## Chapter 22

# Internet Protocol (IP)

# Introduction

This chapter describes features of the Internet Protocol (IP), support for IP on the router, and how to configure and operate the router to route IP protocols. This chapter describes IPv4. For information on IPv6 and the router's implementation of it, see Chapter 31, Internet Protocol version 6 (IPv6).

IP protocols are widely used and available on nearly every host and PC system. They provide a range of services including remote login, file transfer, and Email. Using IP routers allows these services to be fully supported within an organisation and to other organisations internationally.

IP is often referred to as TCP/IP where TCP refer to Transmission Control Protocol. This is a protocol that runs over IP and provides end-to-end reliability and control of IP network connections. A closely related protocol called UDP (User Datagram Protocol) also runs over IP and is used where reliable transport of datagrams is not required. Both TCP and UDP are used by modules in the router.

The router is capable of routing IP data packets via the wide area network. This allows a group of remote LANs to be joined together as a single IP autonomous system and to be connected to other IP networks, such as the Internet.

Some interface and port types mentioned in this chapter may not be supported on your router. The interface and port types that are available vary depending on your product's model, and whether an expansion unit (PIC, NSM) is installed. For more information, see the Hardware Reference.

As well as the familiar Internet, with uppercase "I", the term *internet* (with lowercase "i") can refer to any network (usually a wide area network) that uses the Internet protocol. This chapter concentrates on this definition—a generalised network that uses IP as its transport protocol.

# IP Packets

The basic unit of data sent through an internet is a *packet* or *datagram*. An IP network functions by moving packets between routers and/or hosts. A packet consists of a *header* followed by the *data* (Figure 22-1 on page 22-6, Table 22-1 on page 22-6). The header contains the information necessary to move the packet across the internet. It must be able to cope with missing and duplicated packets as well as possible fragmentation (and reassembly) of the original packet.

Packets are sent using a *connectionless* transport mechanism. A connection is not maintained between the source and destination addresses; rather, the destination address is placed in the header and the packet is transmitted on a best effort basis. It is up to the intermediate systems (routers and gateways) to deliver the packet to the correct address, using the information in the header. Successive packets may take different routes through the network to the destination. There is a close analogy with the postal delivery system in that letters are placed in individually addressed envelopes and put into the system in the 'hope' that they will arrive. Like an internet, the postal system is very reliable. In an internet, higher layers (such as TCP and Telnet) are responsible for ensuring that packets are delivered in a reliable and sequenced way.

In contrast to a connectionless transport mechanism, a *connection-oriented* transport mechanism requires a connection to be maintained between the source and destination as long as necessary to complete the exchange of packets between source and destination. X.25 is an example of a connection-oriented protocol. A good analogy to a connection-oriented protocol is a telephone call in which both parties verify that they are talking to the correct person before exchanging highly sequenced data (because nothing intelligible results when both talk at the same time), and the connection is maintained until both parties have finished talking. It is not hard to imagine the chaos if the telephone system delivered words in the wrong order.

Figure 22-1: Format of an IP datagram

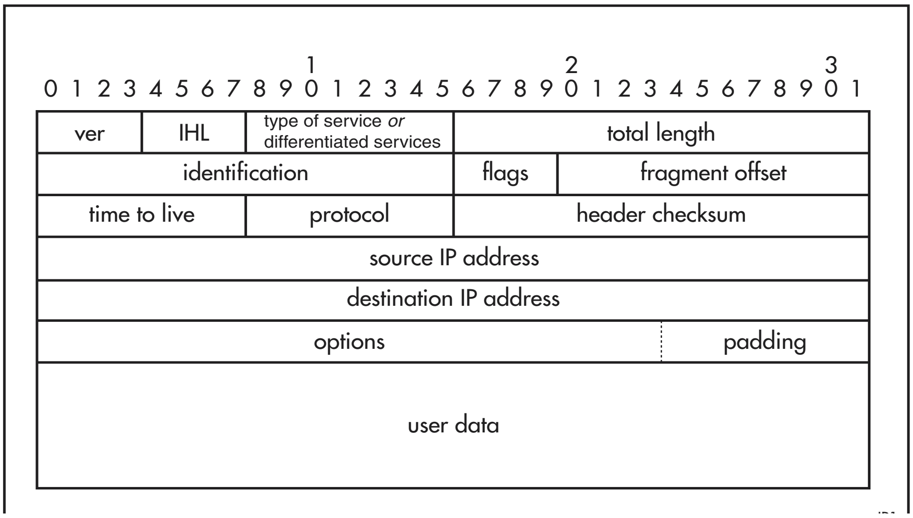


Table 22-1: Functions of the fields in an IP datagram

| **Field** | **Function** |
|---|---|
| ver | Version of the IP protocol that created the datagram. |
| IHL | Length of the IP header in 32-bit words (5 is minimum value). |
| Type of service *or* differentiated services | Type of Service indicates the quality of service (precedence, delay, throughput, and reliability) desired for the datagram.<br>Differentiated Services supersedes this. It contains the 6-bit DSCP and is used to sort traffic as part of a Quality of Service system. For more information, see Chapter 39, Software Quality of Service (QoS) and RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*. |
| Total length | Length of the datagram (both header and user data), in octets. |
| Identification | 16-bit value assigned by the originator of the datagram, used during reassembly. |
| Flags | Control bits indicating whether the datagram may be fragmented, and if so, whether other later fragments exist. |
| Fragment offset | For fragmented datagrams, offset in the original datagram of the data being carried in this datagram. |
| Time to live | Time in seconds the datagram is allowed to remain in the internet system. |
| Protocol | High level protocol used to create the message (analogous to the type field in an Ethernet packet). |

Table 22-1: Functions of the fields in an IP datagram (cont.)

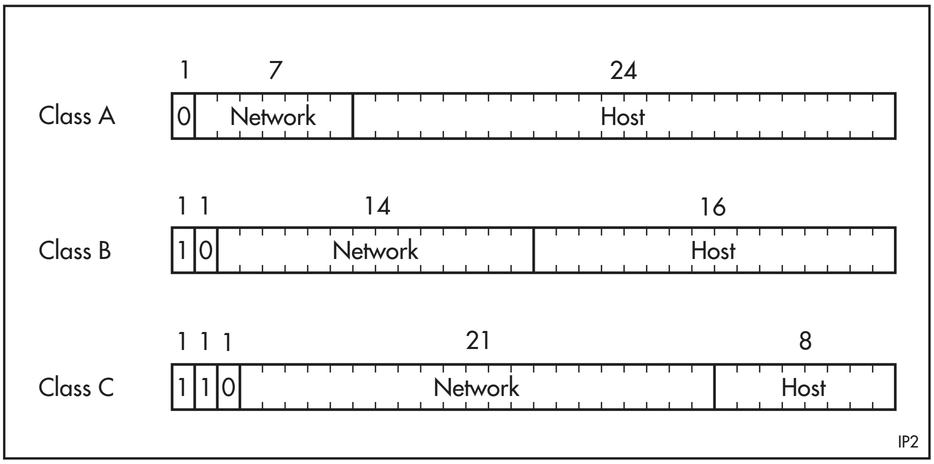
| Field | Function |
|---|---|
| Header checksum | Checksum of the header. |
| Source IP address | 32-bit IP address of the sender. |
| Destination IP address | 32-bit IP address of the recipient. |
| Options | Optional field primarily used for network testing or debugging. |
| Padding | All bits set to zero—used to pad the datagram header to a length that is a multiple of 32 bits. |
| User data | Actual data being sent. |

# Addressing

Internet addresses are fundamental to the operation of the TCP/IP internet. Each packet must contain an internet address to determine where to send the packet. Most packets also require a source address so that the sender of the packet is known. Addresses are 32-bit quantities that are logically divided into fields. They must not be confused with physical addresses (such as an Ethernet address); they serve to address Internet Protocol packets only. Addresses are organised into the classes described in the following table.

| Class | Maximum number of possible networks | Maximum number of hosts per network |
|---|---|---|
| A | 127 | 16,777,216 |
| B | 16,384 | 65,536 |
| C | 2,097,152 | 255 |
| D | Reserved Class | |
| E | Reserved Class | |

Each class differs in the number of bits assigned to the host and network portions of the address. The following figure shows the subdivision of the 32 bits into network and host fields for class A, B, and C networks.



The addressing scheme lets routers efficiently extract the host and network portions of an address. In general, a router is interested only in the network portion of an address.

Class A sets the Most Significant Bit (MSB) to 0 and allocates the next 7 bits to define the network and the remaining 24 bits to define the host. Class B sets the two MSBs to 10 and allocates the next 14 bits to designate the network while the remaining 16 refer to the host. Class C sets the three MSBs to 110 and allocates the next 21 bits to designate the network. The remaining 8 are left to the user to assign as host or subnet numbers.

The term *host* refers to any attached device on a subnet, including PCs, mainframes, and routers. Most hosts are connected to only one network; that is, they have a single IP address. Routers are connected to more than one network and can have multiple IP addresses. The IP address is expressed in *dotted decimal notation* by taking the 32 binary bits and forming 4 groups of 8 bits, each separated by a dot. For example:

```
10.4.8.2 is a class A address

10 is the DDN assigned network number
  .4.8 are (possibly) user assigned subnet numbers
      .2 is the user assigned host number


172.16.9.190 is a class B address

172.16 is the DDN assigned network number
      .9 is the user assigned subnet number
        .190 is the user assigned host number
```

The value 0.0.0.0 defines the default address, while a value of all ones in a host portion (such as 255) is reserved as the broadcast address. Some older versions of UNIX use a broadcast value of all zeros, therefore both the value ‘0’ and the value ‘255’ are reserved within any user assigned host portion. The address 172.16.0.0 refers to **any** host (**not** every host) on **any** subnet within the class B address 172.16. Similarly 172.16.9.0 refers to **any** host on subnet 9, whereas 172.16.9.255 is a packet addressed to **every** host on subnet 9. The router uses this terminology to indicate where packets are to be sent.

An address with 0 in the host portion refers to ‘this particular host’ while an address with 0 in the network portion refers to ‘this particular network’. As mentioned above, a value of all ‘1’ (255) is a broadcast. To reduce loading, IP consciously tries to limit broadcasts to the smallest possible set of hosts; hence, most broadcasts are *directed*. For example 172.16.56.255 is a broadcast to subnet 56 of network 172.16.

A major problem with the IP type of addressing is that it defines connections, not hosts. A particular address, although it is unique, defines a host by its connection to a particular network. Therefore, if the host is moved to another network, the address must also change. The situation is analogous to the postal system. A related problem can occur when an organisation with a class C address finds that they need to upgrade to class B. This involves a total change of every address for all hosts and routers. Thus the addressing system is not scalable.

# Subnets

The growth of the Internet has meant a proliferation in the number of addresses that core routers must handle. More addresses mean more loading, which tends to slow the system down. This can be overcome by minimising the number of network addresses by sharing the same IP prefix (the assigned network number) with multiple physical networks. Generally these would all be within the same organisation although not required. There are two main ways of achieving this: subnetting and proxy ARP. Proxy Address Resolution Protocol (ARP) is discussed in "Address Resolution Protocol (ARP)" on page 22-13.

A subnet is formed by taking the host portion of the assigned address and dividing it into two parts. The first part is the 'set of subnets' while the second refers to the hosts on **each** subnet. For example, the DDN may assign a class B address as 172.16.0.0. The system manager would then assign the lower two octets in some way that makes sense for the network. A common method for class B is to simply use the higher octet to refer to the subnet. Thus there are 254 subnets (0 and 255 are reserved) each with 254 hosts. These subnets need not be physically on the same media. Generally they would be allocated geographically with subnet 2 being one site, subnet 3 another and so on. Some sites may have a requirement for multiple subnets on the same LAN. This could be to increase the number of hosts or simply to make administration easier. In this case, it is normal (but not required) that the subnets be assigned contiguously for this site. This makes the allocation of a subnet mask easier. This mask is needed by the routers to ascertain which subnets are available at each site. Bits in the mask are set to 1 if the router is to treat the corresponding bit in the IP address as belonging to the network portion, or set to 0 if it belongs to the host portion. This allows a simple bit-wise logical "and" to determine if the address should be forwarded.

Although the standard does not require that the subnet mask select contiguous bits, it is normal practice to do so. To do otherwise can make the allocation of numbers rather difficult and prone to errors. Some example masks are:

```
11111111.11111111.11111111.00000000 = 255.255.255.0
<----network----> <subnet> <-host->
```

This would give 254 subnets on a class B network, each with 254 hosts.

```
11111111.11111111.11111111.11110000 = 255.255.255.240
<----network----> <--subnet--><host>
```

This would give 4094 subnets on a class B network, each with 14 hosts, or 14 subnets on a class C network each with 14 hosts.

The official description of subnetting is given in RFC 950. Subnet information and IP addresses are added to the router with the **add ip interface** command on page 22-80 and the **set ip interface** command on page 22-155.

# Assigning an IP Address

To configure the router to perform IP routing (for example, to access the Internet) you need to configure IP. You also need to configure IP if you want to manage the router from a Telnet session or with the GUI.

The AR415S, AR440S, AR441S, and AR442S have IP enabled and an IP address assigned by default. For the default configuration, see the *AR400 Series Router Installation and Safety Guide*. To change the IP address for an interface, enter the command:

```
set ip interface=interface ipaddress=ipadd mask=ipadd
```

To configure IP, first enable IP if necessary, by using the command:

```
enable ip
```

Then add an IP address to each of the router interfaces that you want to process IP traffic. These may include the default VLAN (vlan1), the DMZ (eth1 on the AR450S), and the WAN Ethernet port (eth0).

