfaces are often used in point-to-point connections where an IP address is not required.

ip unnumbered

- Use to set up an unnumbered interface.
- This command enables IP processing on an interface without assigning an explicit IP address to the interface.
- You supply an interface location, which is the type and number of another interface on which the router has an assigned IP address. This interface cannot be another unnumbered interface.
- Example

  host1(config-if)#**ip unnumbered fastEthernet 0/0**

- Use the **no** version to disable IP processing on an interface.

Adding a Host Route to a Peer on a PPP Interface

You can enable the ability to create host access routes on a PPP interface. This feature is useful in B-RAS applications.

ip access-routes

- Use to enable the ability to create host access routes on a PPP interface.
- Example

  host1(config-if)#**ip access-routes**

- Use the **no** version to disable this feature.

Enabling Source Address Validation

Source address validation verifies that a packet has been sent from a valid source address. When a packet arrives on an interface, the router performs a routing table lookup using the source address. The result from the routing table lookup is an interface to which packets destined for that address are routed. This interface must match the interface on which the packet arrived. If it does not match, the router drops the packet.

ip sa-validate

- Use to enable source address validation.
- Example

  host1(config-if)#**ip sa-validate**

- Use the **no** version to disable source address validation.

Enabling Source Address Validation Traps

The **ip sa-validate trap-enable** command enables the generation of traps for source address validation failure.

NOTE: To fully enable source address validation traps, you must also enable the IP trap category with the **snmp-server trap enable** command. See *JUNOSe System Basics Configuration Guide, Chapter 4, Configuring SNMP* for more information.

ip sa-validate trap-enable

- Use to enable the generation of traps for source address validation failure on the router.
- You can specify a VRF context for which you want to enable trap validation for source address validation.
- Example

  host1(config)#**ip sa-validate trap-enable**

- Use the **no** version to disable the generation of source address validation failure traps on the router.

Defining TCP Maximum Segment Size

The **ip tcp adjust-mss** command enables you to modify the TCP maximum segment size (MSS) for TCP sessions.

When defined, the router modifies the maximum segment size (MSS) for TCP SYN packets traveling through the interface. The modification occurs only for packets that contain values smaller than the adjusted MSS value. When the packet does not contain an MSS field value, the router assumes an MSS value of 536 and makes any modifications accordingly.

NOTE: Implementation of the MSS value is identical for both ingress and egress interface traffic. That is, the router uses the same MSS value when adjusting inbound or outbound TCP traffic.

ip tcp adjust-mss

- Use to modify the maximum segment size (MSS) for TCP SYN packets traveling through the interface. The router compares the MSS value of incoming or outgoing packets against the adjusted MSS setting and replaces smaller values that it detects in any packets with the larger setting. If the packet does not contain an MSS value, the router assumes a value of 536 for the packet MSS on which to base the comparison.

NOTE: The purpose behind using MSS is to alleviate problems with Path Discovery (PMTUD) and resulting "black hole" detection issues. (See RFC 2923, "TCP Problems with Path MTU Discovery," for additional information about the "black hole" scenario.)

- Example

  host1(config-if)#**ip tcp adjust-mss 1000**

- Use the **no** version to remove the MSS assignment from the profile.

Setting MSS for TCP Connections

MSS is used by TCP to define the maximum amount of data that a TCP interface can accept in any single packet (or segment size). The MSS value is typically negotiated during connection establishment and is not renegotiated.

By default, the router uses an MSS value of 536 bytes and the advertised MSS is derived from the MTU of the transmitting interface. However, you can use the **tcp mss** command to set the MSS for TCP advertisements.

tcp mss

- Use to specify the MSS value for TCP to advertise.

NOTE: The MSS value is equal to the MTU value minus the IP and TCP headers, so the MSS value is generally 40 bytes less than the MTU.

- Use the *vrfName* variable to specify a VRF to which you want to assign the TCP MSS value.
- Example

  host1(config-if)#**tcp mss 1000**

- Use the **no** version to remove the MSS value so that the router uses the advertised MSS derived from the MTU of the output interface.

Configuring IP Path MTU Discovery

IP hosts transmit large amounts of data to other hosts using a series of IP datagrams. To best use resources, increase performance, and avoid difficult reassembly, hosts try to send datagrams that are as large as possible without requiring fragmentation anywhere along the path from the source to the destination. This datagram size is referred to as the *path MTU* (*PMTU*), and it is equal to the smallest MTU for each hop in the path.

Path MTU discovery is the process of discovering the PMTU value and using that value when transmitting TCP packets in datagrams.

Enabling PMTU Discovery

Use the **tcp path-mtu-discovery** command to enable PMTU discovery on the active virtual router.

tcp path-mtu-discovery

- Use to enable and configure path MTU discovery on the virtual router.
- Issue the command without any keywords to enable path MTU discovery.
- Issue the **age-timer** keyword to set the time (*minutes*) that TCP waits before attempting to increase the path MTU after receiving an ICMP Too Big message or after previously increasing the PMTU successfully (*minutes2*). The range of these two timers is 1–30 minutes. The timer defaults to 10 minutes.
- Issue the **age-timer infinite** keyword to disable PMTU aging functions.
- Example 1—Enables path MTU discovery

  host1:VR1(config)#**tcp path-mtu-discovery**

- Example 2—Sets path MTU discovery age timers differently

  host1:VR1(config)#**tcp path-mtu-discovery age-timer 20 15**

- Example 3—Sets path MTU discovery age timers to the same value (5 minutes)

  host1:VR1(config)#**tcp path-mtu-discovery age-timer 5**

- Example 4—Disables path MTU discovery age timers

  host1:VR1(config)#**tcp path-mtu-discovery age-timer infinite**

- Use the **no** version with a keyword to return the value to its default. Issue the **no** version without any keywords to disable path MTU discovery on the virtual router.

Limiting PMTU

You can limit calculated PMTU values within a range by using the **tcp path-mtu-discovery max-mtu** and **tcp path-mtu-discovery min-mtu** commands. When specifying PMTU limits, keep the following in mind:

- If a PMTU discovery value is lower than the configured minimum MTU setting, PMTU discovery is disabled for that connection.
- If a PMTU discovery value is larger than the configured maximum MTU setting, the configured maximum MTU setting is used.
- The maximum MTU setting must be greater than the minimum MTU setting.

tcp path-mtu-discovery max-mtu

- Use to limit the maximum MTU size used for the path MTU.
- Example

  host1:VR1(config)#**tcp path-mtu-discovery max-mtu 512**

- Use the **no** version to remove any limitation so that the virtual router uses the path MTU discovery value.

tcp path-mtu-discovery min-mtu

- Use to specify the minimum MTU value used for the path MTU. If the discovered PMTU value is less than the minimum setting, path MTU discovery is disabled for this connection.
- Example

  host1:VR1(config)#**tcp path-mtu-discovery min-mtu 255**

- Use the **no** version to remove any limitation so that the virtual router uses the discovered path MTU value.

Specifying Black Hole Thresholds

A black hole threshold is a limit to the number of times a virtual router can retransmit identical sequences of datagrams before the retransmissions are identified as a problem.

Some domains might be configured not to generate certain ICMP messages (like an ICMP destination unreachable message) or to filter all ICMP messages. Under these conditions, the source of oversized ICMP packets never learns that it is sending oversized packets. The device continues sending oversized packets that never get through. This behavior is often referred to as a *black hole*.

tcp path-mtu-discovery black-hole-detect-threshold

- Use to specify the minimum MTU value used for the path MTU. If the discovered PMTU value is less than the minimum setting, path MTU discovery is disabled for this connection.
- Example

  host1:VR1(config)#**tcp path-mtu-discovery black-hole-detect-threshold 200**

- Use the **no** version to disable black hole threshold detection.

Shutting Down an IP Interface

You can disable an interface to the router at the IP level without removing it.

ip shutdown

- Use to shut down an IP interface.
- Example

  host1(config-if)#**ip shutdown**

- Use the **no** version to restart the interface.

Removing the IP Configuration

You can remove the IP configuration from an interface or subinterface.

no ip interface

- Use to remove the IP configuration from an interface or subinterface and disable IP processing on the interface.
- Example

  host1(config-if)#**no ip interface**

Clearing IP Routes

The router enables you to clear the specified routing entries from the routing table. You must specify the IP address prefix and the mask of the IP address prefix to clear specific routes. You can later reinstall the routes you have cleared.

clear ip routes

- Use to clear specified IP routes according to an IP prefix or a VPN routing and forwarding (VRF) table.
- Use an asterisk (*) to clear all dynamic routes from the routing table.
- Example

  host1#**clear ip routes ***

- There is no **no** version.

ip refresh-route

- Use to enable the owning protocols (BGP, IS-IS, OSPF) to reinstall routes removed from the IP routing table by the **clear ip routes** command.
- Example

  host1#**ip refresh-route**

- There is no **no** version.

Clearing IP Interfaces

The router enables you to clear the counters on one or more specified IP interfaces.

clear ip interface

- Use to clear a specified IP interface.
- Example

  host1#**clear ip interface pos 2/0**

- There is no **no** version.

Setting a Baseline

The router enables you to set a baseline for statistics on an IP interface.

baseline ip interface

- Use to set a baseline for a specified IP interface.
- Example

  host1#**baseline ip interface pos 2/0**

- There is no **no** version.

Disabling Forwarding of Packets

The router enables you to disable forwarding of packets on an SRP Ethernet interface.

ip disable-forwarding

- Use to disable forwarding of packets on the SRP Ethernet interface.
- The purpose of this command is to maintain router performance by maximizing the CPU time available for routing protocols. Although you can allow data forwarding on the SRP Ethernet interface, router performance will be affected.
- You see an error message if you try to set this command for interfaces other than the SRP Ethernet interface.
- Example

  host1(config-if)#**ip disable-forwarding**

- Use the **no** version to enable forwarding of packets on the interface.

Enabling Forwarding of Source-Routed Packets

IP packets are normally routed according to the destination address they contain based on the routing table at each hop through a path. The originator or source of the source-routed packets specifies the path (the series of hops) that the packets must traverse; the source makes the routing decisions. The source can specify either of the following types of source routing:

- *Strict-source* routing specifies every hop that the packet must traverse. The specified path consists of adjacent hops. The source generates an ICMP error if the exact path cannot be followed. For example, for a path going from source router A to router B to router C to router D, router A specifies a strict-source route as B, C, D.
- *Loose-source* routing specifies a set of hops that the packet must traverse, but not necessarily every hop in the path. That is, the specified hops do not have to be adjacent. For example, for a path going from source router A to router B to router C to router D, router A specifies a loose-source route as B, D or C, D, or B, C, D.

ip source-route

- Use to enable forwarding of source-routed packets in a VR or VRF.
- Forwarding is disabled by default in all VRs.
- Example

  host1(config)#**ip source-route**

- Use the **no** version to disable forwarding of source-routed packets on the VR or VRF.

Forcing an Interface to Appear Up

The router enables you to force an IP interface to appear as if it is up, regardless of the state of the lower layers.

ip alwaysup

- Use to force an IP interface to appear as up regardless of the state of lower layers.
- This command reduces route topology changes when the network attached to this link is single-homed.
- Example

  host1(config-if)#**ip alwaysup**

- Use the **no** version to make the interface appear in the current state.

Specifying a Debounce Time

You can set a debounce time that requires an IP interface to be in a given state—for example, up or down—for the specified time before the state is reported. This feature prevents a link that briefly goes up or down from causing an unnecessary topology change, for example by causing an interface down condition. However, note that increasing the debounce time increases the latency of updating the routing table to reflect an up or down change, and the latency of routing protocols propagating the state change.

ip debounce-time

- Use to set the interval in milliseconds for which an interface must maintain a given state before the state change is reported.
- Example

  host1(config)#**ip debounce-time 5000**

- Use the **no** version to remove the debounce time requirement.

Adding a Description

The router enables you to add a text description or an alias to a static IP interface or subinterface. Adding a description helps you identify the interface and keep track of interface connections. If no IP interface currently exists, then a static IP interface is automatically created on the current layer 2 interface and the description is applied to that static IP interface. You cannot assign a profile to a layer 2 interface that has a static interface configured above it.

ip description

- Use to assign a text description or an alias to an IP interface or subinterface.
- The description or alias can be a maximum of 256 characters.
- Use the **show ip interface** command to display the text description.

- Example 1

  host1(config-if)#**ip description canada01 ip interface**

- Example 2

  host1(config-subif)#**ip description montreal011 ip subinterface**

- Use the **no** version to remove the text description or alias.

Enabling Link Status Traps

The router enables you to enable link status traps on an interface.

snmp trap ip link-status

- Use to enable link status traps on an interface.
- Example

  host1(config-if)#**snmp trap ip link-status**

- Use the **no** version to disable link status traps on an interface.

Configuring the Speed

The router enables you to set the speed of an IP interface.

ip speed

- Use to set the speed of the interface in bits per second.
- By default, the speed is determined from a lower-layer interface.
- Example

  host1(config-if)#**ip speed 1000**

- Use the **no** version to set the speed to the default, 0.

Configuring Equal-Cost Multipath Load Sharing

Equal-cost multipath (ECMP) sets are formed when the router finds routing table entries for the same destination with equal cost. The router then balances traffic across these sets of equal-cost paths by using one of two ECMP modes—hashed (the default) or round-robin.

Hashed mode uses hashing of source and destination addresses to determine which of the available paths in the ECMP set to use. Hashed mode is the default ECMP mode of operation.

Defining Maximum Paths

You can add routing table entries manually (as static routes), or they are formed as routers discover their neighbors and exchange routing tables (via OSPF, BGP, and other routing protocols).

The **maximum paths** command controls the maximum number of parallel routes that the routing protocol (BGP, IS-IS, OSPF, or RIP) can support.

Round-Robin Mode

Round-robin mode distributes packets equally among the available paths in the ECMP set.

ip multipath round-robin

- Use to specify round-robin as the mode for ECMP load sharing on an interface.
- ECMP uses the round-robin mode when you have configured *all* interfaces in the set to round-robin. Otherwise, ECMP defaults to hashed mode because round-robin mode can cause reordering of packets. You must explicitly ensure that the possible reordering is acceptable on all the member interfaces by setting them to round-robin mode.
- If one of the ECMP next hops is an indirect next hop, ECMP uses hashed mode load balancing.
- Example

  host1(config)#**virtual-router router_0**
  host1:router_0(config)#**interface serial 4/0:1/22.22**
  host1:router_0(config-subif)#**ip multipath round-robin**
  host1:router_0(config-subif)#**exit**

- Use the **no** version to set the ECMP mode to the default, hashed.

maximum-paths

- Use to control the maximum number of parallel routes that the routing protocol supports.
- The maximum number of routes can be in the range 1–16 for BGP, IS-IS, OSPF, or RIP.
- Example

  host1(config-router)#**maximum-paths 2**

- Use the **no** version to restore the default value, 1 for BGP or 4 for IS-IS, OSPF, or RIP.

Fast Reroute Protection

If a link goes down, ECMP uses fast reroute protection to shift packet forwarding to use operational links, thereby decreasing packet loss. Fast reroute protection updates ECMP sets for the interface without having to wait for the route table update process. When the next route table update occurs, a new ECMP set can be added with fewer links or the route might point to a single next hop.

CAUTION: To provide ECMP fast reroute functionality in the event of an interface failure, the members of an equal cost multipath must be resolved to corresponding interfaces. If the member is an indirect next hop, the interface is obtained by using the forwarding equivalence class (FEC) to which the member points. This method of resolving members occurs only if the FEC, pointed to by the indirect next hop, is either an interface or a direct next hop.

An indirect next hop member is not resolved to an interface if it points to another indirect next hop or to an equal cost multipath. ECMP fast reroute functionality is not available if any interfaces that correspond to unresolved indirect next hop members go down.

If you modify an indirect next hop member to point to a different FEC (that is, a different interface, direct next hop, indirect next hop, or ECMP), the indirect next hop member is not resolved for the new changes.

Setting a TTL Value

You can use the **ip ttl** command to set the TTL (time-to-live) field in the IP header for all IP operations. The TTL specifies a hop count. This configured TTL value can be overridden by other commands that specify a TTL.

ip ttl

- Use to set a default value for the IP header TTL field for all IP operations.
- Example

  host1(config)#**ip ttl 255**

- Use the **no** version to restore the default value, 127.

Protecting Against TCP RST or SYN DoS Attacks

You can use the **tcp ack-rst-and-syn** command to help protect the router from denial of service (DoS) attacks.