For the default VLAN, use the command:

```
add ip interface=vlan1 ipaddress=ipadd mask=mask
```

where:

- *ipadd* is an unused IP address on your LAN.
- *mask* is the subnet mask (for example 255.255.255.0)

If IP addresses on your LAN are assigned dynamically by DHCP, you can set the router to request an IP address from the DHCP server by using the commands:

```
add ip interface=vlan1 ipaddress=dhcp
```

```
enable ip remoteassign
```

You do not need to set the **mask** parameter because the subnet mask received from the DHCP server is used.

If you use DHCP to assign IP addresses to devices on your LAN, and you want to manage the router within this DHCP regime, it is recommended that you set your DHCP server to always assign the same IP address to the router. This lets you access the GUI by browsing to that IP address, and also lets you use the router as a gateway device for your LAN. If you need the router's MAC address for this, it can be displayed by using the **show switch** or **show eth=x macaddress**.

Similarly, for the default WAN Ethernet port on the AR450S (eth0) use the command:

```
add ip interface=eth0 ipaddress=ipadd mask=mask
```

where *ipadd* is the globally-unique IP address that your ISP has assigned to you.

For the default DMZ interface on the AR450S, use the command:

```
add ip interface=eth1 ipaddress=ipadd mask=mask
```

where *ipadd* is an unused private or public IP address.

To protect servers on your DMZ (or LAN), you need to configure the firewall (see Chapter 46, Firewall, especially "Configuration Examples" on page 22-54).

To change the IP address for an interface, enter the command:

```
set ip interface=interface ipaddress=ipadd mask=ipadd
```

If you are configuring the router remotely and you change the configuration (for example, the VLAN membership) of the port over which you are configuring, the router is likely to break the connection.

# Multihoming

The router can be configured as a *multihomed* device with multiple IP addresses. Up to 16 logical IP interfaces can be added to a single Layer 2 interface, such as eth0, vlan1 or ppp0, and up to a total of 1280 logical interfaces per router.

An IP interface name is formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Ethernet interfaces can be 0 to 999. For example, eth0-0 is the first logical IP interface assigned to the Ethernet 0 interface and ppp1-8 is the ninth logical IP interface assigned to the ppp1 interface. If a logical interface is not specified, 0 is assumed. For example, 'ppp0' is equivalent to 'ppp0-0'.

Each logical interface has its own unique IP address and mask, and can be assigned its own traffic filter, policy filter, priority filter, GRE entity and security association. Each logical interface has its own interface counters and can be enabled, disabled, or reset independently of other logical interfaces assigned to the same Layer 2 interface. Each additional logical interface created on a Layer 2 interface adds an extra entry to the IP Address Table in the SNMP MIB-II MIB. See Appendix C, SNMP MIBs for a complete description of the objects in MIB-II.

The router does not support a single logical interface being associated with multiple physical interfaces in order to increase the reliability or throughput between directly connected machines by providing alternative physical paths between them. This functionality is provided by Layer 2 multiplexing schemes such as PPP multilink.

# Local Interfaces

A local interface is one that is always available for higher layer protocols to use and advertise to the network. Although a local interface is assigned an IP address, it does not have the usual requirement of connecting to a lower layer physical entity. This lack of physical attachment creates the perception of a local interface always being accessible via the network.

Local interfaces can be utilised by a number of protocols for various purposes. They can be used to improve access to a router, as well as increasing its reliability, security, scalability and protection. In addition, local interfaces can add flexibility and simplify management, information gathering and filtering.

One example of this increased reliability is for OSPF to advertise a local interface as an interface-route into the network irrespective of the physical links that may be "up" or "down" at the time. This provides a higher probability that the routing traffic will be received and subsequently forwarded. Further reliability and performance could be provided by configuring parallel BGP paths to a local interface on a peer device, which would result in improved load sharing.

Access and security can be improved through filtering. Incoming traffic can be filtered by rules that specify local interfaces as the only acceptable destination addresses.

Information gathering and filtering as well as management can potentially be simplified if protocols such as SNMP use local interfaces for receiving and sending trap and log type information.

A local interface is *virtual* in the sense that it is not associated with a physical interface. Each local interface can be assigned an IP address, which can then be used as the source address of IP packets generated internally by IP protocols such as RIP, OSPF, BGP, and PING. These higher layer protocols must assign a source IP address to packets passed to IP for forwarding. The router uses the following rules to determine the source IP address in packets sent by higher layer protocols, in order of preference:

1. If the higher layer protocol's configuration specifies a source IP address to use, then the configured address is used as the packet's source IP address.

   For example, the **sipaddress** parameter in the **ping** command on page 22-135 specifies the source IP address to use in ping packets.

2. If a local IP interface has been assigned an IP address, then the IP address of that local interface is used as the packet's source IP address.

   For example, if OSPF is set to use a local interface (**set ospf** command on page 26-59 of Chapter 26, Open Shortest Path First (OSPF), it uses the IP address specified by the **add ip local** command on page 22-86 or the **set ip local** command on page 22-160.

3. Otherwise, the IP routing module determines the interface over which the packet is to be transmitted, and assigns the IP address of the interface as the packet's source IP address.

To add a new local interface, use the **add ip local** command on page 22-86.

To change the options for an existing local interface, including the default local interface, use the **set ip local** command on page 22-160.

To delete a new local interface, use the **delete ip local** command on page 22-102.

For information about how to configure a higher layer protocol to use the local interface, see the chapter for that protocol. For example, see "How to Set the IP Address that Identifies the Router" on page 27-40 of Chapter 27, Border Gateway Protocol version 4 (BGP-4).

# Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) is used by the router to dynamically learn the location of devices in its networks. Most hosts also have a media-dependent physical address as well as the assigned IP address. For Ethernet, this is a 6-byte, globally unique number. ARP enables the router to learn the physical address of the device that has a given IP address.

When the router receives a packet with an unknown target IP address, it broadcasts an ARP request to determine where to send that packet. The ARP request is a broadcast packet and includes the target IP address. All stations on the LAN receive this broadcast but only one host recognises its own IP address. It replies, thereby giving the router its physical address.

The router creates a dynamic ARP entry in a cache, to record the IP address to physical address mapping (also called a *binding*). It uses that ARP entry to forward further packets to that address.

To display the current contents of the router's ARP cache, use the **show ip arp** command on page 22-181.

The ARP protocol is described in RFC 826, *An Ethernet Address Resolution Protocol—or—Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*.

## Static ARP Entries

If your LAN includes hosts that do not support ARP, you can add a static ARP entry to the cache. However, it is rarely necessary to add an ARP entry this way. Use the command:

```
add ip arp=ipadd [other parameters]
```

Valid ARP entries have a unicast IP address with a unicast MAC address. Devices using multicasting do not need an ARP entry. See the section "IP Multicasting" on page 22-47 for further information.

To modify existing static ARP entries, use the command:

```
set ip arp=ipadd [other parameters]
```

To delete existing static ARP entries, use the command:

```
delete ip arp={ipadd|alldynamic}
```

# Adding Static ARP Entries with Multicast MAC Addresses

Valid ARP entries are normally restricted to unicast IP with unicast MAC addresses. However, ARP entries can be configured with multicast MAC addresses when **macdisparity** is enabled. Static ARP entries with multicast MAC addresses are necessary for some third party networking solutions, such as server clustering.

Before you can add an ARP entry with a multicast MAC address, you must enable **macdisparity** using the command:

```
enable ip macdisparity
```

Once this feature is enabled, you can add an ARP entry with a multicast MAC address using the **add ip arp** command.

## Accepting packets with conflicting addresses

Enabling **macdisparity** also allows the router to accept packets with conflicting IP and MAC addresses. Normally the router discards these packets as being invalid.

Conflicting IP and MAC addresses include:

- A multicast IP address with a unicast MAC address
- A unicast IP address with a multicast MAC address

**Macdisparity** is disabled by default. When disabled, only ARP entries with unicast IP and MAC addresses can be added, and packets with conflicting addresses are discarded. Other routers in the network may not accept packets with conflicting addresses unless configured to. To disable this functionality, use the command:

```
disable ip macdisparity
```

ARP entries with multicast MAC addresses must be removed before the **disable ip macdisparity** command works. To see details on the current ARP entries, use the command:

```
show ip arp
```

To see whether **macdisparity** is enabled or disabled, use the command:

```
show ip
```

For an example of how to use ARP entries with multicast MAC addresses, see *How to Use Windows 2003 Network Load Balancing Clustering with Allied Telesis Switches*. This is available from the Resource Center on your Documentation and Tools CD-ROM, or from www.alliedtelesis.co.uk/en-gb/solutions/techdocs.asp?area=howto.

## Timing Out ARP Entries

The router times out dynamic ARP entries to ensure that the cache does not fill with entries for hosts that are no longer active. If the router stops receiving traffic for the device specified in a dynamic ARP entry, it deletes the ARP entry after a configurable timeout period. Static ARP entries are not aged or automatically deleted.

The ARP cache has 256 hash buckets. When the router learns a new entry, it hashes its IP address and uses the hash to determine into which bucket to place the entry. Therefore the location of an entry in the cache depends on its IP address, and is not related to its age or the time at which the router learns it.

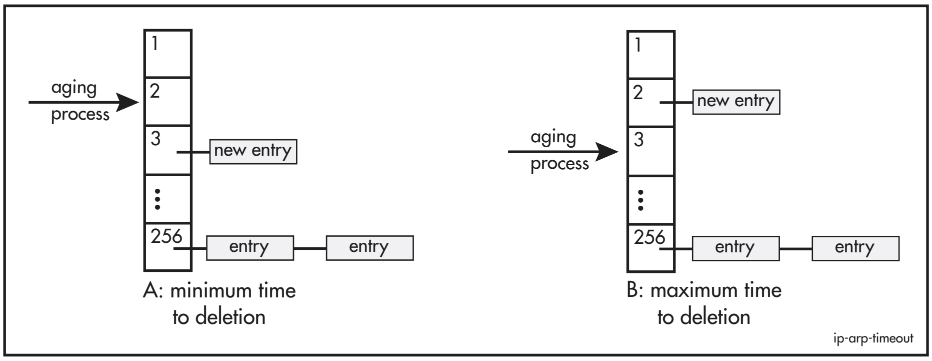
To age out old entries in the cache, an aging process moves through the cache and marks entries. If the router uses the entry to forward traffic, it removes the mark. This removal is referred to as *refreshing* the ARP cache. You can disabled ARP cache refreshing (see **Always deleting entries**) but it is enabled by default. Because ARP cache refreshing removes the mark from entries with traffic, only out-dated entries retain the mark. On the next pass of the aging process, it checks for entries that are still marked and deletes them.

You can configure the speed at which the aging process progresses through the cache, by using the command:

```
set ip arp timeout=multiplier
```

The aging process takes 1 x *multiplier* seconds to process each bucket. The default multiplier is 4, so it takes 4 seconds per bucket, which means 1024 seconds for a single pass through the whole cache. This command is described in more detail in the following **Changing the timeout** section.

The time it takes before the router deletes an entry varies, because it depends on how far away the aging process was from the entry’s bucket when the entry was written. The following figure shows the difference between the minimum and maximum times to deletion. In scenario A, the aging process reaches the new entry as soon as possible. It takes 1 time unit (4 seconds by default) to reach the new entry, and therefore 1024 seconds to reach it a second time and delete it. In scenario B, the aging process takes as long as possible to reach the new entry. It takes 256 time units (1024 seconds by default) to reach the new entry, and therefore 2048 seconds to reach it a second time and delete it.



### Changing the timeout

Increasing the ARP timeout reduces the amount of network traffic because it reduces the risk of deleting a valid entry. Decreasing the timeout makes the router more responsive to changes in network topology.

As described above, the default timeout is between 1024 and 2048 seconds. Instead of changing the timeout by changing the number of seconds, you change the multiplier that determines the speed of the cache's aging process. To do this, use the command:

```
set ip arp timeout=multiplier
```

where *multiplier* is an integer from 1 to 1023, and is 4 in the router's default state. To increase the timeout, increase the multiplier. To decrease the timeout, decrease the multiplier.

For example, to delete ARP entries as quickly as possible, which is after 256 to 512 seconds without traffic, use the command:

```
set ip arp timeo=1
```

### Checking the entry before deleting

By default, the router deletes dynamic ARP entries when they timeout, without checking whether the addresses are still valid on the network. You can prevent the router from deleting valid entries, by enabling it to send ARP requests for the addresses before deleting the entry. To enable the router to do this, use the command:

```
enable ip arp agepoll
```

### Always deleting entries

By default, the router only deletes entries that it has not used since the previous pass of the aging process through the cache. This is because it only deletes entries if they have been marked for deletion, and when the router receives traffic for the entry, it removes the mark. The process of removing the marks is called ARP cache refreshing.

If instead you want the router to always delete entries, whether or not there has been traffic to their address, you can turn off ARP cache refreshing. To do this, use the command:

```
set ip arp refresharp=off
```

When ARP cache refreshing is off, every entry gets a deletion mark when the aging process first reaches it, and still has that mark when the aging process reaches it for a second time. Therefore, without ARP cache refreshing, all entries time out after 1024 to 2048 seconds (by default), or after the configured timeout if it is changed from the default (see **Changing the timeout**).

## Deleting ARP Entries

If you swap a device in your network for another device that has the same IP address, the router may be left with a stale ARP entry and be unable to forward packets to the new device. This is most likely if you swap in the device without taking the link to the router down, for example, if it connects through a hub. Instead of waiting for such entries to time out, you can delete them.

You can delete individual ARP entries, or you can delete all dynamic entries in a single step. Deleting all dynamic entries is particularly useful if you do not know the relevant IP addresses.

To delete a single entry, use the command:

```
delete ip arp=ipadd
```

To delete all dynamic ARP entries, use the command:

```
delete ip arp=alldynamic
```

The **alldynamic** option does not delete static (manually-entered) ARP entries.

The router generates a log message to record that dynamic ARP entries have been deleted.