Normally, when it receives an RST or SYN message, TCP attempts to shut down the TCP connection. This action is expected under normal conditions, but someone maliciously generating valid RST or SYN messages can cause problems for TCP and the network as a whole.

When you enable the **tcp ack-rst-and-syn** command, the router challenges any RST or SYN messages that it receives by sending an ACK message back to the expected source of the message. The source reacts in one of the following ways:

- If the source did send the RST or SYN message, it recognizes the ACK message to be spurious and resends another RST or SYN message. The second RST or SYN message causes the router to shut down the connection.
- If the source did not send the RST or SYN message, the source accepts the ACK message as part of an existing connection. As a result, the source does not send another RST or SYN message and the router does not shut down the connection.

NOTE: Enabling this command slightly modifies the way TCP processes RST or SYN messages to ensure that they are genuine.

tcp ack-rst-and-syn

- Use to help protect the router from TCP RST and SYN denial of service attacks.
- Example

  host1(config)#**tcp ack-rst-and-syn**

- Use the **no** version to disable this protection.

Preventing TCP PAWS Timestamp DoS Attacks

The TCP Protect Against Wrapped Sequence (PAWS) number option works by including the TCP timestamp option in all TCP headers to help validate the packet sequence number.

Normally, in PAWS packets that have the timestamps option enabled, hosts use an internal timer to compare the value of the timestamp associated with incoming segments against the last valid timestamp the host recorded. If the segment timestamp is larger than the value of the last valid timestamp, and the sequence number is less than the last acknowledgement sent, the host updates its internal timer with the new timestamp and passes the segment on for further processing.

If the host detects a segment timestamp that is smaller than the value of the last valid timestamp or the sequence number is greater than the last acknowledgement sent, the host rejects the segment.

A remote attacker can potentially determine the source and destination ports and IP addresses of both hosts that are engaged in an active connection. With this information, the attacker might be able to inject a specially crafted segment into the connection that contains a fabricated timestamp value. When the host receives this fabricated timestamp, it changes its internal timer value to match. If this timestamp value is larger than subsequent timestamp values from valid incoming segments, the host determines the incoming segments as being too old and discards them. The flow of data between hosts eventually stops, resulting in a denial of service condition.

Use the **tcp paws-disable** command to disable PAWS processing.

NOTE: Disabling PAWS does not disable other processing related to the TCP timestamp option. This means that even though you disable PAWS, a fabricated timestamp that already exists in the network can still pollute the database and result in a successful DoS attack. Enabling PAWS resets the saved timestamp state for all connections in the virtual router and stops any existing attack.

tcp paws-disable

- Use to disable the Protect Against Wrapped Sequence (PAWS) number option in TCP segments.
- You can specify a VRF context for which you want PAWS disabled.
- Example

  host1(config)#**tcp paws-disable**

- Use the **no** version to restore PAWS processing (the default mode).

Protecting Against TCP Out of Order DoS Attacks

You can use the group of **tcp resequence-buffers** commands to help protect the router from TCP out-of-order DoS attacks.

TCP guarantees that applications receive data in order. This means that TCP buffers any out-of-order packets it receives until ordered delivery can occur. To prevent buffers from consuming too many resources, TCP limits the amount of data it accepts to the number of data bytes that the receiver is willing to receive and buffer.

TCP does not take into account the buffering scheme that the receiver uses. If the receiver uses a fixed-size receive buffer (that is, buffering all packets) regardless of length, a packet that contains only one data byte might consume many data bytes of buffer space, but only one byte of TCP space.

Under these conditions, an attacker can send a large number of 1-byte packets to an E-series router in which each packet is buffered, consuming an entire packet buffer and eventually consuming a large amount of resources.

To defend against this sort of attack, you can set defaults and limits on the number of outstanding buffers on reordering queues. You can configure these defaults and limits on a per-router, per-virtual router, or per-connection basis.

Limiting Buffers per Router

The **tcp resequence-buffers global-maximum** command enables you to limit the number of outstanding buffers on the entire router.

tcp resequence-buffers global-maximum

- Use to specify a router-wide maximum number of buffers that resequencing queues can contain.
- Specify a value of zero (0) to turn off the limit.
- Example

  host1(config)#**tcp resequence-buffers global-maximum**

- Use the **no** version to revert the global maximum buffer value to its default, 1000 buffers.

Limiting Buffers per Virtual Router

The **tcp resequence-buffers default-vr-maximum** command and **tcp resequence-buffers vr-maximum** command enable you to limit the number of outstanding buffers on existing or newly established virtual routers.

tcp resequence-buffers default-vr-maximum

- Use to specify the default buffer limit assigned to all virtual routers when the virtual router is established.
- Specify a value of zero (0) to turn off the limit assignment.
- Example

  host1(config)#**tcp resequence-buffers default-vr-maximum 200**

- Use the **no** version to revert the virtual router maximum value to its default, 100 buffers.

tcp resequence-buffers vr-maximum

- Use to define the maximum number of buffers that the current or specified virtual router can use.
- Specify a value of zero (0) to turn off the limit assignment.
- Example

  host1(config)#**tcp resequence-buffers vr-maximum**

- Use the **no** version to revert the virtual router maximum value to its default, 100 buffers.

Limiting Buffers per Connection

The **tcp resequence-buffers connection-maximum** command and **tcp resequence-buffers default-connection-maximum** command enable you to limit the number of outstanding buffers on existing or newly established connections.

tcp resequence-buffers connection-maximum

- Use to define the maximum number of buffers that connections on the current or specified virtual router can use.
- Specify a value of zero (0) to turn off the connection maximum.
- Example

  host1(config)#**tcp resequence-buffers connection-maximum 50**

- Use the **no** version to revert the connection maximum value to its default, 10 buffers.

tcp resequence-buffers default-connection-maximum

- Use to specify the default buffer limit assigned to all TCP connections on a virtual router unless a specific limit is set for the VR in which the connection is established.
- Specify a value of zero (0) buffers to turn off the default limit.
- Example

  host1(config)#**tcp resequence-buffers default-connection-maximum 100**

- Use the **no** version to revert the connection maximum value to its default, 10 buffers.

Distributing Routing Table Updates to Line Modules

You can configure the forwarding table hold-down time allotted after a routing table change for the accumulation of additional updates and the subsequent distribution of the set of routing table changes to the line modules.

forwarding-table route-holddown

- Use to enhance SRP performance by increasing the hold-down time allotted for accumulating and distributing sets of routing table changes to the line modules.
- A higher timer value can enhance SRP performance, but it can also delay the implementation of routing table changes on the line modules. Be aware of the possible effect on network performance before you reconfigure the forwarding table hold-down timer.
- Setting the hold-down timer to zero (0) distributes an update after each change to the routing table, which can degrade SRP performance.
- Example

  host1(config)#**forwarding-table route-holddown 15**

- Use the **no** version to set the hold-down timer to the default value, 3 seconds.

IP Tunnel Routing Table

The IP tunnel routing tables include IPv4 routes that point only to tunnels, such as MPLS tunnels. The tunnel routing table is not used for forwarding. Instead, protocols resolve next hops by looking up the routes that point to tunnels. The routes in the tunnel routing table cannot be redistributed. See *JUNOSe BGP and MPLS Configuration Guide, Chapter 2, Configuring MPLS* for more information.

Shared IP Interfaces

You can create multiple *shared* IP interfaces over the same layer 2 logical interface—for example, atm 5/3.101—enabling more than one IP interface to share the same logical resources. You can configure one or more shared IP interfaces. Data sent over shared interfaces uses the same layer 2 interface. You can configure shared interfaces as you would unshared IP interfaces. Each shared interface has its own statistics.

Some layer 2 interfaces require a primary IP interface to negotiate certain IP parameters—for example, IPCP for PPP, ARP for Ethernet, and Inverse ARP for Frame Relay. If you do not configure a primary IP interface in such cases, the layer 2 interface cannot become operationally up.

A primary IP interface is the default interface for receiving data that arrives on the layer 2 interface. If you configure shared IP interfaces for the same layer 2 interface as your primary IP interface, by default data received on the layer 2 interface is received on the virtual router corresponding to the primary IP interface. A primary IP interface and all of its shared IP interfaces have the same interface location. You can configure a shared IP interface to receive data on the same layer 2 interface as a primary IP interface. You can delete primary and shared IP interfaces independently of each other.

You can create a primary IP interface as you do any other IP interface, as shown in the following example:

```
host1(config)#virtual-router vr-a:vrf-2
host1:vr-a:vrf-2:(config)#interface atm 5/3.101
host1:vr-a:vrf-2:(config-if)#ip address 10.1.1.1 255.255.255.255
host1:vr-a:vrf-2:(config-if)#exit
```

You do not have to configure a primary IP interface if you do not need one as described above. In the absence of a primary interface, you can still configure shared IP interfaces; however, in this scenario, data received on the layer 2 interface is discarded.

You cannot create shared IP interfaces for the following kinds of interface:

- IP floating interfaces (IP interfaces that stack over MPLS stacked tunnels)
- Loopback interfaces
- Null interfaces

For information about configuring shared IP interfaces to receive data on the same layer 2 interface as a primary IP interface, see *JUNOSe Broadband Access Configuration Guide, Chapter 25, Configuring Subscriber Interfaces*.

Configuring Shared IP Interfaces

To share IP interfaces:

1. Create a layer 2 interface.

   host1(config)#**interface atm 5/3**
   host1(config-if)#**interface atm 5/3.101**

2. (Optional) Create a primary IP interface.

   host1(config-if)#**ip address 10.1.1.1 255.255.255.255**
   host1(config-if)#**exit**

3. Create the shared IP interface.

   host1(config)#**interface ip si0**

4. Associate the shared IP interface with the layer 2 interface by one of the following methods:

   - Statically

     host1(config-if)#**ip share-interface atm 5/3.101**

   - Dynamically

     host1:vr-a:vrf-1(config-if)#**ip share-nexthop 10.0.0.1**

5. To fully configure the shared interface, assign an address (or make the interface unnumbered).

   host1(config-if)#**ip address 2.2.2.2 255.0.0.0**

interface ip

- Use to create an IP interface for interface sharing.
- Use the specified name to refer to the shared IP interface; you cannot use the layer 2 interface to refer to them, because the shared interface can be moved.
- Example

  host1(config)#**interface ip si0**

- Use the **no** version to delete the IP interface.

ip share-interface

- Use to specify the layer 2 interface used by a shared IP interface. The command fails if the layer 2 interface does not yet exist. The command is not supported (that is, it fails) if you use an RSVP tunnel (for example, **tunnel mpls:1**) to identify the layer 2 interface.
- If you issue this command on a shared IP interface, you cannot issue the **ip share-nexthop** command for the interface.
- After creating the shared IP interface, you can configure it as you do any other IP interface.

- The shared interface is operationally up when the layer 2 interface is operationally up.
- You can create operational shared IP interfaces in the absence of a primary IP interface.
- Example

  host1(config-if)#**ip share-interface atm 5/3.101**

- Use the **no** version to remove the association between the layer 2 interface and the shared IP interface. You can delete shared and primary IP interfaces independently.

ip share-nexthop

- Use to specify that the shared IP interface dynamically tracks a next hop. If the next hop changes, the shared IP interface moves to the new layer 2 interface associated with the IP interface toward the new next hop.
- If you issue this command on a shared IP interface, you cannot issue the **ip share-interface** command for the interface.
- If you issue this command on a shared IP interface, the shared interface cannot dynamically track the next hop for the specified destination if the next-hop IP address is resolvable over MPLS.
- If you specify a virtual router, the command fails if the VR does not already exist. If you do not specify a VR, the current VR is assumed.
- After creating the shared IP interface, you can configure it as you do any other IP interface.
- The shared interface is operationally up when the layer 2 interface associated with the specified next hop is operationally up. However, if the layer 2 interfaced associated with the specified next hop is an MPLS next hop (for example, an RSVP or LDP tunnel), the shared interface remains operationally down.
- Use the **no** version to halt tracking of the next hop.

Moving IP Interfaces

You can move an IP shared interface from one layer 2 interface to another by issuing the **ip share-interface** command to specify a different layer 2 interface. Moving an IP interface does not affect interface statistics, packets forwarded through the interface, or policies attached to the IP interface.

Example The following commands create shared interface si0 on the layer 2 interface atm5/3.101:

```
host1(config)#virtual-router vr-a:vrf-1
host1:vr-a:vrf-1(config)#interface ip si0
host1:vr-a:vrf-1(config-if)#ip share-interface atm 5/3.101
host1:vr-a:vrf-1(config-if)#exit
```

The following commands move shared interface si0 to the layer 2 interface atm5/3.201:

```
host1:vr-a:vrf-1(config)#interface ip si0
host1:vr-a:vrf-1(config-if)#ip share-interface atm 5/3.201
```

IP Shared Interface Statistics

Each shared interface has its own statistics. Packets transmitted on a shared IP interface are always counted only in the shared IP interface.

Subscriber Interfaces

A subscriber interface is an extension of a shared IP interface. Shared IP interfaces are unidirectional—they can transmit but not receive traffic. In contrast, subscriber interfaces are bidirectional—they can both receive and transmit traffic.

For details about configuring and using subscriber interfaces, see *JUNOSe Broadband Access Configuration Guide, Chapter 25, Configuring Subscriber Interfaces*.

Internet Control Message Protocol

IP was not designed to provide reliable delivery service. The higher-layer protocols that operate as clients of IP implement their own reliability procedures if reliable communications are required.

Internet Control Message Protocol (ICMP) provides a mechanism that enables a router or destination host to report an error in data traffic processing to the original source of the packet. ICMP messages provide feedback about problems that occur in the communication environment.

ICMP messages are sent only when errors occur in either the processing of an unfragmented data packet or the first fragment of a fragmented data packet.

ICMP messages are encapsulated as part of the data portion of an IP data packet and are routed like any other IP data packets. Thus, there is no guarantee to the sender of an ICMP message that the message will be delivered to its destination.

The router supports ICMP redirects. When a packet enters an IP interface and exits the same interface, the router may send an ICMP message to the originator of the packet. This message notifies the originator that a better gateway exists to the assigned destination address.

With the **ip redirects** command (used in Interface Configuration mode) you can enable or disable ICMP redirects. This attribute is enabled by default. If it is enabled on the IP interface and if the internal ICMP redirect queue is not full, the router sends an ICMP redirect packet to the originator. When the originator receives the ICMP redirect notification, the originator determines whether to start using the better gateway.

ICMP Tasks

You can enable the following ICMP features:

- ICMP Router Discovery Protocol (IRDP)
- ICMP netmask reply
- Sending of IP redirects
- Generation of ICMP unreachable messages

ip irdp

- Use to enable IRDP processing on an interface.
- Example

  host1(config-if)#**ip irdp**

- Use the **no** version to disable the function.

ip mask-reply

- Use to enable ICMP netmask reply.
- Example

  host1(config-if)#**ip mask-reply**

- Use the **no** version to disable the function.

ip redirects

- Use to enable the sending of redirect messages if software is forced to resend a packet through the same interface on which it was received.
- Example

  host1(config-if)#**ip redirects**

- Use the **no** version to disable the sending of redirect messages.

ip unreachables

- Use to enable the generation of an ICMP unreachable message when a packet is received that the router cannot deliver.
- Example

  host1(config-if)#**ip unreachables**

- Use the **no** version to disable the function.

Specifying a Source Address for ICMP Messages

By default, ICMP uses the IP address of the outgoing interface as the source IP address for the ICMP message. However, you can use the **ip icmp update-source** command to instruct ICMP to use an already configured interface or a specified IP address, as the source address of the ICMP message.

For example, you can specify that ICMP use Fast Ethernet interface 1 in slot 0 as the source for any messages that it sends:

host1(config)#**ip icmp update-source fastEthernet 0/1**

You must use an already configured interface or an existing IP address when using the **ip icmp update-source** command. Also, you cannot specify a multicast address when using this command.

ip icmp update-source

- Use to define an update source address for all ICMP messages that the E-series router generates from the specified interface.
- Example

  host1(config)#**ip icmp update-source 192.35.42.1**

- Use the **no** version to disable the function.

Reachability Commands

Use the **ping** and **traceroute** commands to determine reachability of destinations in the network.