The router replaces the deleted ARP entries when it receives traffic for the relevant addresses. As long as the entries are relearnt quickly enough, deleting dynamic ARP entries does not affect:

- routes
- OSPF neighbour status
- BGP peer status
- the TCP/UDP connection status
- VRRP status

# Proxy ARP

The router uses a technique called *Proxy ARP* (defined in RFC 1027) to allow hosts that do not support routing (i.e. they have no knowledge of the network structure) to determine the physical addresses of hosts on other networks. The router intercepts ARP broadcast packets and substitutes its own physical address for that of the remote host. This occurs only when the router has the *best* route to the remote host. By responding to the ARP request, the router ensures that subsequent packets from the local host are directed to its physical address, and it can then forward these to the remote host. The process is symmetrical. Proxy ARP is enabled by default for Eth and VLAN interfaces. It can be enabled or disabled selectively using the command:

```
set ip interface=interface
   proxy={defroute|local|off|on|strict|}
```

## Local proxy ARP

Local proxy ARP lets you stop MAC address resolution between hosts within an interface’s subnet. This ensures that devices within a subnet cannot send traffic that bypasses the router. This lets you monitor, filter, and control traffic between devices in the same subnet.

Local proxy ARP extends proxy ARP by intercepting and responding to ARP requests between hosts within a subnet. Local proxy ARP responds to ARP requests with the router’s own MAC address details instead of those from the destination host. This stops hosts from learning the MAC address of other hosts within its subnet.

When local proxy ARP is operating on an interface, the router does not generate or forward any ICMP-Redirect messages on that interface.

To create an interface that uses local proxy ARP, use the command:

```
add ip interface=interface ipaddress={ipadd|dhcp}
   proxyarp=local [other-options]
```

To change an interface to use local proxy ARP, use the command:

```
set ip interface=interface ipaddress={ipadd|dhcp}
   proxyarp=local [other-options]
```

To display details about interfaces assigned to the IP module, including whether Proxy ARP is enabled on each interface, use the **show ip interface** command on page 22-202.

# MAC Address Logging

You can log MAC addresses of the equipment connected to the router's LAN interfaces, and which access the WAN interface. This provides an audit trail should anyone hack into the system. If you use NAT, the Layer 2 MAC address must be logged in addition to the IP address.

To enable MAC address logging, use the **enable ip arp log** command on page 22-122.

To disable MAC address logging, use the **disable ip arp log** command on page 22-110.

## Waiting for a Response to an ARP Request

When a router receives a packet and does not have an ARP entry for the destination address, it broadcasts an ARP Request message over the egress IP interface. If the router does not receive a reply within a particular time, it notifies the sending device that the destination is unknown.

You can increase the length of time that the router waits for a response, which is useful for routers that communicate with devices that are slow to respond. To configure the waiting time, use the following command to specify the wait timeout period in seconds:

```
set ip arpwaittimeout=1..30
```

The default is 1 second.

The easiest way to test a changed wait timeout period is to ping an unavailable device. The timeout determines the delay between pinging an IP address and receiving the reply that the device is unreachable.

# DHCP Client

IP interfaces can be configured with a static IP address, or with a dynamic IP address assigned by DHCP (Dynamic Host Configuration Protocol). To configure an IP interface to use an address assigned by DHCP, set the **ipaddress** parameter to DHCP in the **add ip interface** command on page 22-80 and the **set ip interface** command on page 22-155.

When the **ipaddress** parameter of an IP interface is set to DHCP rather than a static IP address, the router's DHCP client obtains the IP address and subnet mask for the interface, and other IP configuration parameters, from a DHCP server. See the description of the **add ip interface** command on page 22-80 for a list of the DHCP reply parameters the router uses to configure IP interfaces.

For example, to configure interface eth0 to obtain its IP address and subnet mask from DHCP, use the command

```
set ip interface=eth0 ipaddress=dhcp
```

If an IP interface is configured to get its IP address and subnet mask from DHCP, the interface does not take part in IP routing until the IP address and subnet mask have been set by DHCP.

Remote address assignment must be enabled with the **enable ip remoteassign** command on page 22-131 before IP interfaces can accept addresses dynamically assigned by DHCP.

# ICMP

The Internet Control Message Protocol (ICMP) allows routers to send error and control messages to other routers or hosts. It provides the communication between IP software on one system and IP software on another. The router implements all non-obsolete ICMP functions (Table 22-2 on page 22-20). Some early systems may not fully implement all ICMP types. Often type 11 (Time To Live Exceeded) is not fully implemented.

The following ICMP messages can be disabled or enabled by the network manager:

- Network unreachable (RFC 792 Type 3 Code 0)
- Host unreachable (RFC 792 Type 3 Code 1)
- ICMP redirect messages (RFC 792 Type 5 Code 0, 1, 2, 3)

To enable ICMP messages, use the **enable ip icmpreply** command on page 22-127.

To disable ICMP messages, use the **disable ip icmpreply** command on page 22-114.

Table 22-2: ICMP messages implemented by the router

| ICMP Message Type | Router Response |
|---|---|
| Echo reply (0) | This is used to implement the **ping** command on page 22-135 that is common to most UNIX and TCP implementations. The router sends out an echo reply in response to an echo request. |
| Destination unreachable (3) | This message is sent when the router drops a packet because it did not have a route to the destination. |
| Source Quench (4) | The router sends this message when it must drop a packet due to limited internal resources. This could be because the source was sending data too fast to be forwarded. |
| Redirect (5) | The router issues this message to inform a local host that its target is located on the same LAN (no routing is required) or when it detects a host using a non-optimal route (usually because a link has failed or changed its status). For example, if the router receives a packet destined to its own MAC address, but with a destination IP address of another host in the local subnet, it sends back an ICMP redirect to the originating host. |
| Echo request (8) | This is related to (1) and results in an echo reply being sent. The router can also generate an echo request as a result of the **ping** command on page 22-135. |
| Time to Live Exceeded (11) | If the TTL field in a packet falls to zero, the router sends this message.This could occur when a route is excessively long or when too many hops are in the path. |

# Routing

The process of routing packets consists of selectively forwarding data packets from one network to another. The router must determine which network to send each packet to, and over which interface to send the packet in order to reach the desired network. This information is contained in the router's *routes*. For each packet, the router chooses the best route it has for that packet and uses that route to forward the packet. In addition, you can define filters to restrict the way packets are sent.

## Types of Routes

The router learns routes from static information entered as part of the configuration process and by listening to any configured routing protocols. The following types of routes are available on the router:

- Interface

  The router creates an interface route when you create the interface. This route tells the router to send packets over that interface when the packets are addressed to the interface's subnet.

- Dynamic

  The router learns dynamic routes from one or more routing protocols such as RIP or OSPF. The routing protocol updates these routes as the network topology changes.

- Static

  You can manually enter routes, which are then called static routes. For configuration instructions, see “Configuring Static Routes” on page 22-21. Uses of static routes include:

  - To specify the default route (to 0.0.0.0). If the router does not have another route to the packet’s destination, it sends it out the default route. The default route normally points to an external network such as the Internet.
  - To set up multiple networks or subnets. In this case you define multiple routes for a particular interface, usually a LAN port. This is a method of supporting multiple subnets on a single physical media.

## The Routing Table

The router maintains its routing information in a table of routes that tells the router how to find a remote network or host. Each route is uniquely identified in the table by its IP address, network mask, next hop, interface, protocol, and policy.

When the router receives an IP packet, and no filters are active that would exclude the packet, the router scans the routing table to find any matching routes on an “up” interface. If it finds multiple matching routes, then the router selects a route using the following process:

1. The router inspects the preference value of each candidate route and selects the route with the lowest preference value.
2. If multiple routes share the lowest preference value, then the router inspects the metric value of each of these routes and selects the route with the lowest metric.
3. If multiple routes share the lowest preference and metric values, then the router inspects the mask of each of these routes and selects the route with the longest mask.

If the router does not find a direct route to the destination, and no default route exists, the router discards the packet and sends an ICMP message to that effect back to the source.

The router maintains the routing table dynamically by using one or more routing protocols such as RIP or OSPF. These protocols act to exchange routing information with other routers or hosts.

## Configuring Static Routes

To create a static route, use the command:

```
add ip route=ipadd interface=interface nexthop=ipadd
    [circuit=miox-circuit] [dlci=dlci] [mask=ipadd]
    [metric=1..16] [metric1=1..16] [metric2=1..65535]
    [policy=0..7] [preference=0..65535] [tag=1..65535]
```

To define a default route, set **ipaddress** to 0.0.0.0 and **nexthop** to the network (router) where default packets are to be directed.

To define a subnet, set **ipaddress** to address of the new subnet, **nexthop** to 0.0.0.0, and **metric** to 1.

To modify an existing static route, use the command:

```
set ip route=ipadd interface=interface mask=ipadd
    nexthop=ipadd [circuit=miox-circuit] [dlci=dlci]
    [metric=1..16] [metric1=1..16] [metric2=1..65535]
    [policy=0..7] [preference=0..65535] [tag=1..65535]
```

To remove a static route altogether, use the command:

```
delete ip route=ipadd mask=ipadd interface=interface
    nexthop=ipadd
```

## Caching Routes

By default, the router caches routes to improve route lookup performance. The route cache holds the most recently used routes. When the router is determining the best route to a destination, it searches the cache first, using a hash function calculated from the destination information. If the router does not find a route in the cache, it searches the entire route table. If the router then finds a route in the route table, it adds that route to the cache.

To disable the cache, use the command:

```
disable ip route cache
```

To enable the cache, use the command:

```
enable ip route cache
```

To see the current contents of the route cache, use the command:

```
show ip route=[ipadd] cache
```

## Dynamic Routing Protocols

In all but the most simple networks, we recommend that you configure at least one dynamic routing protocol. Routing protocols enable the router to learn routes from other routers and switches on the network, and to respond automatically to changes in network topology. The following table describes the options.

| Protocol | Description | For more information |
|---|---|---|
| RIP | A relatively simple protocol particularly suitable for dynamically learning the interior structure of a network. *Interior* refers to routing within an organisation. | Chapter 25, Routing Information Protocol (RIP) |
| OSPF | A more complex protocol suitable for dynamically learning the interior and exterior structure of a network. *Exterior* refers to routing between organisations. | Chapter 26, Open Shortest Path First (OSPF) |
| BGP | A complex protocol capable of managing thousands of routes efficiently. | Chapter 27, Border Gateway Protocol version 4 (BGP-4) |

# Setting Preference of Dynamically-Learned Routes

To set the preference for all routes that the router learns from a particular protocol, use the command:

```
set ip route preference={default|1..65535}
    protocol={bgp-ext|bgp-int|ospf-ext1|ospf-ext2|ospf-inter|
    ospf-intra|ospf-other|rip|all}
```

This may be useful if you have more than one routing protocol defined because if the router has a choice of two valid routes it chooses the one with the lowest value for preference. For example, you can set all RIP routes to have a higher preference value—and therefore lower priority—than OSPF routes.

This command does not change the following:

- default preferences. Therefore, you can use **preference=default** to return to the original setting
- the preference for static routes. Use the **set ip route** command on page 22-164 instead.
- the preference for interface routes. These are always created with a lower preference value than dynamically learned routes, but can also be changed using the **set ip route** command.

# Displaying Route Information

To see the number of routes and other summary information, use the command:

```
show ip route general
```

To see all routes in the routing table, including both static and dynamic routes, use the command:

```
show ip route full
```

To see the number of octets sent and received using each route, use the command:

```
show ip route count
```

To see information about only routes to a particular subnet, specify the subnet address, use the command:

```
show ip route=ipadd [{cache|count|full}]
```

To see a list of all routes to a destination, with the most specific routes first, use the command:

```
show ip route=ipadd
```

The routes may have different metrics, next hops, policy or protocol. A list of routes is uniquely identified by its IP address and net mask.

# Equal Cost Multipath Routing

Equal Cost Multipath Routing (ECMP) allows the router to distribute traffic over multiple equal-cost routes to a destination. When the router sends packets to that destination, it distributes the packets across all equal-cost routes. The router considers a route to be equal cost if it has the same destination IP address, mask, preference, and metric. You can have up to 16 individual routes to each destination.

## Configuring ECMP

The following procedure explains how to use ECMP.

| Step | Command | Action |
|---|---|---|
| 1 | **enable ip route** multipath | If ECMP has been disabled, enable it. ECMP is enabled by default. |
| 2 | **add ip route**=*ipadd* interface=*interface* nexthop=*ipadd* [*other-options*...] | Add static routes as required. You can create multiple static routes to the same destination. |
| 3 | | Configure dynamic routing protocols as required. |
| 4 | **set ip route preference**=1..65535 protocol={bgp-ext\|bgp-int\|ospf-ext1\|ospf-ext2\|ospf-inter\|ospf-intra\|ospf-other\|rip\|all} | If you want routes from different routing protocols to have equal cost, give the protocols the same preference setting. |

To disable ECMP, use the command:

```
disable ip route multipath
```

# Routing Information Protocol (RIP)

Routing Information Protocol (RIP) is a simple distance vector routing protocol. It determines the number of hops between the destination and the router, where one hop is one link. Given a choice of routes, RIP uses the route that takes the lowest number of hops. If multiple routes have the same hop count, RIP chooses the first route it finds.

For more information, see Chapter 25, Routing Information Protocol (RIP).

# Open Shortest Path First (OSPF)

The Open Shortest Path First (OSPF) protocol is documented in RFC 1247. It has a number of significant benefits over RIP, including:

- OSPF supports the concept of areas to allow networks to be administratively partitioned as they grow in size.
- Load balancing, in which multiple routes exist to a destination, is also supported. OSPF distributes traffic over these links.

See Chapter 26, Open Shortest Path First (OSPF) for more details.