- Use the **ping** command to send an ICMP or ICMPv6 echo request packet. In the following example, the request packet is sent to IP address 192.35.42.1, with a packet count of 10 and a timeout value of 10 seconds:

  host1#**ping 192.35.42.1 10 timeout 10**

- Use the **traceroute** command to discover routes that router packets follow when traveling to their destination. In the following example, the trace destination IP address is 192.56.20.1, the maximum number of hops of the trace is 20, and the timeout value is 10 seconds:

  host1#**traceroute 192.56.20.1 20 timeout 10**

ping

- Use to send an ICMP or ICMPv6 echo request packet to the IP address that you specify.
- You can specify a VRF context.
- Use the **source interface** keywords to specify a source interface other than the one from which the probe originates.
- Use the **source address** keywords to specify a source IP address other than the one from which the probe originates.
- You can specify the following options:
  - *packetCount*—Number of packets to send to the destination IP address. If you specify a zero (0), echo requests packets are sent indefinitely.
  - **data-pattern**—Sets the type of bits contained in the packet to all ones, all zeros, a random mixture of ones and zeros, or a specific hexadecimal data pattern that can range from 0x0–0xFFFFFFFF. The default is all zeros.
  - **data-size**—Sets the number of bytes comprising the IP packet and reflected in the IP header in the range 0–64000; the default is 100 bytes.
  - **extended** header attributes—Set the following:
    - A value to be set in the type of service (ToS) byte, in the range 0–255, to support quality of service (QoS) offerings
    - Don't-fragment bit to prevent IP from fragmenting the packet if it is too long for the MTU of a given link; if the nonfragmented packet cannot be delivered, it is discarded.
    - Strict-source or loose-source routing, including the IP address of the hops the packets must traverse. For loose-source-route, you specify some or all of the hops, but they do not have to be adjacent. For strict-source-route, you must specify every adjacent hop through which the packet must traverse.
    - The IP addresses to be recorded for a specified number of routers that the packets traverse.
    - The time that a packet traverses a router to be recorded for a specified number of routers.
    - An interface type and specifier of a destination address on the router that is configured for external loopback; the command succeeds only if the specified interface is configured for external loopback.
    - The traffic class value to match in the Traffic Class field of each packet (IPv6 only)
    - The flow label value to match in the Flow Label field of each packet (IPv6 only)
  - **sweep-interval**—Specifies the change in the size of subsequent ping packets while sweeping across a range of sizes. For example, you can configure the sweep interval to sweep across the range of packets from 100 bytes to 1000 bytes in increments equal to the sweep interval. By default the router increments packets by one byte; for example, it sends 100, 101, 102, 103, ... 1000. If the sweep interval is 5, the router sends 100, 105, 110, 115, ... 1000.

- **sweep-sizes**—Enables you to vary the sizes of the echo packets being sent. This capability is useful for determining the minimum sizes of the MTUs configured on the nodes along the path to the destination address. Determining the minimum size reduces packet fragmentation, which contributes to performance problems. The default is not to sweep (all packets are the same size).
- **timeout**—Sets the number of seconds to wait for an ICMP echo reply packet before the connection attempt times out.
- **ttl**—Sets the time-to-live hop count in the range 1–255; the default is 32.

- The following characters can appear in the display after issuing the **ping** command:
  - !—Reply received
  - .—Timed out while waiting for a reply
  - ?—Unknown packet type
  - A—Address mask request message
  - a—Address mask reply message
  - D—Router discovery advertisement message
  - d—Router discovery request message
  - H—Host unreachable
  - I—Information request message
  - i—Information reply message
  - L—TTL expired message
  - M—Could not fragment, DF bit set
  - m—Parameter problem message
  - N—Network unreachable
  - P—Protocol unreachable
  - Q—Source quench
  - r—Redirect message
  - T—Timestamp request message
  - t —Timestamp reply message
  - U—Destination unreachable
- Example

  host1(config)#**interface serial 5/2:1/1**
  host1(config-if)#**ip address 172.16.1.1 255.255.255.0**
  host1(config-if)#**exit**
  host1#**ping 172.16.1.1 extended interface serial 5/2:1/1**

- There is no **no** version.

traceroute

- Use to discover the routes that router packets follow when traveling to their destination.
- You can specify:
  - A VRF context
  - Destination IP or IPv6 address
  - Source interface for each of the transmitted packets
  - Source address for each of the transmitted packets
  - Maximum number of hops of the trace and a timeout value
  - Size of the IP packets (not the ICMP payload) in the range 0–64000 bytes sent with the **traceroute** command. Including a size might help locate any MTU problems that exist between your router and a particular device.
  - Extended IP header attributes, including the ToS byte (IP only), whether to set the DF bit for the transmitted packets (IP only), the traffic class (IPv6 only), and flow label (IPv6 only).
- You can also force transmission of the packets on a specified interface regardless of what the IP address lookup indicates.
- Example

  host1#**traceroute 172.20.13.1 20 timeout 10**

- There is no **no** version.

Response Time Reporter

The Response Time Reporter (RTR) feature enables you to monitor network performance and resources by measuring response times and the availability of your network devices.

RTR configuration is associated with a specific virtual router, distinct from any other virtual router.

Configuration Tasks

To configure RTR:

1. Configure the probe type—an echo probe or a path echo probe.
2. (Optional) Configure probe characteristics:
   - frequency
   - hops-of-statistics-kept (path echo)
   - max-response-failure (path echo)
   - operations-per-hop (path echo)
   - owner

- receive-interface
- request-data-size
- samples-of-history-kept
- tag
- timeout (echo)
- tos

NOTE: You cannot set any of these characteristics until you have set the probe type. The default values of these characteristics depend on the type of the entry.

3. (Optional) Set reaction conditions.
4. Schedule the probe.
5. (Optional) Capture statistics and collect error information.
6. (Optional) Collect history.

Configuring the Probe Type

To begin configuring RTR, enter RTR Configuration mode and configure the probe type—either an *echo* probe or a *path echo* probe.

rtr

- Use to configure an RTR probe and to enter RTR Configuration mode.
- Example

  host1(config)#**rtr 1**

- Use the **no** version to delete all configuration information for an RTR probe.

type

- Use to set an echo or path echo probe:
  - **echo**—Limited to end-to-end RTR operations; corresponds to SNMP ping
  - **pathEcho**—Finds a path to the destination and echoes each device in the path; corresponds to SNMP traceroute
- You must specify this value before any other.
- If you change the type for an existing RTR entry, all values are reset, including the administrative status. There is no default value.
- More than one RTR entry can become active, provided each entry's target address is unique.

- If you configure multiple RTR entries to use the same target address, you must issue the **receive-interface** command to specify the interface on which the RTR probe expects to receive responses. (For information, see *Setting the Receiving Interface* on page 68.)
- If you use a target address already configured for another RTR entry that is active, the test will not run if both entries are in the same virtual router. If they are in distinct virtual routers, however, there is no restriction.
- Example

  host1(config-rtr)#**type echo protocol ipIcmpEcho 10.10.0.9**

- Use the **no** version to remove the type configured for the probe.

Configuring Optional Characteristics

In addition to configuring the probe's type, you can configure the probe characteristics presented in Table 8.

Table 8: Probe Characteristics

Characteristic	Description
frequency	Time between tests (in seconds)
hops-of-statistics-kept	Hops per path for which statistics are gathered
max-response-failure	Maximum number of consecutive failures
operations-per-hop	Number of probes per hop
owner	Owner of the probe
receive-interface	Interface on which the probe expects to receive responses
request-data-size	Request's payload size
samples-of-history-kept	Maximum number of history samples
tag	User-defined tag
timeout	Probe timeout (in milliseconds)
tos	A value for the TOS byte

frequency

- Use to set the rate (in seconds) that the RTR probe uses to start a response time operation.
- Example

  host1(config-rtr)#**frequency 90**

- Use the **no** version to return to the default value, 60 seconds.

operations-per-hop

- Use to set the number of RTR probe operations sent to a given hop.
- You can apply this option only to a pathEcho type.
- Example

  host1(config-rtr)#**operations-per-hop 5**

- Use the **no** version to return to the default, 3.

owner

- Use to identify the owner of the probe.
- If the SNMP agent is the owner of the probe, the owner's name can begin with *agent*.
- Example

  host1(config-rtr)#**owner 192.10.27.6 rtc.boston.com 555.1212**

- Use the **no** version to return to the default, no owner.

request-data-size

- Use to set the protocol data size, in bytes, in the request packet.
- Example

  host1(config-rtr)#**request-data-size 20**

- Use the **no** version to return to the default value, 1 byte.

tag

- Use to set an identifier for the probe.
- Example

  host1(config-rtr)#**tag westford**

- Use the **no** version to return to the default, no tag.

timeout

- Use to set the time (in milliseconds) that the probe waits for a response.
- You can apply this option only to an echo type.
- Do not set the value for timeout to more than the value set for frequency. If you do, the timeout value is ignored.
- If you set the timeout to 0, no timeout is set.
- Example

  host1(config-rtr)#**timeout 3000**

- Use the **no** version to return to the default value, 5000 milliseconds.

tos

- Use to set the type of service (ToS) byte in the probe's IP header.
- Example

  host1(config-rtr)#**tos 16**

- Use the **no** version to return to the default value, 0. The default applies to both the echo and pathEcho types.

Capturing Statistics

The primary objective of RTR is to collect statistics and information about network performance. You can control the number and type of statistics collected.

hops-of-statistics-kept

- Use to set the number of hops per path for which statistics are collected.
- When the number of hops reaches the specified number (that is, *size*), no additional statistical information about the path is stored.
- This option applies only to pathEcho entries.
- To turn off this feature, set the value to 0.
- Example

  host1(config-rtr)#**hops-of-statistics-kept 5**

- Use the **no** version to set the default, 16 hops.

max-response-failure

- Use to set the maximum number of consecutive failures to respond to a probe's request.
- When the maximum number is reached, the test stops.
- This option applies only to pathEcho entries.
- To turn off this feature, set the value to 0.
- Example

  host1(config-rtr)#**max-response-failure 2**

- Use the **no** version to set the default, 5 consecutive failures.

Collecting History

RTR can collect data samples for a given probe. These samples are referred to as history data. When RTR collects history, it refers to tests. A test is the lifetime of a probe operation.

samples-of-history-kept

- Use to set the maximum number of entries in the history table for each RTR probe.
- This command enables you to control the number of samples saved in the history table.
- If you set the number of samples to 0, no samples are kept.
- Because collecting history increases memory usage, do so only when there is a problem in your network.
- Example

  host1(config-rtr)#**samples-of-history-kept 5**

- Use the **no** version to set the default, 16 hops for pathEcho type, 1 hop for echo type.

Setting the Receiving Interface

When you configure multiple RTR entries to use the same target address, you must issue the **receive-interface** command to set the interface on which the probe expects to receive responses. This action enables the router to map incoming responses to the proper RTR entry, even when multiple RTR entries have the same target address.

receive-interface

- Use to specify the interface on which the RTR probe expects to receive responses.
- You must set this attribute when multiple RTR entries are configured to use the same target address.
- Example

  host1(config-rtr)#**receive-interface fastEthernet 3/0**

- Use the **no** version to restore the default value, which is to receive a response on any interface.

Setting Reaction Conditions

You can set the RTR probe to react to events that take place and to send notifications about these events.

NOTE: The only **no** version for all the **rtr reaction-configuration** commands is **no rtr reaction-configuration** *rtrIndex*. Use the **no** version to clear all traps. This works for all the options.

rtr reaction-configuration action-type

- Use to specify the type of actions to occur depending on the events controlled by RTR.
- The default is to take the traps of enabled events.
- Example

  host1(config)#**rtr reaction-configuration 1 action-type trapOnly**

- There is no **no** version.

rtr reaction-configuration operation-failure

- Use to enable the operation-failure reaction.
- The operation-failure event is triggered when a number of consecutive probe operations are not received or when they are received after a timeout.
- Example

  host1(config)#**rtr reaction-configuration 1 operation-failure 3**

- There is no **no** version.

rtr reaction-configuration path-change

- Use to enable the path-change reaction.
- The path-change event is triggered when a change is detected in the hop table. At most, there can be one such event per test.
- Example

  host1(config)#**rtr reaction-configuration 1 path-change**

- There is no **no** version.

rtr reaction-configuration test-completion

- Use to enable test-completion reaction.
- The test-completion event is triggered when a test is completed successfully.
  - For echo, a successful test means that all probes were sent.
  - For pathEcho, a successful test means that the destination was reached at least once.
- At most, there can be one such event per test.
- Example

  host1(config)#**rtr reaction-configuration 1 test-completion**

- There is no **no** version.

rtr reaction-configuration test-failure

- Use to enable test-failure reaction.
- The test-failure event is triggered when a test fails. Failure is determined in the following ways:
  - If Echo, this event is triggered after testFailureValue probes are either not received or are received after a timeout.
  - If PathEcho, this event is triggered when the test ends and no responses are received from the destination.
- At most, there can be one such event per test.
- Example

  host1(config)#**rtr reaction-configuration 1 test-failure**

- There is no **no** version.

Scheduling the Probe

When you have configured the RTR probe, you must schedule the operation to begin collecting statistics and other information about problems that may arise.

rtr schedule

- Use to create an RTR schedule.
- Example

  host1(config)#**rtr schedule 5**

- Use the **no** version to stop the test. The **no** version stops the probe operation by putting it in the pending state. The **no** version also resets the restart-time attribute and the life attribute.

rtr schedule life

- Use to schedule the test's length.
- Life is a value that depends on the type of the RTR entry; it is not a length of time.
  - If the type is echo, life relates to the number of probes sent until a test finishes.
  - If the type is pathEcho, life relates to the maximum number of hops used by the traceRoute trap.
- Example

  host1(config)#**rtr schedule 5 life 1800**

- Use the **no** version to stop the test. The **no** version stops the probe operation by putting it in the pending state. The **no** version also resets the life attribute.

rtr schedule restart-time

- Use to specify a restart time, in seconds, after which a test is restarted.
- Example

  host1(config)#**rtr schedule 5 restart-time 15**

- Use the **no** version to stop the test. The **no** version stops the probe operation by putting it in the pending state. The **no** version also resets the restart-time attribute.

rtr schedule start-time

- Use to schedule a test's starting time (now or pending).
- Example

  host1(config)#**rtr schedule 5 start-time now**

- Use the **no** version to stop the test. The **no** version stops the probe operation by putting it in the default state, pending.

Shutting Down the Probe

You can shut down the RTR probe operation.

rtr reset

- Use to shut down the RTR, stop all probe operations, and clear the RTR configuration for the given virtual router.

NOTE: We recommend that you use this command only in extremely serious situations, such as problems with the configurations of a number of probe operations.

- Example

  host1(config)#**rtr reset**

- Use the **no** version to negate the reset operation.

Monitoring RTR

You can monitor RTR by displaying status and configuration information.

show rtr application

- Use to display global information about RTR.
- Field descriptions
  - numberOfEntries—Number of RTR entries according to type
  - entriesEnabled—RTR entries with administrative status enabled
  - entriesActive—RTR entries with operational status enabled
- Example

```
host1#show rtr application

          numberOfEntries      entriesEnabled     entriesActive
        ------------------   -----------------  ---------------
echo                    1                   1                1
pathEcho                1                   1                1
total                   2                   2                2
```

show rtr collection-statistics

- Use to display statistical information for a particular probe operation or for all operations.
- Field descriptions
  - rtrIndex—Index number of the RTR probe
  - operationsSent—Number of probe operations sent
  - operationsRecvd—Number of probe operations received
  - lastGoodResponse—Time when last valid probe operation was received
  - operStatus—Operational status of the probe: enabled, disabled
  - minRtt—Minimum round-trip time in milliseconds
  - maxRtt—Maximum round-trip time in milliseconds
  - avgRtt—Average round-trip time in milliseconds
  - rttSumSqr—Sum of the square of all round-trip times in milliseconds
  - testAttempts—Number of times the test ran
  - testSuccesses—Number of times the test ran successfully
  - currentHop—Current hop (TTL) used in the test
  - currentOperation—Current probe operation index sent to the hop
- Example

```
host1#show rtr collection-statistics

  Echo Entries:

  rtrIndex  operationsSent  operationsRcvd  lastGoodResponse
----------  --------------  --------------  ----------------
     1           5208            5187       08/30/2000 05:09
```

```
  rtrIndex  operStatus  minRtt  maxRtt  avgRtt  rttSumSqr
----------  ----------  ------  ------  ------  ---------
     1          enabled   0      1785     3     7109208

  PathEcho Entries:

  rtrIndex  testAttempts  testSuccesses  lastGoodResponse
----------  ------------  -------------  ----------------
     2           156           156       08/30/2000 05:09

  rtrIndex  operStatus  currentHop  currentOperation
----------  ----------  ----------  -----------------
     2       enabled        2             4
```

show rtr configuration

- Use to display the configuration for a particular probe or for all probes.
- Field descriptions
  - rtrIndex—Index number of the RTR probe
  - type—Probe type: echo, pathEcho
  - targetAddress—Address of the probe's target
  - reqSize—Protocol data size in the request packet
  - freq—Rate in seconds that the RTR probe uses to start a response time operation
  - life—Length of the test
  - source—Interface from which the probe is sent
  - restartTime—Restart time of the test in seconds
  - owner—Owner of the probe
  - samples—Maximum number of entries saved in the history table for this RTR probe
  - admin—Administrative status of the probe: enabled, disabled
  - tos—Setting of the type of service (ToS) byte in the probe's IP header
  - reactionConfiguration—RTR reactions that are configured for the probe
  - receiveInterface—Type and specifier of the interface on which the probe expects to receive responses; this field is blank if the optional **receive-interface** characteristic is not configured
  - operFail—Operation failure event is triggered when this number of consecutive probe operations is not received or when the operations are received after a timeout
  - testFail—Test failure event is triggered when this number of probe operations is not received or when the operations are received after a timeout
  - timeout—Time in milliseconds that the probe waits for a response
  - tag—Identifier configured for the probe

- operPerHop—Number of RTR probe operations sent to a given hop
- maxFail—Maximum number of consecutive failures to respond to a probe's request. When the maximum number is reached, the test stops. Applies only to pathEcho entries.
- hopKpt—Number of hops per path for which statistics are collected. When this number is reached, no additional statistical information about the path is stored. Applies only to pathEcho entries.

- Example

```
host1#show rtr configuration
  rtrIndex     type   targetAddress reqSize freq   life
  -------  --------   ------------  ------  ------ -----
      1       echo    10.5.0.200      1     1       20
      2   pathEcho    10.5.0.11       1     1       30

  rtrIndex     source             restartTime  owner
  ----------  ------------------  -----------  ----------
      1       fastEthernet0/0         10
      2                               60

  rtrIndex    samples  admin     tos  reactionConfiguration
  ----------  -------  --------  ---  -------------------------
      1          5     enabled    0
      2          5     enabled    0

  rtrIndex    receiveInterface
  ----------  ----------------
      1       fastEthernet0/0

  rtrIndex    operFail  testFail  timeout  tag
  ----------  --------  --------  -------  ------
      1          1         1      10000

  rtrIndex  operPerHop  maxFail  hopKpt  tag
----------  ----------  -------  ------  ------
      2          5         3      16
```

show rtr history

- Use to display history (data samples) for a particular probe or for all probes.
- Field descriptions
  - rtrIndex—Index number of the RTR probe
  - operation—Index number of the probe operation
  - rtt—Round-trip time in milliseconds
  - statusDescription
    - concurrentLimitFail—Target already being used by another rtrIndex
    - ifInactiveToTarget—Interface used to reach target is not operational
    - invalidHostAddress—Target address is not supported
    - noRouteToTarget—Target address is not reachable
    - responseReceived—Probe operation replied by target

  - requestTimedOut—Probe operation not replied to by target or reply received after timeout
  - unknownDestAddress—Target address is invalid
  - unableToResolveName—Target address could not be looked up
- timeStamp—Date and time when the RTR entry was created
- test—Index number of the pathEcho test
- hop—Index number of the hop count
- operation—Index number of the probe operation
- address—Address of router at the hop

- Example

```
host1#show rtr history
 Echo Entries:

  rtrIndex   operation  rtt   statusDescription timeStamp
  --------   ---------  ---   ----------------- ---------
         1        5476  0     responseReceived  08/30/2000 05:17
         1        5477  0     responseReceived  08/30/2000 05:17
         1        5478  0     responseReceived  08/30/2000 05:17
         1        5479  0     responseReceived  08/30/2000 05:17
         1        5480  0     responseReceived  08/30/2000 05:17

  PathEcho Entries:

  rtrIndex    test    hop   operation  rtt     statusDescription
  ----------  ------  ----  ---------  ------  -----------------
           2     165     3          5       0  responseReceived
           2     165     3          1       0  responseReceived
           2     165     3          2       0  responseReceived
           2     165     3          3       0  responseReceived
           2     165     3          4       0  responseReceived

 rtrIndex    test    hop   operation  timeStamp         address
 ----------  ------  ----  ---------  ----------------  --------
          2     165     3          5  08/30/2000 20:39  10.5.0.11
          2     165     3          1  08/30/2000 20:40  10.5.0.11
          2     165     3          2  08/30/2000 20:40  10.5.0.11
          2     165     3          3  08/30/2000 20:40  10.5.0.11
          2     165     3          4  08/30/2000 20:40  10.5.0.11
```

show rtr hops

- Use to display RTR hops information.
- Field descriptions
  - rtrIndex—Index number of the RTR probe
  - hop—Index number of the hop count
  - address—Address of the router at the hop
  - minRtt—Minimum round-trip time in milliseconds
  - maxRtt—Maximum round-trip time in milliseconds
  - avgRtt—Average round-trip time in milliseconds
  - rttSumSqr—Sum of the square of all round-trip times in milliseconds

- operationsSent—Number of probe operations sent
- operationsRcvd—Number of probe operations received
- lastGoodResponse—Time when last valid probe operation was received

- Example

```
host1#show rtr hops

  rtrIndex    hop   address     minRtt  maxRtt  avgRtt  rttSumSqr
  ----------  ----  ----------- ------  ------  ------  --------
           2     1  192.168.1.1      1     276       1     955363
           2     2  10.2.0.3         0    1109       2   10094451

  rtrIndex hop operationsSent operationsRcvd  lastGoodResponse
  -------- --- -------------- --------------  ------------
         2   1          36985          36838  09/18/2000 20:20
         2   2          30717          21494  09/18/2000 20:20
```

show rtr operational-state

- Use to display RTR operational information.
- Field descriptions
  - rtrIndex—Index number of the RTR probe
  - type—Type of RTR probe: echo, pathEcho
  - entryStatus—If the entry was created via the SNMP DISMAN MIB, the row may be partially constructed; if that is the case, the CLI displays notReady as the entry's status
  - adminStatus—Derived from the **rtr schedule start-time** command; if the option is **now**, the status is enabled; if the option is **pending**, the status is disabled
  - operStatus—Enabled only if entryStatus and adminStatus are enabled and the test is running; operStatus remains enabled if the test finishes and restart time is not 0
- Example

```
host1#show rtr operational-state
 rtrIndex    type      entryStatus  adminStatus  operStatus
 ----------  --------  -----------  ----------- ----------
          1  echo      active       enabled      enabled
          2  pathEcho  active       enabled      enabled
```

Monitoring IP

This section explains how to set a statistics baseline and use the **show** commands to view your IP configuration and monitor IP interfaces and statistics.

System Event Logs

To troubleshoot and monitor IP, use the following system event logs:

- ipAccessList—IP access list matching
- ipEngine—IP chassis manager
- ipGeneral— IP general information
- ipIfCreator—IP interface creator events
- ipInterface—IP interface events
- ipNhopTrackerGeneral—Next-hop tracker for IP shared interfaces
- ipProfileMgr—IP profile manager events
- ipRouteMgr— IP routing policy events
- ipRouteTable—IP routing table events
- ipTraffic—IP frame transmit and receive events
- ipTunnel—IP tunnel events

For more information about using event logs, see the *JUNOSe System Event Logging Reference Guide*.

Establishing a Baseline

IP statistics are stored in system counters. The only way to reset the system counters is to reboot the router. You can, however, establish a baseline for IP statistics by setting a group of reference counters to zero.

baseline ip

- Use to set a statistics baseline for IP statistics. Baselining is not supported for IP socket statistics.
- The router implements the baseline by reading and storing the statistics at the time the baseline is set and then subtracting this baseline whenever baseline-relative statistics are retrieved.
- Use the **delta** keyword with IP **show** commands to specify that baselined statistics are to be shown.
- Example

  host1#**baseline ip**

- There is no **no** version.

baseline ip udp

- Use to set a statistics baseline for UDP statistics.
- The router implements the baseline by reading and storing the statistics at the time the baseline is set and then subtracting this baseline whenever baseline-relative statistics are retrieved.
- Use the **delta** keyword with IP **show** commands to specify that baselined statistics are to be shown.
- Example

  host1#**baseline ip udp**

- There is no **no** version.

baseline tcp

- Use to set a statistics baseline for all (both IPv4 and IPv6) TCP statistics or for only IPv4 or IPv6 statistics.
- The router implements the baseline by reading and storing the statistics at the time the baseline is set and then subtracting this baseline whenever baseline-relative statistics are retrieved.
- Use the **ip** keyword to implement a baseline for only IPv4 statistics.
- Use the **delta** keyword with IP **show** commands to specify that baselined statistics are to be shown.
- Example 1

  host1#**baseline tcp**

- Example 2

  host1#**baseline ip tcp**

- There is no **no** version.

IP show Commands

You can monitor the following aspects of IP using **show ip** commands:

To Display	Command
Access lists	**show access-list** **show ip as-path-access-list**
ARP	**show arp**
General IP information	**show ip**
IP addresses	**show ip address**
Community lists	**show ip community-list**
Routing table	**show ip forwarding-table slot** **show forwarding-table route-holddown**
Interfaces	**show ip interface**
Shared IP interfaces	**show ip interface shares**

To Display	Command
Protocols	**show ip protocols**
Redistribution policies	**show ip redistribute**
Routes	**show ip route**
Interfaces and next hops	**show ip route slot**
Socket statistics	**show ip socket statistics**
Static routes	**show ip static**
TCP ACK, RST, and SYN protection status	**show tcp ack-rst-and-syn**
Black hole threshold information	**show tcp path-mtu-discovery**
TCP statistics	**show tcp statistics**
Traffic	**show ip traffic**
UDP statistics	**show ip udp statistics**
Profiles	**show ip profile**
Route maps	**show route-map**

To set a statistics baseline for IP interfaces, use the **baseline tcp** and **baseline ip udp** commands. Use the **delta** keyword with IP **show** commands to specify that baselined statistics are to be shown.

You can use the output filtering feature of the **show** command to include or exclude lines of output based on a text string that you specify. See *JUNOSe System Basics Configuration Guide, Chapter 2, Command-Line Interface*, for details.

show access-list

- Use to display information about access lists, including the instances of each access list.
- Example

```
host1#show access-list
IP Access List 1:
    permit ip 172.31.192.217 0.0.0.0 0.0.0.0 255.255.255.255
    permit ip 12.40.0.0 0.0.0.3 0.0.0.0 255.255.255.255
    deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
IP Access List 2:
    permit ip 172.19.0.0 0.0.255.255 0.0.0.0 255.255.255.255
    deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
IP Access List 10:
    permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
IP Access List 11:
    deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
```

show arp

- Use to display information about ARP.
- Field descriptions
  - Address—IP address of the entry
  - Age—Time to live for this entry in seconds
  - Hardware Addr—Physical (MAC) address of the entry
  - Interface—Interface-specifier of the entry (for example, fastEthernet6/0 is an Ethernet interface on slot 6, port 0)
  - *—Indicates that an ARP entry was added because of an **arp validate** command, rather than just an **arp** command.
- Example

```
host1#show arp
        Address    Age   Hardware Addr  Interface
 172.31.192.217   21340  00d0.58f2.67e0 loopback1
    192.168.1.0   20730  00e0.09ed.5312 fastEthernet6/0 *
    192.168.1.1   12550  00e0.b06a.4c75 fastEthernet6/0 *
  192.168.1.217   21600  0090.1a00.0230 fastEthernet6/0 *
  192.168.1.255   21600  00f0.c2d1.1200 fastEthernet6/0 *
      12.40.0.2   24320  0020.6393.4233 atm5/0.1
     172.18.2.1   21600  0020.bed2.8738 atm5/1.1
     172.18.2.2   21600  0020.5b91.60f2 atm5/1.1
 172.31.192.206   21600  00d0.43b5.1032 atm5/1.1
```

show forwarding-table route-holddown

- Use to display the configured hold-down time allotted after an initial routing table change for the accumulation and subsequent distribution of a set of routing table updates to the line modules. The default value is 3 seconds; the range of values is 0–30 seconds.
- A higher hold-down setting can enhance SRP performance; however, a higher setting can also delay the implementation of routing table changes on the line modules.
- A hold-down timer value of zero (0) distributes an update after each change to the routing table.
- Example

```
host1#show forwarding-table route-holddown
  Hold-down timer value is 3 seconds.
```

show ip

- Use to display general information about IP.
- Field descriptions
  - IP Router Id—Router ID number
  - Router Name—Router name
  - Default TTL—Default IP TTL (time-to-live) value
  - Reassemble Timeout—Amount of time (in minutes) IP waits for missing packet fragments before it drops the fragments it is holding
  - SA Validate Trap—Whether the source address validation trap is enabled
- Example

```
host1#show ip
  IP Router Id: 192.168.1.155
  Router Name: default
  Default TTL: 60
  Reassemble Timeout: 30
  SA Validate Trap: false
```

show ip address

- Use to display detailed or summary information about a particular IP interface.
- Specify a VRF name to view information for only that VRF.
- Use the **brief** keyword to display summary information about the interface.
- Use the **detail** keyword to display detailed information about the interface.
- Field descriptions
  - Network Protocols—Network protocols configured on this interface
  - Internet address—IP address and subnet mask of this interface
  - Broadcast address—Broadcast address of this interface
  - Operational MTU—MTU of this interface
  - Administrative MTU—Value of the MTU if it has been administratively overridden using the configuration
  - Operational speed—Speed of the interface
  - Administrative speed—Value of the speed if it has been administratively overridden using the configuration
  - Discontinuity Time—Value of the SysUpTime when the interface statistics last started being valid
  - Router advertisement—Status of router discovery advertisement: enabled, disabled
  - Proxy Arp—Status of the feature: enabled, disabled
  - Administrative debounce-time—Configured debounce behavior, enabled or disabled. If enabled, indicates time in milliseconds that the router waits before generating an up or down event in response to a state change in the interface. If the state changes back before the debounce timer expires, no state change is reported.