# Border Gateway Protocol (BGP)

The Border Gateway Protocol (BGP) is documented in RFC 1771. It is an external gateway protocol, designed to exchange routing information between pairs of routers in different routing domains.

See Chapter 27, Border Gateway Protocol version 4 (BGP-4) for more details.

# Metrics

Metrics are used to determine the criteria for using one route over another route. In this sense they *measure* some aspect of the route.

For RIP, the metric is simply the *hop count*, which is a measure of the number of links it takes to get to the specified destination, or in other words, how far away it is.

OSPF has a number of metrics. This, in part, accounts for its better performance in that it is not simply looking at one view of the network. For example, it can also allow for the fact that not all links have the same bandwidth. The router supports only one OSPF metric per interface.

## OSPF Auto Cost Calculation

OSPF interfaces can automatically set the OSPF metric of an IP interface on the basis of the bandwidth of the interface, instead of the system administrator manually setting the OSPF metric. For details, see “Automatic Cost Calculation” on page 26-8 of Chapter 26, Open Shortest Path First (OSPF).

# Policy-Based Routing

Policy routing is a way to route packets that is based on policies or rules set by the network manager. It is an alternative to priority-based routing and to destination routing protocols such as RIP and OSPF that use metrics to determine the shortest or optimal path to a destination. Policy-based routing is useful in providing equal access, protocol-sensitive routing.

The Type of Service (TOS) octet in the IP header comprises three fields: precedence (bits 0 to 2), TOS (bits 3 to 6) and MBZ (bit 7). The precedence field is intended to denote the importance or priority of the datagram, but is not commonly used. The MBZ field should always be zero and is currently unused. The TOS field denotes the type of service required and is used by the network to make trade-offs between throughput, delay, reliability, and cost. The TOS field is treated as an integer value between 0 and 15. RFC 1349 defines the semantics of five specific TOS values shown in the following table.

| Decimal | Binary | Meaning |
|---|---|---|
| 8 | 1000 | Minimise delay |
| 4 | 0100 | Maximise throughput |
| 2 | 0010 | Maximise reliability |
| 1 | 0001 | Minimise cost |
| 0 | 0000 | Normal service |

Although the semantics of the other values are undefined, they are legal TOS values and network devices must not prevent the use of these values in any way.

TOS values may be considered when determining the route to use for an IP packet. All routes have an assigned TOS value. This is normally the default TOS value, unless the route has been learned using a routing protocol that supports TOS, or the TOS value has been statically assigned.

To forward an IP packet, a router uses the packet's destination address to search for a route to the destination. If a route is not found, or if the selected route has an infinite metric, the destination is considered unreachable and the packet is discarded. If a single route is found with a finite metric, it is used. If more than one route is found with a finite metric, the TOS values of the selected routes can be used to refine the selection. A route with a TOS value identical to the TOS value in the IP packet is used in preference to a route with the default TOS value.

The router uses the TOS field in IP routes to implement policy-based routing of IP packets. However, since the TOS field in IP packets is not set or used by many IP implementations, the router makes use of filters to assign the TOS values used for policy routing to IP packets as they are received.

To enable policy routing, the first step is to create a filter to select the IP packets to be routed according to policy with the **add ip filter** command on page 22-70.

The policy filter is then assigned to an interface with the **add ip interface** command on page 22-80 or the **set ip interface** command on page 22-155. The **policyfilter** parameter specifies the policy filter to apply. Packets received via the interface are checked against the entries in the policy filter and if a match is found, the packet is routed according to the policy specified in the matching filter entry.

Note that a traffic filter, a policy filter, and a priority filter can be assigned to an interface.

- Policy and priority filters *affect* packets as they are transmitted, but traffic filters *affect* packets as they are received.
- Policy and traffic filters are *configured* on the receiving interface, but priority filters are *configured* on the transmitting interface.
- Policy and traffic filters are *applied* to packets as they are received, but priority filters are *applied* to packets as they are queued for transmission.

An interface may have a maximum of one traffic filter, one policy filter, and one priority filter, but the same traffic, policy, or priority filter can be assigned to more than one interface.

The final step is to create static routes and assign policy numbers to the routes by using the **add ip route** command on page 22-90 or the **set ip route** command on page 22-164.

When a packet is received via an interface with an assigned policy filter, and the packet matches an entry in the filter, the packet is routed using a route with the same policy number specified in the matching policy filter entry. For example, when a packet matches a policy filter entry that specifies a **policy** value of 3, the packet is routed using a route with a **policy** value of 3.

For IP packets routed according to policy numbers 0 to 7, the TOS octet in the packet's IP header is not modified. For IP packets routed according to policy numbers 8 to 15, the TOS field (bits 3 to 6) in the packet's IP header are set to the policy number less 8 and the packet is routed using a route with a policy equivalent to the policy number less 8. For example, if the policy filter assigns an IP packet a policy number of 14, the packets TOS field is set to 6 (14-8) and the packet is routed using a route with a policy of 6.

# Priority-Based Routing

Priority routing is a way to route packets according to priorities set by the network manager. It is an alternative to policy-based routing and to destination routing protocols such as RIP and OSPF that use metrics to determine the shortest or optimal path to a destination. Priority-based routing is useful in managing high priority interactive traffic and low priority batch traffic over the same link.

To enable priority routing, the network manager defines a set of priorities that make routing decisions. Each priority specifies the criteria by which to select IP packets, and routing actions to perform on the packets that match the criteria.

To create a filter to select IP packets based on priority and assign a priority, use the **add ip filter** command on page 22-70.

The **priority** parameter sets the priority of IP packets from p3 (highest) to p7 (lowest). The default is p5. Priority levels p0, p1, and p2 should not be used because they may conflict with the router's system activities.

Assign the priority filter to an interface by using either the **add ip interface** command on page 22-80 or the **set ip interface** command on page 22-155.

The **priorityfilter** parameter specifies the priority filter to use. Packets transmitted via the interface are checked against entries in this filter. When a match is found, the packet goes into a queue based on the packet's priority. Packets in higher priority queues are forwarded ahead of packets in lower priority queues.

Note that a policy filter, a priority filter, and a traffic filter can be assigned to an interface.

- Priority and policy filters *affect* packets as they are transmitted, whereas traffic filters *affect* packets as they are received.
- Priority filters are *configured* on the transmitting interface, whereas traffic and policy filters are *configured* on the receiving interface.
- Priority filters are *applied* to packets as they are queued for transmission, whereas traffic and policy filters are *applied* to packets as they are received

An interface can have only one traffic filter, one policy filter, and one priority filter. However, the same traffic, policy, or priority filter can be assigned to more than one interface.

## Route Templates

The router uses IP route templates to add IP routes to IP subnetworks discovered during normal operation by other protocols such as IPsec. This is necessary when IP traffic going to the subnetwork must be routed via a route other than the default route.

When a software module other than a routing protocol such as IPsec adds a route to the IP routing table, and is configured to use an IP route template, the software module supplies the IP network address and mask. The IP route template provides all the other parameters for the entry in the IP route table.

To create a route template, use the **add ip route template** command on page 22-93.

To delete a route template, use the **delete ip route template** command on page 22-106.

To modify an existing route template, use the **set ip route template** command on page 22-168.

To display a list of the currently defined route templates or information about a specific route template, use the **show ip route template** command on page 22-219.

# VLAN Tagging on Eth Interfaces

Eth ports on the router can route IP packets between VLANs by applying a VLAN tag to frames that are transmitted out of Eth ports. You can configure up to 1000 logical interfaces on each Eth port to route frames to multiple VLANs. To configure this, use one of the commands:

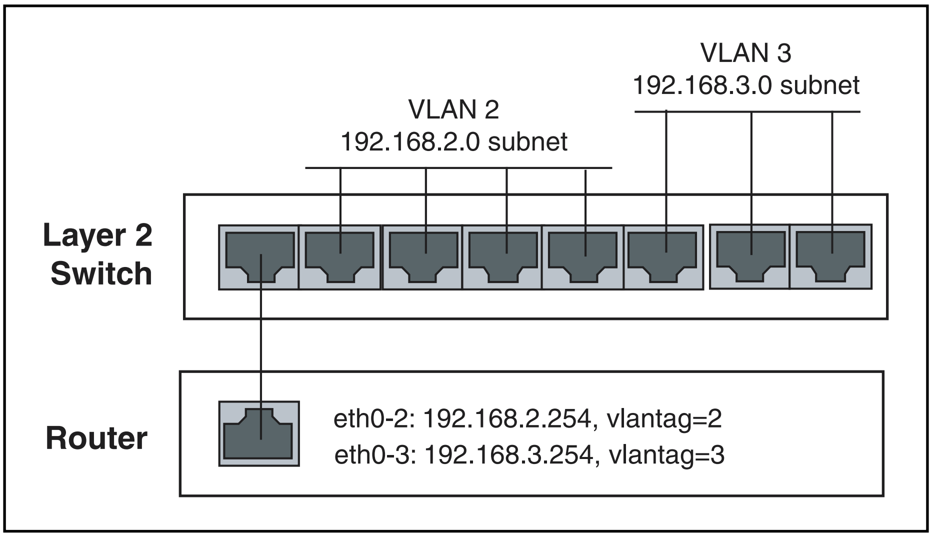
```
add ip interface=interface ipaddress={ipadd|dhcp}
    vlantag={1..4094|none} [vlanpriority={0..7|none}]
    [other-options...]

set ip interface=interface vlantag={1..4094|none}
    [vlanpriority={0..7|none}] [other-options...]
```

Note that if you have configured a large number of interfaces and are using the GUI to view them, the page may take several minutes to display.

**Example** In this scenario, the router is acting as both a gateway and a routing device for a Layer 2 LAN. The eth0 port on the router is connected to a Layer 2 switch through a port that is a tagged member of VLAN 2 and VLAN 3. Frames received on the eth0 port and destined for the Layer 2 switch are assigned the VID of the destination VLAN.

Figure 22-2: Example configuration for VLAN tagging on Eth interfaces



| VLAN on L2 switch | IP subnet | logical Eth interface on router | Gateway address on router | VLAN tag on Eth interface on router |
|---|---|---|---|---|
| vlan 2 | 192.168.2.0/24 | eth0-2 | 192.168.2.254 | 2 |
| vlan 3 | 192.168.3.0/24 | eth0-3 | 192.168.3.254 | 3 |

To configure the router, use the following commands:

```
add ip interface=eth0-2 ipaddress=192.168.2.254
    mask=255.255.255.0 vlantag=2

add ip interface=eth0-3 ipaddress=192.168.3.254
    mask=255.255.255.0 vlantag=3
```

Note that only these logical Eth interfaces transmit tagged frames. Traffic that is transmitted from other interfaces (or logical interfaces) on the router are untagged.

# Domain Name System (DNS)

The Domain Name System was implemented in order to provide users with a means to access remote systems by entering user friendly names rather than numeric IP addresses. Each name takes the form of a period delimited set of user names. This format is the host portion of a URL (Uniform Resource Locator).

From left to right these names begin with local user-related components and end in a domain related components such as .com .org etc. A network of domain name servers maintains the set of the mappings between every domain name and their respective IP addresses.

The network of domain name servers operates in a hierarchy that is similar to the structure of the names themselves and are constantly updating their databases of name to address mappings.

A practical example might help to understand the DNS operation. Among the names registered for Allied Telesis are: alliedtelesis.com, and alliedtelesis.co.uk. Each of these domain names maps respectively to the server IP address for the company's USA and UK web sites.

If a user tries to access our USA website, their PC sends a DNS enquiry to its local DNS server by using our domain name alliedtelesis.com. If this address is already locally cached (following its recent use), the local DNS server returns the IP address that corresponds to alliedtelesis.com, which the client server then inserts into the IP address field of the IP packet. If the domain name is unknown locally, the enquiry is forwarded upwards through the hierarchy of DNS servers until a name-to-address resolution can be made. All this operates transparently to the user who simply enters the domain name and waits for the connection to be established.

This is a very simplified overview. For users wishing to know more, we recommend en.wikipedia.org/wiki/dns This site provides a very good overview of DNS and cites many useful reference documents that includes a list of related RFCs for further reading.

## Usage of Domain Name Servers

The router uses domain name servers for the following two purposes:

- looking up FQDN (Fully Qualified Domain Name) addresses that are specified in commands
- performing DNS relay

### Looking up FQDN addresses specified in commands

A number of commands can specify FQDN addresses, including the following.

```
ping address

telnet address

trace address

finger user@address

mail to=user@address

add pki crl location=url
```

To resolve these FQDN addresses, the router sends a Domain Name System (DNS) request to a defined name server to translate the host name into an IP address.

When the router performs a DNS lookup, it first sends the request to its primary name server. If a response is not received within 20 seconds, it sends the request to its secondary name server.