- Operational debounce-time—Current debounce behavior, enabled or disabled. If enabled, indicates time in milliseconds that the router waits before generating an up or down event in response to a state change in the interface. If the state changes back before the debounce timer expires, no state change is reported.
- Access routing—Access route addition: enabled, disabled
- Multipath mode—Equal cost multipath mode method: hashed, round-robin
- In Received Packets, Bytes—Total number of packets and bytes received on this interface
  - Unicast Packets, Bytes—Unicast packets and bytes received on the IP interface; link-local received multicast packets (non-multicast-routed frames) are counted as unicast packets
  - Multicast Packets, Bytes—Multicast packets and bytes received on the IP interface which are then multicast-routed are counted as multicast packets
- In Policed Packets, Bytes—Packets and bytes that were received and dropped because of rate limits
- In Error Packets—Number of packets received with errors
- In Invalid Source Address Packets—Packets received with invalid source address (for example, spoofed packets)
- In Discarded Packets—Packets received that were discarded for reasons other than rate limits, errors, and invalid source address
- Out Forwarded Packets, Bytes—Total number of packets and bytes that were sent from this interface
  - Unicast Packets, Bytes—Unicast packets and bytes that were sent from this interface
  - Multicast Routed Packets, Bytes—Multicast packets and bytes that were sent from this interface
- Out Scheduler Drops Committed Packets, Bytes—Outgoing packets and bytes dropped by the scheduler even though they had a committed traffic contract
- Out Scheduler Drops Conformed Packets, Bytes—Outgoing packets and bytes dropped by the scheduler even though they conformed to the traffic contract
- Out Scheduler Drops Exceeded Packets, Bytes—Outgoing packets and bytes that were dropped by the scheduler because they exceeded the contract
- Out Policed Packets, Bytes—Outgoing packets and bytes dropped because of rate limiters
- Out Discarded Packets—Outgoing packets that were discarded for reasons other than those dropped by the scheduler and those dropped because of rate limits

- Example

```
host1#show ip address 10.6.136.73
fastEthernet0/0 is up, line protocol is up
  Network Protocols: IP
  Internet address is 10.6.136.73/255.255.128.0
  Broadcast address is 255.255.255.255
  Operational MTU = 0  Administrative MTU = 0
  Operational speed = 1  Administrative speed = 0
  Discontinuity Time = 5766
  Router advertisement = disabled
  Proxy Arp = disabled
  Administrative debounce-time = 10 mSecs
  Operational debounce-time    = disabled
  Access routing = disabled
  Multipath mode = hashed

  In Received Packets 2849, Bytes 759428
    Unicast Packets 2849, Bytes 759428
    Multicast Packets 0, Bytes 0
  In Policed Packets 0, Bytes 0
  In Error Packets 0
  In Invalid Source Address Packets 0
  In Discarded Packets 0
  Out Forwarded Packets 1866, Bytes 84650
    Unicast Packets 1866, Bytes 84650
    Multicast Routed Packets 0, Bytes 0
  Out Scheduler Drops Committed Packets 0, Bytes 0
  Out Scheduler Drops Conformed Packets 0, Bytes 0
  Out Scheduler Drops Exceeded  Packets 0, Bytes 0
  Out Policed Packets 0, Bytes 0
  Out Discarded Packets 0
```

show ip as-path-access-list

- Use to display information about AS-path access lists.
- Example

```
host1#show ip as-path-access-list
AS Path Access List 1:
    permit  .*
AS Path Access List 2:
    deny  .*
AS Path Access List 3:
    permit  _109_
    deny  .*
AS Path Access List 4:
    permit  _109$
    deny  .*
AS Path Access List 10:
    deny  _109$
    permit  ^108_
    deny  .*
```

show ip community-list

- Use to display routes that are permitted by a BGP community list.
- Example

```
host1#show ip community-list
Community List 1:
  permit 752877569 (11488:1)
  permit 752877570 (11488:2)
  permit 752877571 (11488:3)
  permit 752877572 (11488:4)
Community List 2:
  permit 4294967043 (local-as)
```

show ip forwarding-table slot

- Use to display details on the forwarding table for a specific line module, including the memory used by each virtual router configured on the line module and free memory available on the module.
- The Load Errors field records any failed routing table distribution attempt as an error. Attempts can fail for many reasons during normal operation; a failed attempt does not necessarily indicate a problem. It is normal to see many Load Errors per day.
- If the Status field does not indicate Valid, then the routing table distribution has failed constantly for that VR. It is normal and appropriate behavior for the Status field to indicate Valid while the Load Error field increases daily.
- Field descriptions
  - Free Memory—Amount of routing table memory free on the line module, in kilobytes
  - Virtual Router—Name of the virtual routers configured on the line module
  - Memory (KB)—Amount of routing table memory consumed by the virtual router, in kilobytes
  - Load Errors—Count of errors made while loading the routing table on the line module
  - Status—Whether the routing table for the virtual router is valid
- Example

```
host1#show ip forwarding-table slot 9
Free Memory = 3,166KB
  Virtual Router      Memory       Load Errors     Status
                      (KB)
----------------     ---------    -------------    --------
   vr1                 4128             0          Valid
   vr2                 3136             0          Valid
   vr3                 2256             0          Valid
default               1024              0         Valid
------------------------------------------------------------
```

show ip interface

- Use to display the current state of all IP interfaces or the IP interfaces you specify.
- The default is all interface types and all interfaces.
- The **show-virtual-router-keyword** displays virtual router information.
- Field descriptions
  - interface—Interface type and interface specifier
  - interface status—Status of the interface
  - line protocol—Status of the line protocol
  - Description—Text description or alias if configured for the interface
  - Link up/down trap—Status of SNMP link up/down traps on the interface
  - Internet address—IP address of the interface
  - IP Statistics Rcvd:
    - local destination—Frames with this router as their destinations
    - hdr errors—Number of packets containing header errors
    - addr errors—Number of packets containing addressing errors
    - unkn proto—Number of packets received containing unknown protocols
    - discards—Number of discarded packets
  - IP Statistics Frags:
    - reasm ok—Number of reassembled packets
    - reasm req—Number of requests for reassembly
    - reasm fails—Number of reassembly failures
    - frag ok—Number of packets fragmented successfully
    - frag req—Number of frames requiring fragmentation
    - frag fails—Number of packets unsuccessfully fragmented
  - IP Statistics Sent:
    - generated—Number of packets generated
    - no routes—Number of packets that could not be routed
    - discards—Number of packets that could not be routed that were discarded
  - ICMP Statistics Rcvd:
    - errors—Error packets received
    - dst unreach—Packets received with destination unreachable
    - time exceed—Packets received with time-to-live exceeded
    - param probs—Packets received with parameter errors
    - src quench—Source quench packets received
    - redirect—Receive packet redirects

- echo req—Echo request (ping) packets
- echo rpy—Echo replies received
- timestamp req—Requests for a timestamp
- timestamp rpy—Replies of timestamp requests
- addr mask req—Mask requests sent
- addr mask rpy—Mask replies sent

- ICMP Statistics Sent:
  - errors—Error packets sent
  - dst unreach—Packets sent with destination unreachable
  - time excd—Packets sent with time-to-live exceeded
  - param probs—Packets sent with parameter errors
  - src quench—Source quench packets sent
  - redirect—Send packet redirects
  - timestamp req—Requests for a timestamp
  - timestamp rpy—Replies to timestamp requests
  - addr mask req—Address mask requests
  - addr mask rpy—Address mask replies
- In Received Packets, Bytes—Total number of packets and bytes received on the IP interface
  - Unicast Packets, Bytes—Unicast packets and bytes received on the IP interface; link-local received multicast packets (non-multicast-routed frames) are counted as unicast packets
  - Multicast Packets, Bytes—Multicast packets and bytes received on the IP interface which are then multicast-routed are counted as multicast packets
- In Forwarded Packets, Bytes—Packets and bytes forwarded into an output IP interface
- In Total Dropped Packets, Bytes—Total number of packets and bytes that were dropped on the interface; sum of all the drop reasons indented below this field
  - In Policed Packets—Packets discarded on a receive IP interface because of token bucket limiting, a drop action in a policy, or discarded MAC validation packets
  - In Invalid Source Address Packets—Packets discarded on a receive IP interface due to invalid IP source address (sa-validate enabled)
  - In Error Packets—Packets discarded on a receive IP interface due to IP header errors
  - In Discarded Packets—Packets discarded on the ingress interface due to a configuration problem rather than a problem with the packet itself
  - In Fabric Dropped Packets—Packets discarded on a receive IP interface due to internal fabric congestion

- Out Forwarded Packets, Bytes—Total number of packets and bytes forwarded out the IP interface
  - Unicast Packets, Bytes—Unicast packets and bytes forwarded out the IP interface
  - Multicast Routed Packets, Bytes—Multicast packets and bytes forwarded out the IP interface
- Out Requested Packets, Bytes—Packets and bytes requested to be forwarded out an IP interface
- Out Total Dropped Packets, Bytes—Total number of packets and bytes that were discarded on the egress interface; sum of all the drop reasons indented below this field
  - Out Scheduler Drops Committed Packets, Bytes—Packets and bytes dropped by the scheduler even though they had a committed traffic contract
  - Out Scheduler Drops Conformed Packets, Bytes—Packets and bytes dropped by the scheduler even though they conformed to the traffic contract
  - Out Scheduler Drops Exceeded Packets, Bytes—Packets and bytes dropped by the scheduler because they exceeded the contract
  - Out Policed Packets—Packets discarded on the egress interface due to rate limiting
  - Out Discarded Packets—Packets discarded on the egress interface due to a configuration problem rather than a problem with the packet itself
  - Out Fabric Dropped Packets—Packets dropped due to internal fabric congestion

- Example

```
host1#show ip interface detail fastEthernet 0/0
fastEthernet0/0 is up, line protocol is up
  Description: boston00 fast ethernet interface
  Link up/down trap is disabled

  Internet address is 1.1.1.2/255.255.255.0
IP statistics:
  Rcvd:  0 local destination
         0 hdr errors, 0 addr errors
         0 unkn proto, 0 discards
  Frags: 0 reasm ok, 0 reasm req, 0 reasm fails
         0 frag ok, 0 frag creates, 0 frag fails
  Sent:  31656835 generated, 0 no routes, 0 discards
ICMP statistics:
  Rcvd:  0 errors, 0 dst unreach, 0 time exceed
         0 param probs, 0 src quench, 0 redirect,
         0 echo req, 31656816 echo rpy
         0 timestmp req, 0 timestmp rpy
         0 addr mask req, 0 addr mask rpy
  Sent:  0 errors, 0 dst unreach, 0 time excd
         0 param probs, 0 src qnch, 0 redirect
         0 timestamp req, 0 timestamp rpy
         0 addr mask req, 0 addr mask rpy
In Received Packets 246220, Bytes 344624800
    Unicast Packets 246162, Bytes 344621410
    Multicast Packets 58, Bytes 3390
```

```
In Forwarded Packets 245464, Bytes 343566400
In Total Dropped Packets 756, Bytes 1058400
    In Policed Packets 756
    In Invalid Source Address Packets 0
    In Error Packets 0
    In Discarded Packets 0
    In Fabric Dropped Packets 0

  Out Forwarded Packets 117, Bytes 87297
    Unicast Packets 117, Bytes 87297
    Multicast Routed Packets 0, Bytes 0
  Out Requested Packets 117, Bytes 87297
  Out Total Dropped Packets 0, Bytes 0
    Out Scheduler Drops Committed Packets 0, Bytes 0
    Out Scheduler Drops Conformed Packets 0, Bytes 0
    Out Scheduler Drops Exceeded  Packets 0, Bytes 0
    Out Policed Packets 0
    Out Discarded Packets 0
    Out Fabric Dropped Packets 0
```

If you are losing packets because of fabric congestion, you can use the In Fabric Dropped Packets and Out Fabric Dropped Packets statistics to help determine the location of the bottleneck. Both statistics count the same thing—the same packets dropped because of fabric congestion—but in different directions.

At any given time, the total number of packets dropped in the fabric for all interfaces in the chassis is equal to the sum of all In Fabric Dropped Packets for all interfaces in the chassis, which equals the sum of all Out Fabric Dropped Packets for all interfaces in the chassis.

Packets not dropped for another listed reason are considered to have been dropped in the fabric. The router calculates In Fabric Dropped Packets by subtracting the total number of inbound packets dropped for all other reasons from the In Total Dropped Packets number. The router calculates Out Fabric Dropped Packets by subtracting the total number of outbound packets dropped for all other reasons from the Out Total Dropped Packets number.

The router calculates In Total Dropped Packets by subtracting In Forwarded Packets from In Received Packets. The router calculates Out Total Dropped Packets by subtracting Out Forwarded Packets from Out Received Packets. These statistics are reported while traffic is moving through the router. The router can get false statistics based on packets being forwarded or received after polling and based on which of the statistics is reported first. For example, In Forwarded Packets can be reported as greater than In Received Packets. Rather than displaying In Total Dropped Packets as a negative value, the command displays it as the sum of all drop reasons other than fabric drops; fabric drops are reported as 0, but might actually be nonzero. If you halt traffic, the In Total Dropped Packets and Out Total Dropped Packets values are always correct.

show ip interface shares

- Use to display information about shared IP interfaces.
- If you specify an IP interface specifier, the command displays information only for that interface and any shared IP interfaces associated with it.
- Field descriptions
  - Interface—Interface specifier or name of the interface
  - IP-Address—IP address associated with the interface
  - Status—Operational state of the interface
  - Protocol—State of the protocol running on the interface
  - Virtual Router—Virtual router in which the interface is configured
- Example 1

```
host1#show ip interface shares brief

Interface        IP-Address         Status    Protocol   Virtual Router
null0            255.255.255.255/32  up         up
fastEthernet0/0  10.13.5.17/24       up         up
loopback100      202.1.1.1/24        up         up
atm4/0.1         10.1.1.1/24         up         up
ip si0           Unnumbered          up         up        vr-a
ip si1           Unnumbered          up         up        vr-b:vrf-1
```

- Example 2

```
host1#show ip interface shares brief atm 4/0.1

Interface        IP-Address         Status    Protocol   Virtual Router
atm4/0.1         10.1.1.1/24         up         up
ip si0           Unnumbered          up         up        vr-a
ip si1           Unnumbered          up         up        vr-b:vrf-1
```

- Example 3—For a description of the following fields, see the **show ip address** command

```
host1#show ip interface shares atm 4/0.1
atm4/0.1 is up, line protocol is up
  Network Protocols: IP
  Unnumbered Interface on loopback100
  ( IP address  202.1.1.1 )
  Operational MTU = 1500  Administrative MTU = 0
  Operational speed = 155520000  Administrative speed = 0
  Discontinuity Time = 0
  Router advertisement = disabled
  Administrative debounce-time = disabled
  Operational debounce-time    = disabled
  Access routing = disabled
  Multipath mode = hashed

  In Received Packets 120, Bytes 12000
    Unicast Packets 60, Bytes 6000
    Multicast Packets 60, Bytes 6000
  In Policed Packets 0, Bytes 0
  In Error Packets 0
  In Invalid Source Address Packets 0
  Out Forwarded Packets 101, Bytes 5252
    Unicast Packets 101, Bytes 5252
    Multicast Routed Packets 0, Bytes 0
```

```
  Out Scheduler Drops Committed Packets 0, Bytes 0
  Out Scheduler Drops Conformed Packets 0, Bytes 0
  Out Scheduler Drops Exceeded  Packets 0, Bytes 0
  Out Policed Packets 0, Bytes 0

ip si0 is up, line protocol is up
  Network Protocols: IP
  Virtual Router vr-a
  Layer 2 interface atm4/0.1
  Unnumbered Interface on loopback100
  ( IP address  202.1.1.1 )
  Operational MTU = 1500  Administrative MTU = 0
  Operational speed = 155520000  Administrative speed = 0
  Discontinuity Time = 0
  Router advertisement = disabled
  Administrative debounce-time = disabled
  Operational debounce-time    = disabled
  Access routing = disabled
  Multipath mode = hashed

  In Received Packets 0, Bytes 0
    Unicast Packets 0, Bytes 0
    Multicast Packets 0, Bytes 0
  In Policed Packets 0, Bytes 0
  In Error Packets 0
  In Invalid Source Address Packets 0
  Out Forwarded Packets 101, Bytes 5252
    Unicast Packets 101, Bytes 5252
    Multicast Routed Packets 0, Bytes 0
  Out Scheduler Drops Committed Packets 0, Bytes 0
  Out Scheduler Drops Conformed Packets 0, Bytes 0
  Out Scheduler Drops Exceeded  Packets 0, Bytes 0
  Out Policed Packets 0, Bytes 0

ip si1 is up, line protocol is up
  Network Protocols: IP
  Virtual Router vr-b:vrf-1
  Layer 2 interface atm4/0.1
.
.
.
  Out Policed Packets 0, Bytes 0
```

- Example 4

```
host1#show ip interface shares ip si0
ip0 is up, line protocol is up
  Network Protocols: IP
  Layer 2 interface atm4/0.1
  Unnumbered Interface on loopback100
  ( IP address  202.1.1.1 )
  Operational MTU = 1500  Administrative MTU = 0
  Operational speed = 155520000  Administrative speed = 0
  Discontinuity Time = 0
  Router advertisement = disabled
  Administrative debounce-time = disabled
  Operational debounce-time    = disabled
  Access routing = disabled
  Multipath mode = hashed

  In Received Packets 0, Bytes 0
    Unicast Packets 0, Bytes 0
    Multicast Packets 0, Bytes 0
  In Policed Packets 0, Bytes 0
  In Error Packets 0
  In Invalid Source Address Packets 0
  Out Forwarded Packets 101, Bytes 5252
    Unicast Packets 101, Bytes 5252
    Multicast Routed Packets 0, Bytes 0
  Out Scheduler Drops Committed Packets 0, Bytes 0
  Out Scheduler Drops Conformed Packets 0, Bytes 0
  Out Scheduler Drops Exceeded  Packets 0, Bytes 0
  Out Policed Packets 0, Bytes 0
```

show ip profile

- Use to display information about a specific IP profile.
- Field descriptions
  - IP profile—Profile name
  - IP address—IP address and subnet mask of the interface or none if the interface is unnumbered
  - Unnumbered interface—Specifier for the unnumbered interface or none if the interface is numbered
  - Router—Router name
  - Directed Broadcast—Enabled or disabled
  - ICMP Redirects—Enabled or disabled
  - Access Route Addition—Enabled or disabled
  - Network Address Translation—Enable or disable; domain location (inside or outside)
  - Source-Address Validation—Enabled or disabled
  - Ignore DF Bit—Enabled or disabled
  - Administrative MTU—MTU size

- Auto Detect—Router automatically detects packets that do not match any entries in the demultiplexer table; enabled or disabled
- Auto Configure—Dynamic creation of subscriber interfaces on a primary IP interface; enabled or disabled
- IP FlowStats—Enabled or disabled

- Example

```
host1#show ip profile foo
IP profile : foo
IP address                 : none
Unnumbered interface       : none
Router                     :
Directed Broadcast         : Enabled
ICMP Redirects             : Disabled
Access Route Addition      : Enabled
Network Address Translation: Enabled, domain inside
Source-Address Validation  : Enabled
Ignore DF Bit              : Disabled
Administrative MTU         : 0
Auto Detect                : Disabled
Auto Configure             : Disabled
Auto Detect                : Disabled
IP FlowStats               : Enabled
```

show ip protocols

- Use to display configured protocols.
- Field descriptions
  - For BGP*:*
    - Redistributing—Protocol to which BGP is redistributing routes
    - Default local preference—Local preference value
    - IGP synchronization—Status of IGP synchronization: enabled, disabled
    - Always compare MED—Status of multiexit discrimination: enabled, disabled
    - Router flap damping—Status of route dampening: enabled, disabled
    - Administrative Distance—External, internal, and local administrative distances
    - Neighbor Address—IP address of the BGP neighbor
    - Neighbor Incoming/Outgoing update distribute list—Number of the access list for outgoing routes
    - Neighbor Incoming/Outgoing update prefix list—Number of the prefix list for incoming or outgoing routes
    - Neighbor Incoming/Outgoing update prefix tree—Number of the prefix tree for incoming or outgoing routes
    - Neighbor Incoming/Outgoing update filter list—Number of filter list for incoming routes
    - Routing for Networks—Network for which BGP is currently injecting routes

- For IS-IS:
  - System Id—6-byte value of the system
  - IS-Type—Routing type of the router: Level 1, Level 2
  - Distance—Administrative distance for IS-IS learned routes
  - Address Summarization—Aggregate addresses defined in the routing table for multiple groups of addresses at a given level or routes learned from other routing protocols
  - Routing for Networks—Network for which IS-IS is currently injecting routes
- For OSPF:
  - Router ID—OSPF process ID for the router
  - Distance—Administrative distance for OSPF learned routes
  - Redistributing—Protocol to which OSPF is redistributing routes
  - Address Summarization—Aggregate addresses defined in the routing table for multiple groups of addresses at a given level or routes learned from other routing protocols
  - Routing for Networks—Network for which OSPF is currently injecting routes
- For RIP:
  - Router Administrative State—RIP protocol state. Enable means that the interface is allowed to send and receive updates. Disable means that the interface, if it is configured, is not enabled to run yet.
  - System version—RIP versions allowed for sending and receiving RIP updates. The router version is currently set to RIP1, which sends RIP version 1 but will receive version 1 or 2. If the version is set to RIP2, the router will send and receive version 2 only. The default is configured for RIP1.
  - Update interval—Current setting of the update timer (in seconds)
  - Invalid after—Current setting of the invalid timer (in seconds)
  - hold down time—Current setting of the hold down timer (in seconds)
  - flushed interval—Current setting of the flush timer (in seconds)
  - Filter applied to outgoing route update—Access list applied to outgoing RIP route updates
  - Filter applied to incoming route update—Access list applied to incoming RIP route updates
  - Global route map—Route map that specifies all RIP interfaces on the router
  - Distance—Value added to RIP routes added to the IP routing table; the default is 120.

- Interface—Interface type on which RIP protocol is running
- Redistributing—Protocol to which RIP is redistributing routes
- Routing for Networks—Network for which RIP is currently injecting routes

- Example

```
host1#show ip protocols
Routing Protocol is “bgp 100”
  Redistributing: ospf
  Default local preference is 100
  IGP synchronization is enabled
  Always compare MED is disabled
  Router flap damping is disabled
  Administrative Distance: external 20 internal 200 local 200
  Neighbor(s):
    Address 1.1.1.1
    Outgoing update distribute list is 2
    Outgoing update prefix list is efg
    Incoming update prefix tree is abc
    Incoming update filter list is 1
  Routing for Networks:
    192.168.1.0/24

Routing Protocol is “isis isisOne”
  System Id: 0000.0000.0011.00  IS-Type: level-1-2
  Distance: 115
  Address Summarization:
    None
  Routing for Networks:
    fastEthernet0/0

Routing Protocol is “ospf 1” with Router ID 192.168.1.151
  Distance is 110
  Redistributing: isis
  Address Summarization:
    None
  Routing for Networks:
    192.168.1.0/255.255.255.0 area 0.0.0.0

Routing Protocol is “rip”
  Router Administrative State: enable
  System version RIP1: send = 1, receive = 1 or 2
  Update interval: 30 seconds
  Invalid after: 180  seconds
  hold down time: 120 seconds
  flushed interval: 300 seconds
  Filter applied to outgoing route update is not set
  Filter applied to incoming route update is not set
  No global route map
  Distance is 120
  Interface         Tx   Rx   Auth
  fastEthernet0/0   1    1,2  none
  Redistributing: ospf
  Routing for Networks:
    192.168.1.0/255.255.255.0
```

show ip redistribute

- Use to display configured route redistribution policy.
- Field descriptions
  - To—Protocol that routes are distributed into
  - From—Protocol that routes are distributed from
  - status—Redistribution status
  - route map number—Number of the route map
- Example

```
host1#show ip redistribute
To ospf, From static is enabled with route map 4
To ospf, From connected is enabled with route map 3
```

show ip route

- Use to display the current state of the routing table, including routes not used for forwarding.
- You can display all routes, a specific route, best route to a resolved domain name, all routes beginning with a specified address, routes for a particular protocol (BGP, IS-IS, OSPF, or RIP), locally connected routes, internal control routes, static routes, or summary counters for the routing table.
- Field descriptions
  - Protocol/Route type codes—Protocol and route type codes for the table that follows
  - Prefix—IP address prefix of network destination
  - Length—Network mask length for prefix
  - Next Hop—IP address of the next hop to the route, whether it is a local interface or another router
  - Dist—Administrative distance for the route; see Table 6
  - Met—Number of hops
  - Intf—Interface type and interface specifier
- Example 1

```
host1#show ip route
Protocol/Route type codes:
  I1- ISIS level 1, I2- ISIS level2,
  I- route type intra, IA- route type inter, E- route type external,
  i- metric type internal, e- metric type external,
  O- OSPF, E1- external type 1, E2- external type2,
  N1- NSSA external type1, N2- NSSA external type2
  L- MPLS label, V- VR/VRF, *- indirect next-hop

  Prefix/Length    Type    Next Hop       Dist/Met   Intf
  -------------    ----    --------       --------   ------
  172.16.2.0/24     Bgp     192.168.1.102   20/1       fastEthernet0/0
  10.10.0.112/32    Static  192.168.1.1     1/1        fastEthernet0/0
  10.1.1.0/24       Connect 10.1.1.1        0/1        atm3/0.100
```

- Example 2

```
host1#show ip route static
Protocol/Route type codes:
  I1- ISIS level 1, I2- ISIS level2,
  I- route type intra, IA- route type inter, E- route type external,
  i- metric type internal, e- metric type external,
  O- OSPF, E1- external type 1, E2- external type2,
  N1- NSSA external type1, N2- NSSA external type2
  L- MPLS label, V- VR/VRF, *- indirect next-hop

 Prefix/Length  Type   Next Hop   Dist/Met  Intf
 -------------  ----   --------   --------  --------------
10.10.0.112/32  Static 192.168.1.1  1/1     fastEthernet0/0
```

- Example 3

```
host1#show ip route summary
Unicast routes:
8 total routes, 576 bytes in route entries
0 isis routes
0 rip routes
3 static routes
2 connected routes
1 bgp routes
0 ospf routes
2 other internal routes
0 access routes
0 internally created access host routes

Last route added/deleted: 2::4/128 by BGP
At MON FEB 04 2008 14:18:25 UTC

Unicast routes used only for Multicast RPF check:
0 total routes, 0 bytes in route entries
0 isis routes
0 rip routes
0 static routes
0 connected routes
0 bgp routes
0 ospf routes
0 other internal routes
0 access routes
0 internally created access host routes
0 mbgp routes
0 dvmrp routes

Last route added/deleted: null by Invalid
At MON FEB 04 2008 14:18:04 UTC

MPLS tunnel routes (not used for forwarding):
3 total routes, 216 bytes in route entries
1 bgp tunnel routes
1 ldp tunnel routes
1 rsvp tunnel routes

Last route added/deleted: 2::4/128 by BGP Tunnel
At MON FEB 04 2008 14:18:26 UTC
```

- Example 4

```
host1#show ip route all
Protocol/Route type codes:
  I1- ISIS level 1, I2- ISIS level2,
  I- route type intra, IA- route type inter, E- route type external,
  i- metric type internal, e- metric type external,
  O- OSPF, E1- external type 1, E2- external type2,
  N1- NSSA external type1, N2- NSSA external type2
  L- MPLS label, V- VR/VRF, *- indirect next-hop

  Prefix/Length      Type     Next Hop          Dist/Met    Intf
  -------------      ----     --------          --------    ------
  0.0.0.0/0        Static  192.168.1.1      1/1        fastEthernet0/0
1.1.1.1/32       I2-E-i  192.168.1.105    115/10     fastEthernet0/0
6.6.6.0/24       Static  192.168.1.1      1/1        fastEthernet0/0
6.33.5.0/24      Static  0.0.0.0          1/1        loopback2
8.8.8.0/24       I2-E-i  192.168.1.105    115/10     fastEthernet0/0
9.9.9.9/32       I2-E-i  192.168.1.105    115/10     fastEthernet0/0
10.0.0.0/8       I2-E-i  192.168.1.105    115/10     fastEthernet0/0
10.10.0.156/32   Static  192.168.1.1      1/1        fastEthernet0/0
11.1.1.1/32      I2-E-i  192.168.1.105    115/10     fastEthernet0/0
11.11.11.12/32   I2-I-i  192.168.1.105    115/10     fastEthernet0/0
22.2.0.0/16      I2-I-i  92.168.1.105     115/10     fastEthernet0/0
34.0.0.0/8       I2-E-i  192.168.1.105    115/10     fastEthernet0/0
172.20.32.0/24   Static  192.168.1.1      1/1        fastEthernet0/0
174.20.32.0/24   I2-I-i  192.168.1.105    115/20     fastEthernet0/0
176.20.32.0/24   Connect 176.20.32.1      0/1        loopback1
192.168.1.0/24   Connect 192.168.1.214    0/1        fastEthernet0/0
201.1.1.0/24     I2-E-i  192.168.1.105    115/10     fastEthernet0/0
201.2.1.0/24     I2-E-i  192.168.1.105    115/10     fastEthernet0/0
201.3.1.0/24     I2-E-i  192.168.1.105    115/10     fastEthernet0/0
202.1.1.1/32     I2-E-i  192.168.1.105    115/10     fastEthernet0/0
207.1.1.0/24     I2-E-i  192.168.1.105    115/10     fastEthernet0/0
```

- Example 5—Indirect Next Hop (* displayed)

```
host1#show ip route
Protocol/Route type codes:
  I1- ISIS level 1, I2- ISIS level2,
  I- route type intra, IA- route type inter, E- route type external,
  i- metric type internal, e- metric type external,
  O- OSPF, E1- external type 1, E2- external type2,
  N1- NSSA external type1, N2- NSSA external type2
  L- MPLS label, V- VR/VRF, *- indirect next-hop

  Prefix/Length     Type      Next Hop      Dst/Met             Intf
------------------ ------- --------------- ---------- ------------------------
21.21.21.2/32      Static  0.0.0.0         1/0        loopback0[V:pe2]
2.2.2.2/32         O-I     30.30.30.2      110/3      ATM2/0.30
                           31.31.31.2      110/3      ATM2/0.31
10.10.10.0/24      Connect 10.10.10.1      0/0        ATM2/0.10
20.20.20.0/24      Connect 20.20.20.1      0/0        ATM2/0.21
4.4.4.4/32         Bgp     2.2.2.2*        200/2
                           3.3.3.3*        200/2
5.5.5.5/32         Bgp     4.4.4.4*        20/2
```

- Example 6—Indirect Next Hop with detail

```
host1#show ip route 4.4.4.4 detail
Protocol/Route type codes:
  I1- ISIS level 1, I2- ISIS level2,
  I- route type intra, IA- route type inter, E- route type external,
  i- metric type internal, e- metric type external,
  O- OSPF, E1- external type 1, E2- external type2,
  N1- NSSA external type1, N2- NSSA external type2
  L- MPLS label, V- VRF

4.4.4.4/32   Type: Bgp   Distance: 200   Metric: 0   Tag: 0
   Indirect NHop: virtual-router: pe1
   Address 1.1.1.1 Type Bgp Index 1
      NHop: 10.10.10.2 IfIndx: 28 Intf: ATM2/0.10
      NHop: 20.20.20.2 IfIndx: 28 Intf: ATM2/0.20

   Indirect NHop: virtual-router: pe1
   Address 2.2.2.2 Type Bgp Index 2
      NHop: 10.10.10.2 IfIndx: 28 Intf: ATM2/0.10
      NHop: 20.20.20.2 IfIndx: 28 Intf: ATM2/0.20
```

show ip route slot

- Use to display the interface and next hop for an IP address in the routing table of a line module.
- A next hop is displayed only for protocols where ARP is used to resolve the addresses, such as for fastEthernet, gigabitEthernet, bridged Ethernet over ATM, and so on.
- Field descriptions
  - IP address—Address reachable via the interface
  - Interface—Interface type and specifier associated with the IP address; displays “Local Interface” if a special interface index is present in the routing table for special IP addresses, such as broadcast addresses
  - Next Hop—IP address of the next hop router to reach the IP address; displays “*---*” if no next hop is associated with the IP address; displays “Down” if the ECMP set for a specific route on a slot is down
- Example 1

```
host1#show ip route slot 6 10.10.0.231
IP address     Interface            Next Hop
------------   ----------------     ------------
10.10.0.231    fastEthernet 6/0     10.10.0.231
```

- Example 2

```
host1#show ip route slot 9 90.248.1.2
IP address     Interface            Next Hop
------------   ----------------     ------------
90.248.1.2     serial9/23:2         ---
```

- Example 3

```
host1#show ip route slot 9 90.249.255.255
IP address     Interface            Next Hop
------------   ----------------     ------------
90.249.255.255 Local Interface      ---
```

show ip socket statistics

- Use to display basic information about BSD sockets that have been instantiated in the VR in whose context you issue the command. The information includes the connection information (source and destination IP address and port numbers), socket type, the options in effect on the socket, and the socket's state.
- Use the **detailed** keyword to display blocks of extensive information about every socket, such as how many times various APIs have been called and the socket event log. The **detailed** keyword displays information about only the sockets that are associated with the VR in whose context you issue the command or sockets that are not associated with any VR.
- Baselining is not supported for this command.
- Field descriptions
  - *socketNumber ipAddress*:*portNumber* -- > *ipAddress*:*portNumber*—Socket and the IP address and port number for each end of the connection, with the E-series router shown on the left and the remote peer on the right
  - type—Type of connection: SOCK_STREAM (uses TCP) or DGRAM (datagram; uses UDP)
  - opts—Options set on the individual sockets
    - SO_DEBUG—Turn on debugging; has no effect
    - SO_ACCEPTCONN—Socket can accept incoming connections
    - SO_REUSEADDR—Allow reuse of the local address
    - SO_KEEPALIVE—Do keepalives on the connection
    - SO_DONTROUTE—Do not route packets, use interface addresses
    - SO_BROADCAST—Broadcasts can be sent over the socket
    - SO_USELOOPBACK—Bypass the hardware if/when possible
    - SO_LINGER—Linger on a close() if data is present
    - SO_OOBINLINE—Leave received out-of-band data in-line
    - SO_REUSEPORT—Allow reuse of local port
  - so_state—State of each socket; knowledge of BSD Sockets API is useful to understand this information
    - SS_NOFDREF—No file table reference any more
    - SS_ISCONNECTED—Socket is connected to a peer
    - SS_ISCONNECTING—Socket is in process of connecting to peer
    - SS_ISDISCONNECTING—Socket is in process of disconnecting
    - SS_CANTSENDMORE—Socket cannot send more data to peer
    - SS_CANTRCVMORE—Socket cannot receive more data from peer
    - SS_RCVATMARK—Socket at mark on input
    - SS_PRIV—Socket is privileged for broadcast, raw