### Automatically appending a domain name to a host name

You can set up the router to automatically add a domain name when you specify a hostname. To do this:

1. set the *sysName* MIB object to the router’s fully qualified domain name (such as router.company.com) by using the **set system name** command on page 4-31 of Chapter 4, Configuring and Monitoring the System
2. define a name server by using the **set ip nameserver** command on page 22-162

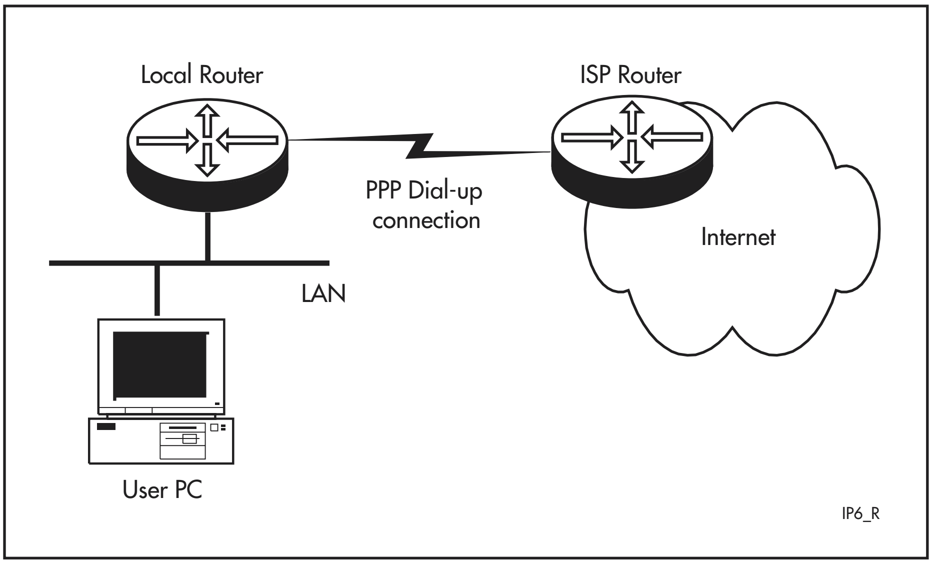
For example, if the system name is “router.company.com”, then entering the command **telnet mainhost** attempts a Telnet connection to the host “mainhost.company.com”, as long as “mainhost” is not an IP nickname. IP nicknames take precedence.

## Performing DNS relay

The DNS relay agent receives DNS requests from hosts and forwards them to the router's own configured DNS server. The DNS relay agent is disabled by default, and can be enabled by using the **enable ip dnsrelay** command on page 22-124. To disable it, use the **disable ip dnsrelay** command on page 22-111.

To display the current state of the DNS relay agent, use the **show ip** command on page 22-178.

Figure 22-3: Local router acting as a DNS relay for an Internet connection



# Configuring Domain Name Servers

The router can obtain the addresses of domain name servers in the following two ways:

- by static configuration
- by learning them dynamically

## Statically configuring domain name servers

To specify a DNS server, use the command:

```
add ip dns [domain={any|domain-name}]
    {interface=interface|primary=ipadd [secondary=ipadd]}
```

### Using different servers for different hosts

By using the **domain** parameter in the **add ip dns** command, you can configure different DNS servers for different host names. The **domain** parameter specifies a suffix that must be present on a host name in order for the router to use the name servers specified in the command.

If the **domain** parameter is not specified, the name servers are used as the default name servers. All DNS requests that do not match another specified domain are sent to the default name servers. This is equivalent to specifying **domain=any**.

For example, to add primary and secondary name servers with IP addresses of 192.168.10.1 and 192.168.10.2 respectively, for use when resolving host names in the domain apples.com, use the command:

```
add ip dns domain=apples.com primary=192.168.1.1
    secondary=192.168.1.2
```

If a request is sent for the domain www.fruit.apples.com, the DNS servers at 192.168.1.1 or 192.168.1.2 are used because the domain matches apples.com.

If a request is sent for the domain ftp.fruitpunch.apples.com, these DNS servers are also used because the domain matches apples.com.

If a request is sent for the domain www.armadillo.com, the domain does not match apples.com so these DNS servers are **not** used.

### Primary and secondary servers

Servers are defined as either primary or secondary. The first enquiry is always sent to the primary server. If the primary server is unavailable, a timeout occurs and the enquiry is then be referred to the secondary server.

For example, to add primary and secondary name servers with IP addresses of 202.36.163.1 and 202.36.163.3 respectively for use as default name servers, use the command:

```
add ip dns domain=any primary=202.36.163.1
    secondary=202.36.1.3
```

These servers are used for all host names that do not match any of the domains that are configured with their own set of name servers. For the example in the previous section, in which **domain=apples.com**, a request sent for www.armadillo.com would use the DNS servers at 202.36.163.1 and 202.36.163.3.

## Learning DNS server addresses dynamically

DNS addresses can be learnt by:

- DHCP option 6 on Eth and VLAN interfaces
- IPCP options 129 (Primary server) and 131 (secondary server) on PPP interfaces

If no name servers have been manually configured, and a name server address is learnt on Eth, PPP or VLAN interfaces by one of the means above, this server (or servers) is automatically used for the default name servers. Name servers configured in this way are identified by an "*" in the "Domain" column of the **show ip dns** output table (Figure 22-26 on page 22-193).

### Learning over a specified interface

Additionally, the **add ip dns** command on page 22-68 can specify a particular Eth, PPP or VLAN interface over which the router can learn primary or secondary DNS server addresses. Typically the PPP interface is a dial-up connection to an ISP that provides the DNS name server IPCP option.

When the PPP interface is already up when the host receives the DNS request, and a DNS server address was not set during IPCP negotiation, the DNS request is discarded. When the PPP interface is down, the interface is activated and IPCP negotiation is used to learn the DNS server address. When a DNS server address is learned as a result of the IPCP negotiation, the DNS request is forwarded to that address. Otherwise, the DNS request is discarded.

The above explanation applies to the local router acting as a DNS relay for a PC connecting to the Internet via an ISP.

### Over-ruling previous configuration

If the router was originally configured to learn name servers dynamically over a particular interface for use in resolving host names in the specified domain, this configuration can be overridden by specifying values for one or both of the static name server parameters (**primary** and **secondary**). Similarly, if static name server addresses were originally configured, use of the **interface** parameter causes name server information learned dynamically to overwrite the static name server configuration. Static name server addresses are lost.

Also, dynamically learnt name server addresses can be deleted with the **delete ip dns** command on page 22-97. A deleted automatic configuration may subsequently reappear if the interface concerned is reset.

### Related information: Sending DNS server addresses

If you are using a router as an access concentrator (therefore at the ISP end of the PPP link, rather than the client end), you can configure it to offer a specific DNS server address to the client router via IPCP. To do this, use the command:

```
set ppp dnsprimary=ipadd dnssecondary=ipadd
```

When the router is acting as a DHCP server, specify DNS servers for it to offer by using the **dnsserver** parameter in the **add dhcp policy** command on page 23-13 of Chapter 23, Dynamic Host Configuration Protocol (DHCP).

# DNS Caching

DNS caching allows the router to store recently requested domain or host addresses so they can be quickly retrieved if an identical DNS request is received. DNS caching reduces traffic on the Internet and improves performance for both DNS and DNS relay under heavy usage. The DNS cache has a limited size, and times out entries after a specified period of up to 60 minutes.

When a domain or host is requested, the cache is searched for a matching entry. If a match is found, a response is sent to the requesting PC or host. If a matching entry is not found, a request is sent to a remote server.

To add a DNS server to the list of DNS servers used to resolve host names into IP addresses, use the **add ip dns** command on page 22-68.

Once the DNS servers have been configured, set the configuration information with the **set ip dns** command on page 22-146.

For example, to add or set the IP addresses of the default primary and secondary name servers to 192.168.20.1 and 192.168.20.2 respectively, use the commands:

```
add ip dns primary=192.168.20.1 secondary=192.168.20.2
```

```
set ip dns primary=192.168.20.1 secondary=192.168.20.2
```

To set the DNS cache size and timeout values, use the **set ip dns cache** command on page 22-148.

To delete name server information from the DNS server, use the **delete ip dns** command on page 22-97.

# Dynamic DNS Client

You can configure the router as a dynamic DNS client, for operation with the service offered by DynDNS.com (see www.dyndns.com/services/dns/.

The dynamic DNS client allows you to associate the router's public IP address with up to five "hostnames" (FQDNs) registered with DynDNS.com. When the router's public IP address changes, the dynamic DNS client sends an HTTP update to DynDNS.com, notifying them of the address change. DynDNS.com then associates the new IP address with your registered hostnames, and propagates this change through the DNS system.

This means that you can host servers behind the router's public IP address without actually owning your own domain name, and even if your public IP address changes periodically.

## Configuring Dynamic DNS

To configure dynamic DNS, perform the following steps:

1. Register with DynDNS.com and select a domain name to use
2. Enable dynamic DNS, by using the command:

   ```
   enable ddns
   ```

3. Specify the hostname, interface, username, and password for the client to use, and other options if required, by using the command:

   ```
   set ddns [server=server] [port=port] [user=userid]
       [password=password] [dynamichost=hostnames]
       [customhost=hostnames] [statichost=hostnames]
       [primaryint=interface] [secondaryint=interface]
       [wildcard={on|off|nochg}]
       [mailexchanger={mailexchanger|nochg}]
       [backmx={yes|no|nochg}] [offline={yes|no}]
   ```

4. Check that the client is correctly sending updates, by using the command:

   ```
   show ddns
   ```

For a detailed step-by-step example and troubleshooting tips, see *How To Use Dynamic DNS To Allow You To Host Servers Behind A Dynamically-Assigned Public IP Address*. This How To Note is available in the Resource Center of the Documentation and Tools CDROM, or from www.alliedtelesis.co.uk/site/solutions/techdocs.asp?area=howto.

# Named Hosts

An important function of the IP module is to provide access to Telnet services. Normally such services are accessed by specifying the IP address of the full domain name of the service provider in the **telnet** command on page 61-34 of Chapter 61, Terminal Server:

```
telnet payroll.admin.thecompany.com
telnet 172.16.8.5
```

A single router may provide access for users to many services. To make access to these services easier for users, the IP module provides a host *nickname* table that maps an IP address or a full domain name to a short, easy to remember nickname. To add entries to the host name table, use the **add ip host** command on page 22-79.

For example, to add the nickname "payroll" for the IP host with IP address 172.16.8.5 and domain name "payroll.thecompany.com", use the command:

```
add ip host=payroll ipaddress=172.16.8.5
```

To modify an entry, use the **set ip host** command on page 22-154.

To delete an entry altogether, use the **delete ip host** command on page 22-100.

# Traffic Filters

Filters provide a mechanism for determining whether to process IP packets received over network interfaces. When an IP packet matches one of the patterns in a filter, the filter determines whether the packet is discarded or passed to the IP routing module.

Filtering is configured on a per-interface basis for packets received over the interface. Filtering decisions can be based on combinations of source address, destination address, TCP port, and protocol.

A *filter* is a list of *patterns*. A *pattern* consists of the following:

- a half pattern used to compare against the source address and port of an IP packet.
- a half pattern used to compare against the destination address and port of an IP packet.
- a protocol used to compare against the protocol of an IP packet.
- an ICMP message type used to compare against the type field of an ICMP packet.
- a flag used to compare against the presence or absence of an IP options field in an IP packet header.
- a maximum reassembly size used to compare against the reassembled packet size for IP fragments.
- a flag used to compare against the initiating end of a TCP session.
- an action, either *inclusion* or *exclusion*. Inclusion is the action of allowing the IP packet to be processed further and forwarded. Exclusion is the action of discarding the IP packet.

The filter is terminated by an implicit *match all* pattern, with an exclusion action. This pattern cannot be removed and does not appear in any displays.

A *half pattern* is a combination of an *IP address*, *network mask* and *port*. The IP address and network mask are represented in dotted decimal notation. The port is a TCP or UDP port number.

A specific half pattern matches exactly one address and port combination. A general half pattern matches a range of addresses and/or ports. When two specific half patterns are combined, the resulting specific pattern matches exactly one connection between two specific address/port pairs. Any other combination of specific and/or general half patterns produces a general pattern matching more than one address/port pair.

A linear search is performed on the filter. Searching stops at the first match found, so the order of patterns is important. Specific patterns always appear before general patterns. Within the specific patterns the order of patterns does not affect filter results since each pattern matches a specific and exclusive case. Within the general patterns, the order of patterns affects filter results since each pattern matches a range of address/port combinations that may overlap with another pattern.

For example, if the aim of the filter is to include all connections from a particular network except for a small range of addresses (e.g. a particular subnet), the exclusion pattern for the subnet must appear before the inclusion pattern for the network. Otherwise, packets from the subnet are included (and forwarded for processing) by the inclusion pattern without being compared against the exclusion pattern.

Regardless of whether a pattern is specific or general, its position in the filter effects the efficiency of the filter. Patterns that match the most commonly expected conditions should appear ahead of patterns matching less common conditions. This reduces the number of comparisons required to get a match.

To add an entry to a filter, use the **add ip filter** command on page 22-70. The **sport**, **dport**, **icmpcode**, and **icmptype** parameters can be a decimal number or one of a list of predefined names. The **log** parameter determines whether matches to a filter entry result in a message being sent to the router’s Logging facility, and the content of the log messages.

To modify an entry in a filter, use the **set ip filter** command on page 22-150.

To delete an entry in a filter, use the **delete ip filter** command on page 22-98.

To display filters and patterns currently defined and the number of matches, use the **show ip filter** command on page 22-196.

For overall efficiency, most traffic received by the router should be forwarded. The router should not be filtering out most of the traffic it receives. The efficiency of the filtering process can be maximised by careful ordering of all filters, including general filters, to reduce the number of comparisons required for the majority of IP packets. The counts of matches displayed in the output of the **show ip filter** command on page 22-196 can aid in determining the most efficient ordering of patterns within filters.

Defining a filter does not automatically enable it. To enable it, you must assign it to a network interface on the router with the **add ip interface** command on page 22-80.

To change the filter used on an interface, use the **set ip interface** command on page 22-155.

To display information about the interfaces assigned to the IP module, including the associated filter (if any) for each interface, use the **show ip interface** command on page 22-202.

A traffic filter, policy filter, and priority filter can be assigned to an interface. Traffic filters are configured on receiving interfaces and are applied to packets as they are received (packets are checked for a match to a filter entry). Traffic filters either discard packets or allow them into the router for processing and forwarding.

Priority filters are configured on transmitting interfaces and are applied to packets as they are queued for transmission (packets are checked for a match to a filter entry). Priority filters are applied to packets that have been received, processed, and assigned to an interface for transmission. Priority filters determine the order in which packets are sent.