  - SS_NBIO—Socket allows nonblocking operations
  - SS_ASYNC—Socket allows asynchronized I/O notifications
  - SS_ISCONFIRMING—Socket is deciding to accept connection request
- pending xmit byte count = 0 recv count—Number of bytes that are pending to be sent (queued up) and received
- Keep alive idle time—Number of seconds before TCP sends an initial keepalive probe to an idle remote node
- keep alive poll time—Interval in seconds at which TCP sends keepalive probes to idle remote nodes
- Additional state flags—State of the following flags in the socket_stats structure: ss_Bound, ss_BindError, ss_ListenOk, ss_ListenError, ss_AcceptOk, ss_AcceptError, ss_RsAcceptOk, ss_RsAcceptError, ss_ConnectOk, ss_ConnectErrors, ss_ConnectToOk, ss_ConnectToError, ss_CalledShutdown, and ss_CalledRsSocreate.
- Counters that show how often the indicated routine has been called: so_SendtoCalls, so_SendMsgCalls, so_SendCalls, so_SockWriteCalls, so_SendErrors, so_SentBytes, so_BsdCloseNotClosed, so_RecvBytes, so_RecvErrors, so_RecvFroms, so_Recvs, so_RecvMsgs, so_Reads
- Socket Event Log (most recent at bottom)—Event log on this socket. Each one shows a call to a particular function within the socket library. Includes a repetition counter that displays only nonzero values.
  - Call to sofree()—Call included because in some circumstances an sofree() call does not result in the socket being destroyed (and memory being returned to the free pool)
  - Call to rsSocket()—Call to create the socket using rsSocket() as opposed to socket()
  - Call to socket()—8-bit value indicating how the call went
  - Call to connect()—8-bit value indicating how the call went
  - Call to listen()—8-bit value indicating how the call went
  - Call to accept()—8-bit value indicating how the call went
  - Call to bind()—8-bit value indicating how the call went
  - Call to connectto()—8-bit value indicating how the call went
  - Call to rsAccept()—8-bit value indicating how the call went
  - Call to sobind()—8-bit value indicating how the call went
  - Call to solisten()—8-bit value indicating how the call went
  - Call to soclose()—8-bit value indicating how the call went
  - Call to soabort()—8-bit value indicating how the call went
  - Call to soaccept()—8-bit value indicating how the call went
  - Call to soconnect()—8-bit value indicating how the call went
  - Call to soconnect2()—8-bit value indicating how the call went
  - Call to sodisconnect()—8-bit value indicating how the call went
  - Call to soshutdown()—8-bit value indicating how the call went

- Call to sowakeup()—8-bit value indicating what kind of wakeup it is. 1 (SELREAD) indicates that data is available on the socket for the application. 2 (SELWRITE) means that more buffer space is available and the application can queue up more data to be transmitted.
- Call to soclose()—8-bit value indicating how the call went
- Call to sendto()—16-bit value indicating the return status
- Call to write()—16-bit value indicating the return status
- Call to sendmsg()—16-bit value indicating the return status
- Call to send()—16-bit value indicating the return status
- Call to recvfrom()—16-bit value indicating the return status
- Call to recv()—16-bit value indicating the return status
- Call to recvmsg()—16-bit value indicating the return status
- Call to read()—16-bit value indicating the return status

- Example 1

```
host1#show ip socket statistics
   5 10.13.5.70:23 --> 10.10.132.71:2000
    type: 1 (SOCK_STREAM)
    opts = 13 SO_DEBUG SO_REUSEADDR SO_KEEPALIVE
    so_state = 177 SS_NOFDREF SS_CANTSENDMORE SS_CANTRCVMORE SS_PRIV

  18 0.0.0.0:23 --> 0.0.0.0:0
    type: 1 (SOCK_STREAM)
    opts = 7 SO_DEBUG SO_ACCEPTCONN SO_REUSEADDR
    so_state = 128 SS_PRIV
```

- Example 2—Additional fields displayed by **detailed** keyword

```
host1#show ip socket statistics detailed
18 0.0.0.0:23 --> 0.0.0.0:0
    type: 1 (SOCK_STREAM)
    opts = 7 SO_DEBUG SO_ACCEPTCONN SO_REUSEADDR
    so_state = 128 SS_PRIV
  pending xmit byte count = 0  recv count 0
  Keep alive idle time = 14400  keep alive poll time = 150
  Additional state flags:
      so_Bound
      so_ListenOk
      ss_CalledRsSocreate

        so_SendtoCalls = 0
       so_SendMsgCalls = 0
          so_SendCalls = 0
     so_SockWriteCalls = 0
         so_SendErrors = 0
          so_SentBytes = 0
  so_BsdCloseNotClosed = 0
          so_RecvBytes = 0
         so_RecvErrors = 0
          so_RecvFroms = 0
              so_Recvs = 0
           so_RecvMsgs = 0
              so_Reads = 0
```

```
Socket Event Log (most recent at bottom)
  rssocket
  sobind - 0
  bind - 0
  solisten - 0
  listen - 0
```

show ip static

- Use to display the status of static routes in the routing table.
- You can specify an IP mask that filters specific routes.
- Field descriptions
  - Prefix—IP address prefix
  - Length—Prefix length
  - Next Hop—IP address of the next hop
  - Met—Number of hops
  - Dist—Administrative distance of the route; see Table 6
  - Tag—Tag value of the route
  - Intf—Interface type and interface specifier
  - Verify—Status of the RTR or BFD operation associated with the specified static route; this field is blank if the **verify** (BFD) or **verify rtr** (RTR) keywords were not specified as part of the **ip route** command. The display can include the following:
    - BFD up/down—Current status of the associated BFD operation
    - operation number—Number of the associated RTR operation
    - up/down—Current status of the associated RTR operation
    - (lr)—Indicates that although the associated RTR operation is currently down, the router will install this route in the routing table, provided that no other static route to the same network prefix is available; this field appears for an RTR operation that is down when the **last-resort** keyword is specified as part of the **ip route verify rtr** command
- Example

```
host1#show ip static
Prefix/Length      Next Hop    Met   Dist   Tag    Intf               Verify
1.1.1.2/32         1.1.1.2       0     1        0 FastEthernet4/0    2 up
1.1.1.2/32         1.1.1.2       0     1        0 FastEthernet4/1
10.10.133.17/32    10.6.128.1    1     1        0 unresolved         1 down
11.11.11.11/32     3.3.3.3       0     1        0 unresolved         1 down(lr)
```

show tcp ack-rst-and-syn

- Use to display the status of TCP ACK, RST, and SYN protection.
- Example

```
host1#show tcp ack-rst-and-syn
TCP Ack Rst and Syn Protection is ENABLED
```

show tcp resequence-buffers

- Use to display the configuration, current per-VR, and per-router state of the TCP resequencing buffer management functions.
- Use the *vrfName* variable to specify a specific VRF for which you want to view information.
- Field descriptions

  TCP Resequence Buffer Management Configuration

  - Global Maximum—Number of buffers that can be on the reordering queues of all connections in all virtual routers
  - Default Per-VR Maximum—Default maximum number of buffers for all connections in a single VR
  - Default Connection Maximum—Default maximum number of buffers for each connection in each virtual router
  - This VR Maximum—Maximum number of outstanding resequencing buffers in the current VR
  - This VR Connection Maximum—Maximum number of outstanding resequencing buffers on any one connection in this VR

  TCP Resequence Buffer Management State

  - Global buffers in use—Total number of outstanding resequencing buffers in the router
    - High Water—Largest number of outstanding resequencing buffers that the router has experienced since the last reset
  - VR Buffers in use—Number of outstanding resequencing buffers in the current virtual router
    - High Water—Largest number of outstanding resequencing buffers for the current virtual router since the last reset
  - Buffers Discarded Because Global Limit Exceeded—Number of resequencing buffers discarded because the global limit was reached
  - Buffers Discarded Because VR Limit Exceeded—Number of resequencing buffers that have been discarded in this virtual router because the virtual router buffer limit was reached
- Example

```
host1#show tcp resequence-buffers
TCP Resequence Buffer Management Configuration
  Global Maximum: ###
    Default Per-VR Maximum: 250
    Default Connection Maximum: 15
  This VR Maximum: 300
  This VR Connection Maximum: 15

TCP Resequence Buffer Management State
  Global buffers in use: 5
             High Water: 15
      VR Buffers in use: 17
             High Water: 32
  Buffers Discarded Because Global Limit Exceeded: 25
      Buffers Discarded Because VR Limit Exceeded: 15
```

show tcp path-mtu-discovery

- Use to display PMTU information.
- Field descriptions
  - TCP PMTU Discovery—State of the PMTUD functions (ENABLED or DISABLED)
  - Administrative Minimum MTU—Administrative minimum PMTU that is supported or *none* if there is no minimum
  - Administrative Maximum MTU—Administrative maximum PMTU that is supported or none if there is no maximum
  - Timer 1—Value of timer 1 in minutes
  - Timer 2—Value of timer 2 in minutes
  - Black Hole Detect Threshold—Number of retransmissions allowed before TCP/PMTUD assumes that there is a black hole and attempts to reduce impact in the MSS
  - # ICMP TooBigs—Number of ICMP Too Big messages that have been received
  - # ICMP TooBigs for unk. connections—Number of ICMP Too Big messages that have been received which were not for a valid connection
- Example

```
host1#show tcp path-mtu-discovery
TCP PMTU Discovery is ENABLED
    Administrative Minimum MTU: 512
    Administrative Maximum MTU: 65535
    Timer 1: 10 minutes
    Timer 2: 2 minutes
Black Hole Detect Threshold: 0 retransmissions
# ICMP TooBigs: 0
# ICMP TooBigs for unk. connections: 0
```

show tcp paws

- Use to display the TCP PAWS status.
- Example

```
host1#show tcp paws
TCP PAWS is disabled
```

show tcp statistics

- Use to display all TCP statistics.
- Baselining is supported for this command.
- Use the **ip** keyword to display only IPv4 statistics.
- Use the **ipv6** keyword to display only IPv6 statistics.
- Use the **brief** keyword to display summary information or the **detailed** keyword to display extensive information.

- Use the **diagnostic** keyword to display diagnostic information collected on the TCP statistics in addition to the detailed information. This command shows information only for the connections that are active within the context of the VR in which you issue the command.
- Field descriptions
  - TCP Global Statistics Connections:
    - attempted—Number of outgoing TCP connections attempted
    - accepted—Number of incoming TCP connections accepted
    - established—Number of TCP connections established
  - TCP Global Statistics Rcvd:
    - total pkts—Total number of packets received
    - in-sequence pkts—Number of packets received in sequence
    - bytes—Number of bytes received
    - chksum err pkts—Number of checksum error packets received
    - authentication err pkts—Number of authentication error packets received
    - bad offset pkts—Number of bad offset packets received
    - short pkts—Number of short packets received
    - duplicate pkts—Number of duplicate packets received
    - out of order pkts—Number of packets received out of order
  - TCP Global Statistics Sent:
    - total pkts—Total number of packets sent
    - data pkts—Number of data packets sent
    - bytes—Number of bytes sent
    - retransmitted pkts—Number of packets retransmitted
    - retransmitted bytes—Number of bytes retransmitted
  - Global Diagnostic Data Unknown Connection log—Includes the following global statistics:
    - Source address/port – local port—Shows the 32 most recent TCP connection attempts that were rejected, including the remote node's IP address and port, the local port for the connection attempt, and the number of identical attempts that have been received on that port in a row. The reason for rejection is not given. This information may be useful in tracking down DoS attacks.
    - # connection-reqs rejected—Total number of connection attempts that have been rejected
    - # connection-reqs pending—Current number of connection attempts that are pending, awaiting additional data from the peer
    - # sonewconn calls that fail—Number of calls to sonewconn that have failed. This statistic often indicates that either a socket connection limit has been reached or that there was no memory to hold the socket data structures.

- TCP Session Statistics
  - Local addr—Local address of the TCP connection
  - Local port—Local port number of the TCP connection
  - Remote addr—Remote address of the TCP connection
  - Remote port—Remote port number of the TCP connection
  - State—Current state of the TCP connection
  - Authentication—Authentication status of the TCP connection
- TCP Session Statistics Sent:
  - total pkts—Total number of packets sent on the TCP connection
  - data pkts—Number of data packets sent on the TCP connection
  - bytes—Number of bytes sent on the TCP connection
  - retransmitted pkts—Number of packets retransmitted on the TCP connection
  - retransmitted bytes—Number of bytes retransmitted on the TCP connection
- TCP Session Statistics Rcvd:
  - total pkts—Total number of packets received on the TCP connection
  - in-sequence pkts—Number of packets received in sequence on the TCP connection
  - bytes—Number of bytes received on the TCP connection
  - chksum err pkts—Number of checksum error packets received on the TCP connection
  - bad offset pkts—Number of bad offset packets received on the TCP connection
  - short pkts—Number of short packets received on the TCP connection
  - duplicate pkts—Number of duplicate packets received on the TCP connection
  - out of order pkts—Number of packets received out of order on the TCP connection
- Diagnostics: PRU_ Operations counters—Number of calls for each of the indicated PRU_operations within the TCP service API. These are per-connection statistics.
- Wildcard Matches—Number of packets received that matched this TCP connection due to wildcard matching. Matching is expected for listening server connections, such as Telnet, but is not expected for established connections. This is a per-connection statistic.
- Rcv'd Packets after connection closed—Number of packets received on the connection after the connection has been closed (and before the data structure gets removed). This is a per-connection statistic.
- Connect request rejected—Number of times an incoming connection request was not approved. This is a per-connection statistic.

- Connect request approval pending—Number of times that an incoming connection request was held pending, waiting for a subsequent packet. This is a per-connection statistic.
- New soconnect failed—Number of times a SONEWCONN() was tried on a listening connection and failed. This is a per-connection statistic.
- # Write-Wakeups—Number of times a "write wakeup" occurred on the connection. This is a per-connection statistic.
- # Read wakeups—Number of times a "read wakeup" occurred on the connection. This is a per-connection statistic.
- # receives after close—Number of packets received with data after the connection entered the close-wait state. This is a per-connection statistic.
- Retransmit timer—Current value of the retransmit timer
- Persistence timer—Current value of the persistence timer
- Keepalive timer—Current value of the keepalive timer
- 2MSL timer—Current value of the 2MSL (max segment lifetime) timer
- tcpDisconnect()s—Number of times BsdTcp::tcpDisconnect() was called. This is a per-connection statistic.
- keep T/O pre-estab—Number of times the keepalive timer expired before the connection reached the established state. This is a per-connection statistic.
- tcpkeeptimeo_idle—Number of times the keepalive timer popped, but no keepalive was sent because of connection idle-time considerations. This is a per-connection statistic.
- TCP Connection Event Log (most recent at bottom)—Event log for the TCP connection. It shows the last 32 events that occurred on the connection. The most recent event is at the bottom of the list. This is per-connection data.
  - TCPS_ELOG_PRU_ATTACH
  - TCPS_ELOG_PRU_BIND

The following events can be recorded:

Fast Timeout	Did a PRU_CONNECT
2MSL Timeout	Did a PRU_CONNECT2
Retransmit Timeout	Did a PRU_DISCONNECT
Persist Timeout	Did a PRU_ACCEPT
Received FIN packet	Did a PRU_SHUTDOWN
Received SYN packet	Did a PRU_RCVD
Received Retransmission	Did a PRU_SEND
Transmit a FIN packet	Did a PRU_ABORT
Transmit a SYN packet	Did a PRU_SENSE
Retransmit a packet	Did a PRU_RCVOOB
Did a PRU_ATTACH	Did a PRU_SENDOOB

Did a PRU_DETACH	Did a PRU_SOCKADDR
Did a PRU_BIND	Did a PRU_PEERADDR
Did a PRU_LISTEN	The keepalive timer popped. An 8-bit argument that describes how the timer was handled: ■ Ignored because the session was not established (that is, not in the OPEN state) ■ Ignored due to idle-timeout considerations ■ A packet was sent ■ Ignored because the connection did not have the keepalive option set OR the connection was in the process of closing

- RST/SYN-Ack DoS Protection—Specifies when this function is enabled
  - RSTs acked—Number of RSTs received and then acknowledged by the TCP stack.

NOTE: This count is maintained even when the protection functions are disabled. The value indicates the count of packets that would have been acknowledged if the protections were enabled. Providing this information can help determine whether attacks are occurring.