Policy filters are configured on receiving interfaces and are applied to packets as they are received (packets are checked for a match to a filter entry). The policy filter's effect, however, is seen in the way filtered packets are transmitted. After packets have passed any traffic filters, the policy filter preferentially selects a route from the route table on which to forward the packet.

An interface may have a maximum of one traffic filter, one policy filter and one priority filter, but the same traffic, policy or priority filter can be assigned to more than one interface.

# SNMP

SNMP (Simple Network Management Protocol) is defined in RFCs 1155–1157, 1213, 1351, and 1352. The router's implementation of SNMP is based on RFC 1157, *A Simple Network Management Protocol (SNMP)*, and RFC 1812, *Requirements for IP Version 4 Routers*. SNMP provides a mechanism for management entities, or stations, to extract information from the *Management Information Base* (MIB) of a managed device.

SNMP can use a number of different protocols as its underlying transport mechanism, but the most common transport protocol, and the only one supported by the router, is UDP. Therefore the IP module must be enabled and properly configured in order to use SNMP. SNMP *trap* messages are sent to UDP port 162; all other SNMP messages are sent to UDP port 161. The router's SNMP agent accepts SNMP messages up to the maximum UDP length the router can receive.

The router implements an enterprise MIB (enterprise number 293), and a number of other standard MIBs including MIB-II (RFC 1213), Frame Relay DTE MIB (RFC 1315), Ethernet-like Interface Types MIB (RFC 1398), Bridge MIB (RFC 1493) and the Host Resources MIB (RFC 1514). See Chapter 55, Simple Network Management Protocol (SNMP) for a detailed description of the router's SNMP agent and the commands required to configure SNMP on the router. See Appendix C, SNMP MIBs for a detailed description of the MIBs and objects supported by the router's SNMP agent.

The router's standard **set** and **show** commands can also be used to access objects in the MIBs supported by the router.

# Control and Debug Commands

Several commands control the overall operation of the IP module. The IP module is disabled by default. To enable the IP module, use the **enable ip** command on page 22-122. To disable it, use the **disable ip** command on page 22-109.

All setup information is retained if the module is shut down. It is not necessary to enter new setup information after turning on the module.

The IP module operates in one of two modes, server mode or forwarding mode. In server mode the router does not route IP packets, but provides Telnet services, responds to SNMP requests, and uses TFTP to download software upgrades. In forwarding mode, the router routes IP packets, as well as performing all the functions of server mode. The default operational mode is forwarding. The operational mode is set with the **enable ip forwarding** command on page 22-126 and the **disable ip forwarding** command on page 22-113.

The current operational mode is retained when the IP module is disabled, and restored when the IP module is re-enabled. To display a snapshot of the current state of the IP module, use the **show ip** command on page 22-178.

The router stores all setup information (such as IP addresses) in non-volatile memory. To purge information stored in the IP module, use the **purge ip** command on page 22-137.

To enable the IP debugging facility, use the **enable ip debug** command on page 22-123. To disable it, use the **disable ip debug** command on page 22-110.

To display information about active TCP sessions, including the state and port number, use the **show tcp** command on page 22-226.

If a TCP connection is specified, detailed debugging information for that connection is displayed. To display similar information for active UDP sessions, use the **show ip udp** command on page 22-222.

# Ping and Trace Route

The Ping (*Packet Internet Groper*) and trace route functions verify connections between networks and network devices.

## Ping

Ping uses ICMP *Echo Request* messages to test the connectivity between two network devices running the Internet Protocol (IP).

The router's extended **ping** command on page 22-135 also supports IPv6, OSI, IPX, and AppleTalk protocols, using the protocol's native echo request message.

Echo request packets are sent to the destination addresses and responses are recorded. To initiate the transmission of ping packets, use a **ping** command on page 22-135.

Any parameters not specified use the defaults configured with a previous invocation of the **set ping** command on page 22-171.

As each response packet is received, a message is displayed on the terminal device from which the command was entered and details are recorded. To display default configuration and summary information, use the **show ping** command on page 22-224.

To halt a ping in progress, use the **stop ping** command on page 22-233.

If you can ping the end destination, then the physical and Layer 2 links are functioning, and any difficulties are in the network or higher layers.

If pinging the end destination fails, check your routes and ping intermediate network addresses. If you can successfully ping some network addresses, and not others, you can deduce which link in the network is down.

Note that if Network Address Translation (NAT) is configured on the remote router, pinging devices connected to it may give misleading information.

## Trace Route

You can use trace route to discover the route that packets pass between two systems running the IP protocol. Trace route sends an initial UDP packets with the Time To Live (TTL) field in the IP header set starting at 1. The TTL field is increased by one for every subsequent packet sent until the destination is reached. Each hop along the path between two systems responds with a TTL exceeded packet and from this the path is determined.

To initiate a trace route, use the command **trace** command on page 22-234. Any parameters not specified use the defaults configured with a previous invocation of the **set trace** command on page 22-174.

As each response packet is received a message is displayed on the terminal device from which the command was entered and the details are recorded. To display the default configuration and summary information, enter the command:

```
show trace
```

To halt a trace route that is in progress, enter the command:

```
stop trace
```

# Finger

The finger user information protocol provides a mechanism for exchanging user information between a finger client and a finger server.

A finger client is used to query a finger server for information about a specific user, or to request a list of all logged in users on the server. The information returned depends on the implementation of the finger server. However, in most cases the information returned includes the user's login name, real name, home directory, shell type, login details, and mail status. Other information may also be returned, and signature files may also be appended to the reply. To send a finger query to the finger server on a specific host, use the **finger** command on page 22-134.

The response from the finger server is sent to the terminal or telnet session from which the command was entered.

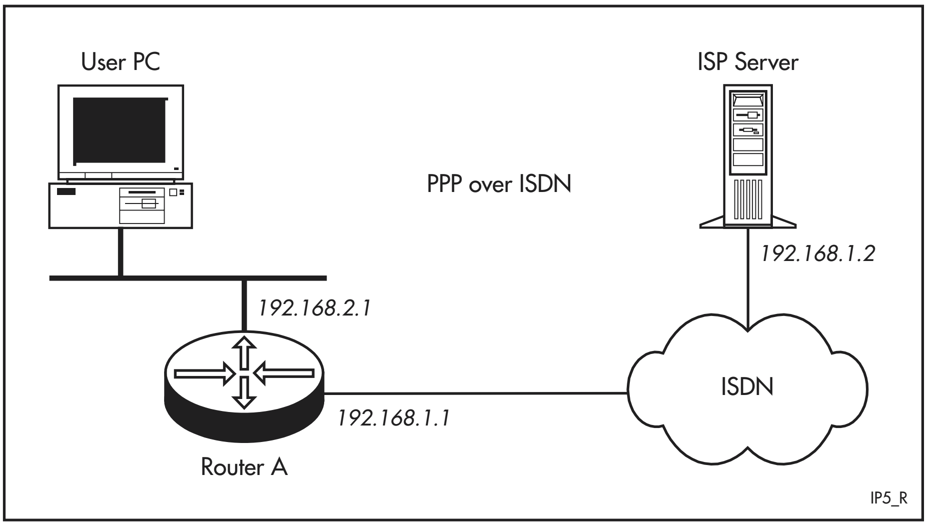
A finger server may also be configured so that when a query is received from a remote client, the finger query initiates a script file on the server that can be set to run system commands. While this opens a significant hole in system security, it makes it possible to control a remote host from anywhere within a network without having to log in to the host. IP filtering and careful selection of the commands that may be run via this method should provide basic security, although users should enable this function only when they are aware of the security implications.

**Example** This example shows how to use the finger client to obtain new mail from a local ISP whenever a link to that ISP is brought up.

In this example, the local ISP provider's mail host is running a finger server configured to respond to a finger query from a subscriber by downloading any new mail for the subscriber's account. Whenever the link to the ISP is brought up, the mail host is automatically polled to see if there is any new mail, without any user intervention.

The user's PC is connected to the LAN. A router on the LAN is used to access the ISP via a Basic Rate ISDN interface (BRI), which brings up the link on demand. On the router, a trigger is created that is activated when the ISDN call comes up and becomes active. The trigger runs a script that sends a finger query to the ISP with the username set to the account name of the subscriber (Figure 22-4 on page 22-42).

Figure 22-4: Using finger to trigger mail downloads from a mail host.



### To configure finger to trigger mail downloads from a mail host

1. **Configure the ISDN connection to the ISP.**

   Configure the Q.931 profile for the local territory:

   ```
   set q931=0 profile=nz
   ```

   Create an ISDN call, named "myisp", with ISDN number 123456:

   ```
   add isdn call=myisp number=123456 prec=out outsub=local
      searchsub=local
   ```

   Create a PPP interface to use the ISDN call, and enable the idle timer to make the link a dial-on-demand connection with a timeout period of 60 seconds (the default):

   ```
   create ppp=0 over=isdn-myisp
   
   set ppp=0 idle=on
   ```

2. **Configure IP.**

   Enable IP and add IP interfaces for the Ethernet LAN and the PPP connection to the ISP:

   ```
   enable ip
   
   add ip interface=ppp0 IP=192.168.1.1
   
   add ip interface=eth0 ip=192.168.2.1
   ```

3. **Create a trigger to send the finger query.**

   Create a script file to send the finger command to the ISP:

   ```
   add script=finger.scp text="finger
      myaccountname@192.168.1.2"
   ```

   Create the trigger to activate this script:

   ```
   create trigger=1 interface=ppp0 event=up script=finger.scp
   ```

   When the PPP link is brought up by a subscriber trying to access the ISP from anywhere on the LAN, the router sends a finger query to the ISP, which causes new mail on the server to return to the subscriber.

# Security Options

The IP module provides a number of features for securing networks in addition to those that IP traffic filters (see "Traffic Filters" on page 22-36) provide and to access restrictions to the router's SNMP agent (see "SNMP" on page 22-38).

To enable the router to check for and discard any packets with spoofed IP addresses, use the **enable ip spoofcheck** command on page 22-132. To disable it, use the **disable ip spoofcheck** command on page 22-119.

By default, spoof checking is enabled. The router determines that a packet has a spoofed IP address if the source address matches the address of a local interface. These packets usually indicate that either a device on the network is misconfigured, or that a malicious device has launched an attack. IP spoof checking only applies to packets routed by the CPU.

To enable source routing of IP packets, use the **enable ip srcroute** command on page 22-132. To disable it, use the **disable ip srcroute** command on page 22-119.

By default, source routing is disabled. Source routing is rarely used for legitimate purposes and is commonly used to circumvent packet-filtering firewalls.

To enable filtering of IP packets with a small fragment offsets or overlapping fragments, use the **enable ip fofilter** command on page 22-125. To disable it, use the **disable ip fofilter** command on page 22-112.

Attacks using tiny or overlapping fragments are designed to foil security schemes based on packet filtering mechanisms. Tiny fragments are too small to contain the full TCP header, making filter pattern matching difficult, while overlapping fragments can be used to 'replace' portions of preceding valid fragments with data that would otherwise be considered 'invalid'.

# Security Associations

A Security Association (SA) defines a security transform to be applied to all traffic between any of its members. A security association exists on one or more routers and defines a Virtual Private Network (VPN). A Security Association is identified by its Security Parameters Index (SPI), which must be the same on all instances of the security association throughout the VPN.

The members of an IP security association are IP addresses or contiguous groups of IP addresses. A member is defined by a base IP address and a network mask. A member is local to a router when it is separated from the Internet by the router, otherwise it is remote. All members of a security association are local to one router and remote to the other routers in the VPN.

The IP routing module uses security associations to implement IP payload encryption. AT-VPNet uses hardware encryption resources, security associations and IP payload encryption to create secure virtual private networks across the Internet.

To create a Security Association, use the **create sa** command on page 48-70 of Chapter 48, IP Security (IPsec).

To add members to the security association, specify the member's IP addresses with the **add sa member** command on page 48-48 of Chapter 48, IP Security (IPsec),

The **mask** parameter can be used to add a contiguous range of IP addresses as members of the security association. See the **create sa** and **add sa member** commands in Chapter 48, IP Security (IPsec).

To assign an IP interface with zero or more security associations, use the **add ip sa** command on page 22-95.

The IP SA commands provide support for RFCs 1825, 1827, and 1829, which have been superseded by IP Security. See Chapter 48, IP Security (IPsec) and RFCs 240–2412 for more information about IPsec.

To remove a security association from an IP interface, use the **delete ip sa** command on page 22-107.

To display a list of security associations currently assigned to an IP interface, use the **show ip sa** command on page 22-221.

If an IP interface does not have an assigned security association, all IP packets transiting the interface are processed normally by the IP routing module. If an IP interface uses one or more security associations it passes all packets transiting the interface to the SA module. The SA module examines the source and destination addresses of the packet to determine if the packet is in any of the security associations used by the interface. If a match is found, the SA module passes the packet to the ENCO module on the channel to which the security association is attached. In this way the configured transform is applied to the packet. If a packet is not in one of the security associations, it is discarded or sent through the interface depending on the setting of the IP interface's **samode** parameter. To set this parameter, use either the **add ip interface** command on page 22-80 or the **set ip interface** command on page 22-155.

IP datagrams discarded by a security association are logged by the Logging facility with a message type/subtype of IPFIL/SA.

# Broadcast Forwarding

The broadcast forwarding facility provides a mechanism for redirecting UDP broadcast packets to other hosts, routers or networks in an internet. A typical example would be the redirection of NETBIOS broadcasts between a Windows NT server on a central head office LAN and Windows NT workstations attached to remote LANs. NETBIOS is just one of a number of UDP broadcast packets that may require forwarding. Others include TFTP, DNS, BOOTP and Time. BOOTP forwarding, defined in RFC 1542, is a special case and the router handles it separately.