  - Bogus RSTs—Number of RSTs that were judged to be invalid (that is, their timer expired) and therefore ignored
  - SYNs acked—Number of SYNs received and then acknowledged by the TCP stack.

NOTE: This count is maintained even when the protection functions are disabled. The value indicates the count of packets that would have been acknowledged if the protections were enabled. Providing this information can help determine whether attacks are occurring.

  - Bogus SYNs—Number of RSTs that were judged to be invalid (that is, their timer expired) and therefore ignored
  - Data Insertions rejected—Number of packets received and dropped because they are believed to have been inserted by an attacker

NOTE: This count is maintained even when the protection functions are disabled. The value indicates the count of packets that would have been rejected if the protections were enabled. Providing this information can help determine whether attacks are occurring.

- PMTUD information—Information regarding path MTU discovery
  - PMTUD—Status of path MTU discovery on the virtual router: enabled or disabled
  - Administrative Minimum MTU—Minimum MTU that is enabled on any connection; a value of "none" indicates that the minimum is zero (0)
  - Administrative Maximum MTU—Maximum MTU that is enabled on any connection; a value of "none" indicates that the maximum is 65535
  - Timer 1—Amount of time the virtual router waits after receiving an ICMP Too Big message before attempting to increase the path MTU
  - Timer 2—Amount of time the virtual router waits after successfully increasing the MTU before attempting to increase it more
  - # ICMP TooBigs—Number of ICMP Too Big messages that the router has received. When PMTU is disabled, this counter does not increase.
  - # ICMP TooBigs for unk. connection—Number of ICMP Too Big messages that the router has received for TCP connections that do not exist. When PMTU is disabled, this counter does not increase.
  - PMTU Increase Attempts—Number of attempts the router has made to increase the PMTU
  - Black Hole Detect Threshold—Number of successive transmissions that must occur on a connection before that connection treats retransmissions as indications that something is wrong
  - Override MSS—MSS that is advertised to peers, overriding the MSS that is derived from the interface MTU. This line does not appear in the output if you do not set the value.
- MTU/MSS information—Information regarding path MTU/MSS
  - PMTU—Status of MTU/MSS on this virtual router: enabled or disabled
  - MSS in effect—MSS currently being used for transmission to the peer. This number changes while various network events occur to cause the router to increase or decrease its estimate of the MSS.
  - Calculated MSS to peer—MSS that path MTU discovery has calculated (if PMTUD is enabled) to the peer
  - MSS received from peer—MSS that the peer received in a TCP MSS option. If no option is received, the value is zero (0).
  - Application set MSS—MSS that an application might have set for the connection
  - Xmit Interface MSS—MSS for the interface used to transmit packets to the peer; calculated as the interface MTU minus the size of the TCP and IP headers.
  - MSS Sent to Peer—MSS that has been advertised to the peer
  - "ICMP DestUn, Frag Req'd and DF Set" messages—Number of ICMP "Destination Unreachable: Fragmentation Required and DF set" messages that the router has received

- Number of attempts to increase PMTU—Number of times the router has attempted to increase the PMTU by probing with a packet that is larger than the known MTU
- Time to next increase attempt—Amount of time, in seconds, until the router retries to increase the MTU
- Black Hole Detection State—State of the black hole detection mechanism: none, detecting, probable, or unknown

- Out-of-Order Packet Queue Information—Information regarding packet queue buffers
  - Buffers Outstanding—Number of buffers currently on the connection reordering queue
  - High Water—Most buffers that have ever been on the connection reordering queue
  - Buffers discarded—Number of buffers that were discarded because keeping them would have exceeded the connection maximum
- TCP PAWS is [enabled/disabled]—Status of the TCP PAWS option; enabled indicates that PAWS is functioning normally (default mode) for TCP segments; disabled indicates that PAWS is disabled for TCP segments

- Example 1

```
host1#show ip tcp statistics
TCP Global Statistics:
  Connections: 7358 attempted, 4 accepted, 7362 established
        0 dropped, 14718 closed
  Rcvd: 75923 total pkts, 53608 in-sequence pkts, 3120303 bytes
        0 chksum err pkts, 0 authentication err pkts, 0 bad offset pkts
        0 short pkts, 0 duplicate pkts, 0 out of order pkts
  Sent: 82352 total pkts, 44404 data pkts, 657095 bytes
        34 retransmitted pkts, 487 retransmitted bytes

TCP Session Statistics:
  Local addr: 0.0.0.0, Local port: 23
  Remote addr: 0.0.0.0, Remote port: 0
  State: LISTEN Authentication: None
  Rcvd: 4 total pkts, 0 in-sequence pkts, 0 bytes
        0 chksum err pkts, 0 bad offset pkts, 0 short pkts
        0 duplicate pkts, 0 out of order pkts
  Sent: 0 total pkts, 0 data pkts, 0 bytes
        0 retransmitted pkts, 0 retransmitted bytes

  Local addr: 192.168.1.250, Local port: 23
  Remote addr: 10.10.0.77, Remote port: 2170
  State: ESTABLISHED Authentication: None
  Rcvd: 61 total pkts, 34 in-sequence pkts, 41 bytes
        0 chksum err pkts, 0 bad offset pkts, 0 short pkts
        0 duplicate pkts, 0 out of order pkts
  Sent: 64 total pkts, 45 data

  Local addr: 192.168.1.250, Local port: 23
  Remote addr: 10.10.0.77, Remote port: 2170
  State: ESTABLISHED Authentication: None
  Rcvd: 61 total pkts, 34 in-sequence pkts, 41 bytes
        0 chksum err pkts, 0 bad offset pkts, 0 short pkts
        0 duplicate pkts, 0 out of order pkts
  Sent: 64 total pkts, 45 data pkts, 2304 bytes
        0 retransmitted pkts, 0 retransmitted bytes
```

```
 Local addr: 192.168.1.250, Local port: 23
  Remote addr: 192.168.1.139, Remote port: 1038
  State: ESTABLISHED Authentication: None
  Rcvd: 295 total pkts, 159 in-sequence pkts, 299 bytes
        0 chksum err pkts, 0 bad offset pkts, 0 short pkts
        0 duplicate pkts, 0 out of order pkts
  Sent: 281 total pkts, 210 data pkts, 3089 bytes
        0 retransmitted pkts, 0 retransmitted bytes
```

- Example 2—Additional fields displayed by **diagnostic** keyword

```
host1#show ip tcp statistics diagnostic
...
Global Diagnostic Data
    Unknown Connection log
Source address/port -> local port
         128.127.126.125/124 -> 8080  count: 3
         111.111.111.111/222 -> 3333  count: 4
# connection-reqs rejected: 0
    # connection-reqs pending: 0
    # sonewconn calls that fail: 0
...
  Diagnostics:
    PRU_ Operations counters:
      PRU_ATTACH: 0
      PRU_DETACH: 0
      PRU_BIND: 1
      PRU_LISTEN: 1
      PRU_CONNECT: 0
      PRU_ACCEPT: 0
      PRU_DISCONNECT: 0
      PRU_SHUTDOWN: 0
      PRU_RCVD: 0
      PRU_SEND: 0
      PRU_ABORT: 0
      PRU_CONTROL: 0
      PRU_SENSE: 0
      PRU_RCVOOB: 0
      PRU_SENDOOB: 0
      PRU_SOCKADDR: 0
      PRU_PEERADDR: 0
      PRU_CONNECT2: 0
      PRU_FASTTIMO: 0
      PRU_SLOWTIMO: 0
      PRU_PROTORCV: 0
      PRU_PROTOSEND: 0
    Wildcard Matches: 2
    Rcv'd Packets after connection closed: 0
    Connect request rejected: 0
    Connect request approval pending 0
    New soconnect failed 0
    # Write-Wakeups: 0
    # Read wakeups 0
    # receives after close 0
    Retransmit timer: 0
    Persistance timer: 0
    Keepalive timer: 0
    2MSL timer: 0
    tcpDisconnect()s: 0
    keep T/O pre-estab: 0
    tcpkeeptimeo_idle: 0
...
```

```
TCP Connection Event Log (most recent at bottom)
      TCPS_ELOG_PRU_ATTACH
      TCPS_ELOG_PRU_BIND
```

- Example 3—Additional fields displayed by **detailed** keyword

```
host1#show ip tcp statistics detailed
...

RST/SYN-Ack Protection is: ENABLED
    RSTs acked: 0
    ...Bogus RSTs: 0
    SYNs acked: 0
    ...Bogus SYNs: 0
    Data Insertions rejected: 0
PMTUD Information:      PMTUD: ENABLED
    Administrative Minimum MTU: 512
    Administrative Maximum MTU: none
    Timer 1: 10 minutes
    Timer 2: 2 minutes
    # ICMP TooBigs: 0
    # ICMP TooBigs for unk. connection: 0
    PMTU Increase Attempts: 17
    Black Hole Detect Threshold: 50 retransmissions
...
MTU/MSS Information
    ENABLED on this connection
    MSS in effect: 536
    Calculated MSS to peer: 536
    MSS received from peer: 0
    Application set MSS: 0
    Xmit Interface MSS: 0
    MSS Sent to Peer: 0
    "ICMP DestUn, Frag Req'd and DF Set" messages: 0
    Number of attempts to increase PMTU: 0
    Time to next increase attempt: 0 seconds
    Black Hole Detection State: none
...
Out-of-order Packet Queue Information

  Buffers Outstanding: 25
           High Water: 28
    Buffers discarded: 15
...
TCP-Paws is disabled
```

show ip traffic

- Use to display statistics about IP traffic.
- You can use the ipTraffic log to show consumable IP traffic to the SRP module; the traffic is filterable per router and IP interface. You can show ICMP, TCP, and UDP traffic with the icmpTraffic, udpTraffic, and tcpTraffic logs.
- Field descriptions
  - IP Statistics Rcvd:
    - router Id—Router ID number
    - total—Number of frames received
    - local destination—Frames with this router as their destination

- hdr errors—Number of packets containing header errors
- addr errors—Number of packets containing addressing errors
- unkn proto—Number of packets received containing unknown protocols
- discards—Number of discarded packets

- IP Statistics Frags:
  - reassembled—Number of reassembled packets
  - reasm timed out—Number of reassembled packets that timed out
  - reasm req—Number of requests for reassembly
  - reasm fails—Number of reassembly failures
  - frag ok—Number of fragmented packets reassembled successfully
  - frag fail—Number of fragmented packets reassembled unsuccessfully
  - frag creates—Number of packets created by fragmentation
- IP Statistics Sent:
  - forwarded—Number of packets forwarded
  - generated—Number of packets generated
  - out disc—Number of outbound packets discarded
  - no routes—Number of packets that could not be routed
  - routing discards—Number of packets that could not be routed and were discarded
- IP Statistics Route:
  - routes in table—Number of routes in the routing table
- ICMP Statistics Rcvd:
  - total—Total number of ICMP packets received
  - errors—Number of error packets received
  - dst unreach—Number of packets received with destination unreachable
  - time exceed—Number of packets received with time-to-live exceeded
  - param probs—Number of packets received with parameter errors
  - src quench—Number of source quench packets received
  - redirects—Number of receive packet redirects
  - echo req—Number of echo request (ping) packets
  - echo rpy—Number of echo replies received
  - timestamp req—Number of requests for a timestamp
  - timestamp rpy—Number of replies to timestamp requests
  - addr mask req—Number of mask requests received
  - addr mask rpy—Number of mask replies received

- ICMP Statistics Sent:
  - total—Total number of ICMP packets sent
  - errors—Number of error packets sent
  - dest unreach—Number of packets sent with destination unreachable
  - time excd—Number of packets sent with time-to-live exceeded
  - param prob—Number of packets sent with parameter errors
  - src quench—Number of source quench packets sent
  - redirects—Number of send packet redirects
  - echo req—Number of echo request (ping) packets
  - echo rpy—Number of echo replies sent
  - timestamp req—Number of requests for a timestamp
  - timestamp rpy—Number of replies to timestamp requests
  - addr mask req—Number of address mask requests sent
  - addr mask rpy—Number of replies to address mask requests
- UDP Statistics Rcvd:
  - total—Total number of UDP packets received
  - checksum—Number of checksum error packets received
  - no port—Number of packets received for which no E-series router application listener was listening on the destination port
- UDP Statistics Sent:
  - total—Total number of UDP packets sent
  - errors—Number of error packets sent
- TCP Global Statistics Connections:
  - attempted—Number of outgoing TCP connections attempted
  - accepted—Number of incoming TCP connections accepted
  - established—Number of TCP connections established
  - dropped—Number of TCP connections dropped
  - closed—Number of TCP connections closed
  - currently established—Number of TCP connections currently established
- TCP Global Statistics Rcvd:
  - total pkts—Total number of TCP packets received
  - in-sequence pkts—Number of packets received in sequence
  - bytes—Number of bytes received
  - chksum err pkts—Number of checksum error packets received
  - authentication err pkts—Number of authentication error packets received
  - bad offset pkts—Number of packets received with bad offsets

- short pkts—Number of short packets received
- duplicate pkts—Number of duplicate packets received
- out of order pkts—Number of packets received out of order

- TCP Global Statistics Sent:
  - total pkts—Total number of TCP packets sent
  - data pkts—Number of data packets sent
  - bytes—Number of bytes sent
  - retransmitted pkts—Number of packets retransmitted
  - retransmitted bytes—Number of retransmitted bytes
- OSPF Statistics—Provides statistics on OSPF
- IGMP Statistics—Provides statistics about queries, reports sent or received
- ARP Statistics—Not supported for this version of the router

- Example

```
host1#show ip traffic
IP statistics: Router Id: 172.31.192.217
  Rcvd:  97833 total, 171059 local destination
  0 hdr errors, 0 addr errors
         167 unkn proto, 0 discards
   Frags: 4 reassembled, 30 reasm timed out, 8 reasm req
         0 reasm fails, 145 frag ok, 0 frag fail
         290 frag creates
 Sent:  15 forwarded, 25144 generated, 0 out disc
         0 no routes,0 routing discards
  Route: 57680 routes in table
    0 timestamp req, 0 timestamp rpy
        0 addr mask req, 0 addr mask rpy
ICMP statistics:
  Rcvd: 561 total, 0 errors, 15 dst unreach
        0 time exceed, 0 param probs, 0 src quench
        0 redirects, 0 echo req, 0 echo rpy
        0 timestamp req, 0 timestamp rpy
        0 addr mask req, 0 addr mask rpy
  Sent: 463866 total, 0 errors, 163676 dest unreach
        0 time excd, 0 param prob, 0 src quench
        20 redirects, 463846 echo req, 0 echo rpy
        0 timestamp req, 0 timestamp rpy
        0 addr mask req, 0 addr mask rpy
UDP Statistics:
  Rcvd: 93326 total, 0 checksum errors, 90610 no port
  Sent: 0 total, 0 errors
TCP Global Statistics:
  Connections: 7358 attempted, 4 accepted, 7362 established
        0 dropped, 14718 closed
Rcvd: 75889 total pkts, 53591 in-sequence pkts, 3120283 bytes
        0 chksum err pkts, 0 authentication err pkts, 0 bad offset
        0 short pkts, 0 duplicate pkts, 0 out of order pkts
  Sent: 82318 total pkts, 44381 data pkts, 656321 bytes
        34 retransmitted pkts, 487 retransmitted bytes
OSPF Statistics:
IGMP Statistics:
ARP Statistics:
```

show ip udp statistics

- Use to display UDP statistics.
- Field descriptions
  - UDP Statistics Rcvd:
    - total—Total number of UDP packets received
    - checksum—Number of checksum error packets received
    - no port—Number of packets received for which no E-series router application listener was listening on the destination port
  - UDP Statistics Sent:
    - total—Total number of UDP packets sent
    - errors—Number of error packets sent
- Example

```
host1#show ip udp statistics
UDP Statistics:
  Rcvd: 39196 total, 0 checksum errors, 29996 no port
  Sent: 210 total, 0 errors
```

show profile brief

- Use to list all profile names.
- Field descriptions
  - Profile—Profile names
- Example

```
host1#show profile brief
Profile :
foo
trill
profile4
```

show route-map

- Use to display the configured route maps.
- The displayed information includes the instances of each access list such as **match** and **set** commands.
- Example

```
host1(config)#route-map westford permit 10
host1(config-route-map)#match community 44
host1(config-route-map)#set local-pref 400
host1(config-route-map)#exit
host1(config)#exit
host1#show route-map westford
route-map 1, permit, sequence 10
  Match clauses:
    match community 44
  Set clauses:
    set local-pref 400
```