Broadcast forwarding is configured by defining, for each interface, a list of one or more UDP ports to listen on, and the destination IP addresses to which any UDP broadcasts are to be forwarded. By default, broadcast forwarding is disabled. When broadcast forwarding is enabled and configured, UDP listen ports are opened for each of the UDP ports on which UDP broadcasts are to be forwarded. When a UDP broadcast packet is received on an interface for one of the configured ports, it is forwarded to each of the destinations listed for that interface.

## Example: Forwarding to a unicast address

In this example, a number of Windows NT workstations on a remote office LAN are attached to a single Windows NT server on the head office LAN (Figure 22-5, Table 22-3). Since all broadcasts originate from or are intended for a single NT server on the head office LAN, the destination address for the NETBIOS port is set to the NT server's unicast IP address. In this case, broadcast forwarding needs to be configured on the remote router. This method may become administratively difficult when many destinations on the same network must be specified.

Figure 22-5: Example configuration for broadcast forwarding to a unicast address

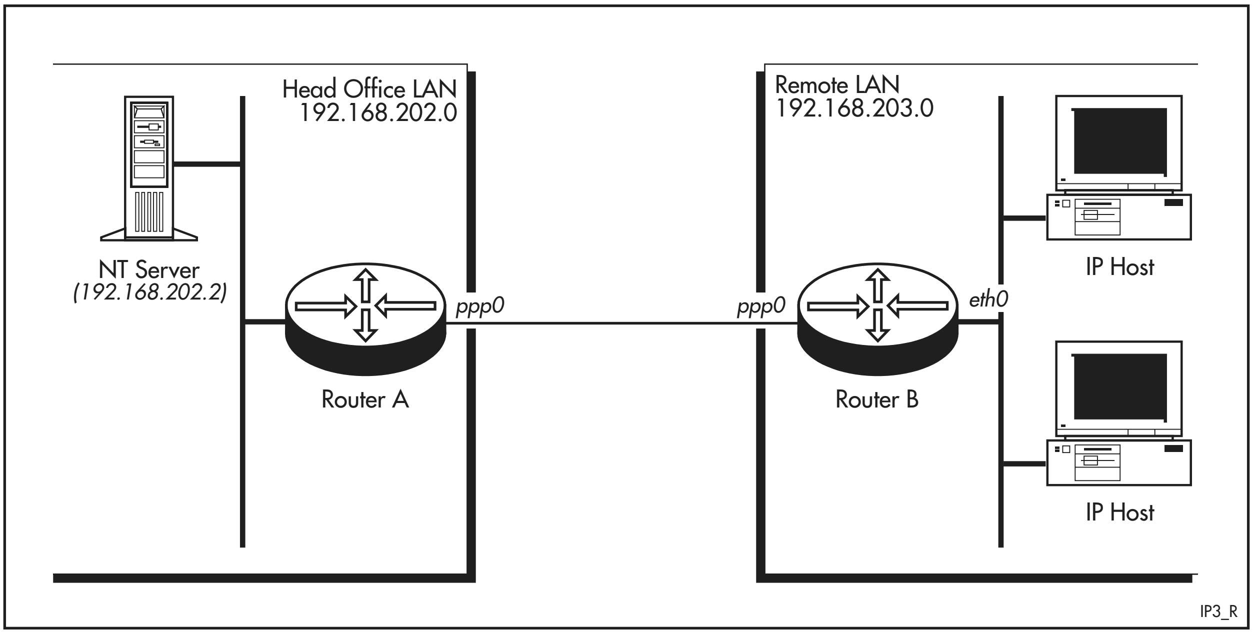


Table 22-3: Example configuration parameters for broadcast forwarding to a unicast address

| Parameter | Head Office | Remote Office |
|---|---|---|
| Router Name | A | B |
| IP address of LAN | 192.168.202.0 | 192.168.203.0 |
| Ethernet interface | eth0 | eth0 |
| PPP (WAN link) interface | ppp0 | ppp0 |
| IP address of NT Server | 192.168.202.2 | - |
| UDP protocol to forward | NETBIOS | - |

### To configure broadcast forwarding to a unicast address

1. **Enable broadcast forwarding.**

```
enable ip
enable ip helper
```

2. **Configure the UDP protocols to be forwarded.**

All NETBIOS broadcasts received by Router B's Ethernet interface are to be forwarded to the NT server with IP address 192.168.202.2.

```
add ip interface=eth0 ipaddress=192.168.203.1
add ip helper port=netbios destination=192.168.202.2
    interface=eth0
```

## Example: Forwarding to a broadcast address

In this example, a number of Windows NT workstations on a remote office LAN are attached to several Windows NT servers on the head office LAN (Figure 22-6, Table 22-4). Since broadcasts originate from or are intended for several NT servers on the head office LAN, the destination address for the specified port is set to the subnet broadcast address for the head office LAN. In this case, broadcast forwarding must be configured on both the remote router and the head office router. When the head office router receives the UDP packet it re-broadcasts the packet on to the remote LAN.

There are two consequences to consider with this method: 1) broadcast traffic increases on the head office LAN, and 2) you can create broadcast loops if not carefully configured. The broadcast forwarding facility in this mode is acting like a pseudo-bridge, but without the protection of protocols such as Spanning Tree to detect loops.

Figure 22-6: Example configuration for broadcast forwarding to a multicast address

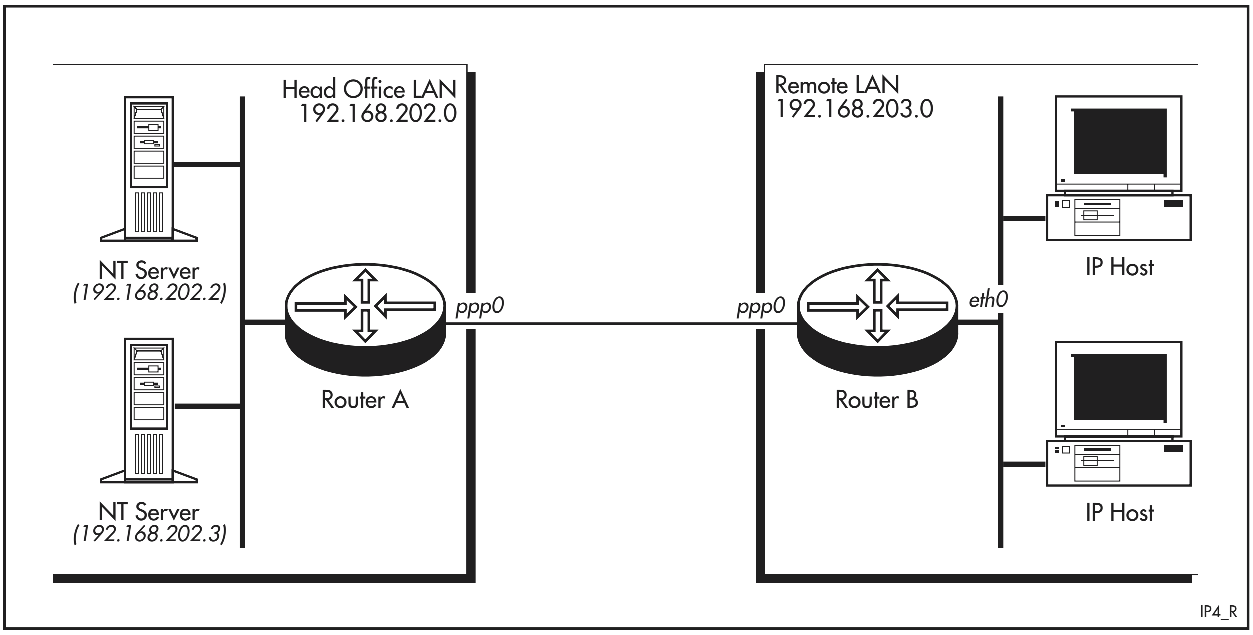


Table 22-4: Example configuration parameters for broadcast forwarding to a multicast address

| Parameter | Head Office | Remote Office |
|---|---|---|
| Router Name | A | B |
| IP address of LAN | 192.168.202.0 | 192.168.203.0 |
| Ethernet interface | eth0 | eth0 |
| PPP (WAN link) interface | ppp0 | ppp0 |
| IP address of NT Servers | 192.168.202.2<br>192.168.202.3 | - |
| UDP protocol to forward | NETBIOS | - |

### To configure broadcast forwarding to a broadcast address

1. **Enable broadcast forwarding.**

   Enable broadcast forwarding on Router A and Router B, using the following command on each router:

   ```
   enable ip
   enable ip helper
   ```

2. **Configure the UDP protocols to be forwarded.**

   All NETBIOS broadcasts received by the remote router's Ethernet interface are to be forwarded to the head office LAN with IP address 192.168.202.0. On Router B, use the command:

   ```
   add ip interface=eth0 ipaddress=192.168.203.2

   add ip helper port=netbios destination=192.168.202.255
      interface=eth0
   ```

   All NETBIOS broadcasts received by the head office router's PPP interface ppp0 are to be broadcast on to the head office LAN. On Router A, use the command:

   ```
   create ppp=0 over=syn0

   add ip interface=ppp0 ipaddress=0.0.0.0

   add ip helper port=netbios destination=192.168.202.255
      interface=PPP0
   ```

# IP Multicasting

IP multicasting, defined in RFC 1112, *Host Extensions for IP Multicasting,* and RFC 1812, *Requirements for IP version 4 Routers*, is the process of transmitting an IP datagram to a group of hosts. A *host group* may contain zero or more hosts. A multicast datagram is delivered to each member of the group as if the datagram had been sent individually to each host as a unicast datagram.

A single IP address identifies a host group. IP addresses from 224.0.0.0 to 239.255.255.255 are reserved for use as multicast addresses, and each address identifies a host group. The IP address 224.0.0.0 is guaranteed not to be assigned to any host group. The IP address 224.0.0.1 is assigned to the permanent group of all IP hosts and gateways, and is used to address all multicast hosts on the directly connected network. There is no multicast IP address for all hosts on the Internet.

Host groups are dynamic – hosts can join or leave host groups at any time. Any host on the Internet can be a member of any host group, and can be a member of any number of groups at the same time. A host does not need to be a member of a host group to send a multicast datagram to the group.

A multicast datagram can be transmitted to both the local network and all remote networks that are reachable within the IP TTL (time-to-live) value for the datagram. To send an IP multicast datagram, a host transmits the datagram as a local multicast datagram to all members of the host group on the directly connected network. Multicast routers on the local network forward the multicast datagram to all other networks with members in the host group. On the remote destination network, the local multicast router transmits the datagram as a local multicast onto the directly connected network.

Multicasting can be performed dynamically using DVMRP, PIM Sparse Mode or PIM Dense Mode, or statically. Both dynamic and static multicasting use IGMP to manage group membership. IGMP, DVMRP, PIM Sparse Mode and PIM Dense Mode are described in Chapter 24, IP Multicasting.

## Static Multicast Forwarding

If neither DVMRP nor PIM is enabled for full dynamic multicasting, IP interfaces can be statically set to receive or send all multicast packets. The default is for all interfaces to receive all multicast packets, but not to forward any. If either DVMRP or PIM are enabled, they dynamically determine the forwarding behaviour of interfaces, and the static multicast setting of an interface is ignored (Chapter 22, IP Multicasting).

The router can be configured to send and receive multicast datagrams; to send only or receive only; or to neither send nor receive them. To configure static IP multicasting when the IP interface is created, use the command:

```
add ip interface=interface ipaddress={ipadd|dhcp}
    [multicast={off|send|receive|both|on}] [other-options]
```

To configure on an existing IP interface, use the command:

```
set ip interface=interface
    multicast={both|off|on|receive|send} [other-options]
```

To display the state of static IP multicasting and counts of multicast packets processed, use the **show ip interface** command on page 22-202.

Static multicast routing is configured on a per-interface basis. All logical IP interfaces on the same IP interface use the same multicast setting, so changing the setting for multicasting on one logical interface affects all other logical interfaces in the IP interface. For multicast datagrams being forwarded by the router, an IP interface with more than one logical interface forwards one multicast datagram out the interface. However, this does not necessarily apply to multicast packets originating from the router. For example, in the case of OSPF on a router with a number of logical interfaces, each Hello packet sent must be sent on a per logical interface basis, since the packet checking code at the destination checks the source address of the packet for a match.

# Network Address Translation (NAT)

Network Address Translation (NAT), defined in RFC 1631, provides a solution to one of the major problems facing the Internet – IP address depletion. NAT allows the reuse of IP addresses by taking advantage of the fact that a small percentage of hosts in a stub domain communicate outside the domain at any one time. A *stub* domain is a domain such as a corporate network that handles traffic originating from or destined for hosts in the domain. Many hosts never communicate outside of their stub domain. Only a subset of the IP addresses inside a stub domain needs to be translated into globally unique IP addresses when outside communication is required.

NAT lets IP addresses inside a stub domain be reused by other stub domains, meaning they need not be unique. For instance, a single Class A address could be used by a stub domain. RFC 1597 specifies a list of network addresses that are reserved for such a purpose. NAT is configured at each exit point router between a stub domain and the Internet backbone. When there is more than one exit point, each NAT entity must use the same translation table. The router at the exit point is configured with a group of IP addresses that are globally unique on the Internet. The source IP address of a packet received at the exit point is translated to use one of the globally unique addresses, and then forwarded to the Internet.

Enhanced NAT (ENAT) is a variation that uses a single global Internet IP address. Each new session crossing the exit point router is assigned a set of unique TCP/UDP port numbers. For example, consider a TELNET packet sent from a private network to a host on the Internet. The source IP address is changed to the single global IP address and the source TCP port number is substituted for one that is unique. The router maintains a list of current sessions using the IP source address, original source port, substituted port, destination port, and destination IP address information. The advantage of this scheme is that only a single IP address is required from an Internet Service Provider (ISP) to connect an entire private network. The private network can easily be shifted to another ISP simply by changing the one global IP address.

A disadvantage of NAT and ENAT is that both require intervention into the IP packets themselves. In particular, the source or destination address must be modified for each packet traversing the NAT entity. This means patching the IP checksum and also their pseudo checksums when TCP or UDP traffic is being forwarded. Some IP protocols embed IP information into the data stream that also requires patching. As a result, only payload encryption is possible on such links.

A major benefit of NAT and ENAT technologies is greatly enhanced security. Many firewalls use ENAT technology.

The term *private* network refers to stub domains because they are typically assigned IP addresses using RFC 1597. Traffic or routing information from these networks should never be sent directly to the Internet. Officially assigned IP addresses are globally unique. The terms *global interface* and *global IP address* specify the interface and officially assigned IP address for that interface, which connects the private network to the Internet.

The router supports the following NAT methodologies.

| Method | Description |
|---|---|
| Static NAT | One fixed IP address is translated to one fixed global address. The benefit of this scheme is that all the hosts on a private network can be numbered using RFC 1597, but certain servers' IP addresses can be statically translated to external global IP addresses, when a session traverses the router running NAT. All possible TCP/UDP and ICMP sessions and flows to the specified host can originate from the global Internet. |
| Dynamic NAT | Any host from a specified range of hosts can use a pool of global IP addresses to connect to the global Internet. Addresses are allocated on a first come first served basis. When no sessions or flows for a particular private IP-to-global-IP translation remain, the global IP address is returned to the pool for re-use. |
| Dynamic ENAT | A single global Internet IP address is required. Both the private IP address and protocol dependent port numbers are translated. The router maintains a list of all currently used port numbers and allocates a unique number for each new session or flow being translated. Sessions and flows can originate from the private network. |

| **Method** | **Description** (cont.) |
|---|---|
| Static ENAT | A private IP address and port number are mapped to a global IP address and port number. This method allows a server on the private network to be made available to the global Internet. The server still remains very secure since sessions or flows to the specified port can originate from the global Internet. |
| Interface ENAT | A variation of dynamic ENAT in which the global IP address for translation is determined dynamically from a specified interface. This method allows a private network to use ENAT translation on a router with a PPP link to an ISP who dynamically issues global IP addresses. Sessions and flows can originate from the private network. |

By default, NAT is disabled on the router. NAT must be explicitly enabled with the **enable ip nat** command on page 22-130.

NAT is automatically disabled when the firewall is enabled because the firewall provides NAT services. However, the NAT configuration is retained so that it can be (manually) re-enabled if the firewall is disabled.

To add translations to the network address translation table, use the **add ip nat** command on page 22-87.

The state of TCP sessions using the NAT gateway is maintained. The router monitors the flag bits of TCP packets passing through the NAT gateway. When the NAT gateway detects a TCP session has closed, the entry is deleted after a timeout period.

Handling UDP packets is based on the concept of a *flow*. Although UDP is a connectionless protocol, typically when two hosts communicate using UDP, there is an exchange of information between UDP ports on different hosts. This traffic can be thought of as a flow because the port numbers do not change during this exchange of information. When a flow starts, a timer is started. Each received packet resets the timer. If the timer expires, the flow entry is deleted.

The method for handling ICMP messages depends on the NAT method that is used. With static or dynamic NAT, IP addresses in the messages are translated as expected. With ENAT, using a single global IP address requires that ICMP messages be handled specially. ICMP messages can be categorised as follows:

- ICMP Destination Unreachable, Time Exceeded, Parameter Problem, and Source Quench packets sent to the global IP address are handled by matching the returned 64 bytes of header to a current session or flow. If no match is found the message is silently discarded. Outbound ICMP messages are fixed as normal with the source IP address being replaced with the global address.
- ICMP Redirect messages are ignored as they have no meaning across a NAT gateway.
- ICMP echo requests and Timestamps are handled slightly differently. An entry is created for the packet and a timer started. The sequence number is replaced with a unique port number, the original sequence number stored, and the fixed up packet forwarded on. When a reply is returned it is matched, fixed up and returned to the sender. The entry is then discarded. If no reply is received then the entry is deleted when the timer expires.

The global IP address for a PPP interface can be assigned dynamically. This is the case for a router connected to an ISP who dynamically assigns IP addresses.

If an interface ENAT has been created over the PPP interface, NAT monitors all PPP interface state changes. If the global IP address associated with a PPP interface is marked as “dynamically set”, then if the PPP interface is down and a packet is ready to be transmitted, NAT calls PPP to bring up the interface and queue the packet for later handling. When the PPP interface comes up, NAT determines the current IP address for the global interface, and based on that, forwards the stored packets. Note, if the global interface is marked as “dynamically set” and the interface goes down, then all current sessions or flows are deleted as they are no longer valid.

In this implementation all received traffic on the global interface is dropped (and possibly logged) unless either the connection or flow was initiated from the private network, or a specific configuration entry exists to explicitly allow the traffic.

## PPTP Pass Through

PPTP is the pre-defined service name you can specify when adding a local private IP network to the address translation table that NAT uses. The PPTP value is 1723.

To specify PPTP, use the command:

```
add ip nat ip=ipadd gblport=pptp port=pptp protocol=tcp
    [mask=ipadd] [gblip=ipadd] [gblmask=ipadd]
    [gblinterface=interface]
```

The **port** parameter specifies the port number or service name for the port used on the private IP host when specifying a static ENAT entry.

The **gblport** parameter specifies the port number or service name for the port available to global Internet access, when creating a static ENAT.

# Remote Address Assignment

The remote IP address assignment facility enables unnumbered PPP interfaces (such as PPP interfaces with an IP address of 0.0.0.0) to be dynamically assigned an IP address during the PPP link’s negotiation process.

If a PPP interface is created with an IP address of 0.0.0.0, and remote IP address assignment is enabled, during the IP control protocol (IPCP) negotiation process the router allows the remote PPP peer to set the IP address of the local PPP interface.

If the local PPP interface has an IP number other than 0.0.0.0, or if remote IP address assignment is disabled, the router does not allow the remote PPP peer to set the IP address of the local PPP interface.

To enable a remote IP address assignment, use the **enable ip remoteassign** command on page 22-131. To disable one, use the **disable ip remoteassign** command on page 22-118.

The current status of the remote IP assignment option is displayed in the output of the **show ip** command on page 22-178.

# IP Address Pools

An IP address pool is a named collection of IP addresses that ACC, PPP and other modules can use to assign IP addresses to dynamic connections. The advantage of an address pool is that a finite number of IP addresses can be re-used by many clients. When a client is finished with the IP address (for example, when a dial-in SLIP connection terminates) the IP address is returned to the pool and is available for another client to use.

The router supports multiple methods for assigning IP addresses to dynamic dial-in calls. The IP address assigned to a dial-in call is selected as follows:

1. If the user is authenticated via RADIUS, and the RADIUS response supplies an IP address, then that IP address is used.
2. If the user is authenticated using TACACS, then the domain name is appended to the login name and a Domain Name Service (DNS) request is issued to resolve the name to an IP address.
3. If the user is authenticated using TACACS and an ACC domain name has been specified with the **add acc domainname** command on page 19-20 of Chapter 19, Asynchronous Call Control then the domain name is appended to the login name and a Domain Name Service (DNS) request is issued to resolve the name to an IP address.
4. If the user is authenticated using TACACS and an ISDN domain name has been specified with the **add isdn domainname** command on page 11-70 of Chapter 11, Integrated Services Digital Network (ISDN), then the domain name is appended to the login name and a Domain Name Service (DNS) request is issued to resolve the name to an IP address.
5. If the user is authenticated via the router's internal User Authentication Database, and an IP address is set in the User Authentication Database for that user, then that IP address is used.
6. If the call is an ACC call on an asynchronous port with an IP address set, then that IP address is used.
7. If the ACC or PPP call has an IP pool set, and the request to the IP pool is successful, then that IP address is used.

To create an IP address pool, use the **create ip pool** command on page 22-96.

To destroy an IP address pool, use the **destroy ip pool** command on page 22-108.

To display the currently configured IP address pools, and the status of the IP addresses in the pools, use the **show ip pool** command on page 22-211.

To associate an IP address pool with an ACC call so that SLIP dial-in connections using that call uses IP addresses from the IP address pool, use either of the commands:

```
add acc call=call-name ippool=pool-name
    [other-acc-options...]
```

```
set acc call=call-name ippool=pool-name
    [other-acc-options...]
```

To disassociate an IP address pool from an ACC call so that dial-in connections using that call no longer use IP addresses from the IP address pool, use the command:

```
set acc call=call-name ippool=none
```

To associate an IP address pool with a PPP interface so that connections using that interface use IP addresses from the IP address pool, use either of the commands:

```
create ppp=ppp-interface over=physical-interface
    ippool=pool-name [other-ppp-options...]

set ppp=ppp-interface ippool=pool-name [other-ppp-options...]
```

To disassociate an IP address pool from a PPP interface so that connections using that interface no longer use IP addresses from the IP address pool, use the command:

```
set ppp=ppp-interface ippool=none
```

To associate an IP address pool with a PPP template so that dynamic PPP interfaces created using the PPP template use IP addresses from the IP address pool, use either of the commands:

```
create ppp template=template ippool=pool-name
    [other-template-options...]

set ppp template=template ippool=pool-name
    [other-template-options...]
```

To disassociate an IP address pool from a PPP template so that dynamic PPP interfaces created using the PPP template no longer use IP addresses from the IP address pool, use the command:

```
set ppp template=template ippool=none
```

# Configuration Examples

Examples in this section show the following configurations:

- Basic IP Setup over PPP
- Configuring IP Filters

For an example of a basic IP LAN, see "Configuration Example" on page 25-5 of Chapter 25, Routing Information Protocol (RIP).

## Basic IP Setup over PPP

This example connects two routers through a PPP connection and uses RIP as the routing protocol. The routers are connected to each other using the Point-to-Point Protocol (PPP) over a wide area data communications link. Each router has a single Ethernet LAN segment attached to it, as shown in the following figure.

Figure 22-7: Example configuration for a basic TCP/IP network

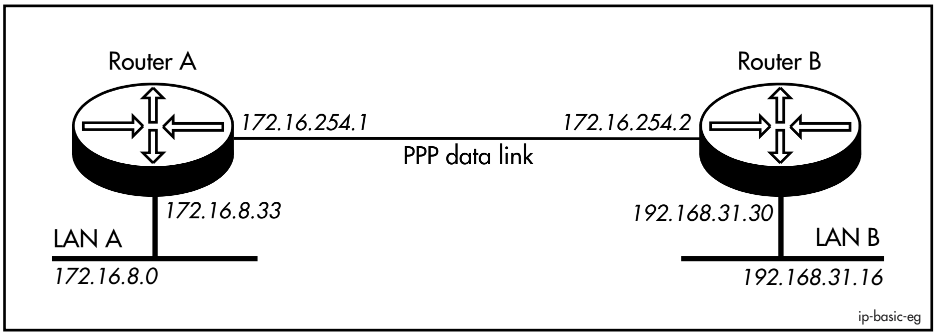


Table 22-5: Example configuration parameters for a basic TCP/IP network

| Parameter | Router A | Router B |
|---|---|---|
| LAN IP subnet address | 172.16.8.0 | 192.168.31.16 |
| LAN network class | B | C |
| LAN number of subnet bits | 8 | 4 |
| LAN IP network mask | 255.255.255.0 | 255.255.255.240 |
| Ethernet IP address | 172.16.8.33 | 192.168.31.30 |
| Synchronous port | 0 | 0 |
| PPP interface | 0 | 0 |
| PPP IP subnet address | 172.16.254.0 | 172.16.254.0 |
| PPP interface IP address | 172.16.254.1 | 172.16.254.2 |

### To configure a basic IP network

1. **Configure the PPP Link.**

   See "Configuring a PPP link" on page 15-34 of Chapter 15, Point-to-Point Protocol (PPP) for a step-by-step example of how to set up a PPP link. Use the following command on each router:

   ```
   create ppp=0 over=syn0
   ```

**2. If necessary, initialise and enable the IP routing module.**

If the router has previously been configured for IP, you may want to use the following commands on both routers to clear the IP configuration and enable the IP routing module:

```
purge ip
enable ip
```

The **purge ip** command disables the IP routing module, so you must explicitly enable the module.

**3. Add interfaces to the IP routing module.**

The interfaces must now be assigned to the IP routing module. Use the following commands on Router A:

```
add ip interface=eth0 ip=172.16.8.33 mask=255.255.255.0
add ip interface=ppp0 ip=172.16.254.1 mask=255.255.255.0
```

The IP routing module on Router B must now be configured, using a similar sequence of commands. The main difference is that Router B has a Class C network on the Ethernet interface. This requires a different network mask. Use the following commands for Router B:

```
add ip interface=eth0 ip=192.168.31.30
   mask=255.255.255.240
add ip interface=ppp0 ip=172.16.254.2 mask=255.255.255.0
```

The metrics for the interfaces defaults to 1. The IP module is now enabled, linked to the physical interfaces, and operational. By default, the router does not receive or transmit route information until it has been configured to use a routing protocol. For this example assume RIP is used.

**4. Configure RIP as the routing protocol.**

A routing protocol must now be enabled to allow the routers to communicate and to update the internal routing tables. This example uses RIP. This is to be broadcast onto the Ethernet LAN, but is to be directed explicitly to each end of the PPP link. For Router A, use the following commands:

```
add ip rip interface=eth0
add ip rip interface=ppp0
show ip rip
```

Specifying only the interface causes RIP to be broadcast to the whole network or subnet.

For Router B use:

```
add ip rip interface=eth0
add ip rip interface=ppp0
show ip rip
```

The router configuration is now complete.

### To test the configuration

1. **Check the routes.**

   Provided the interfaces are connected to other systems acting as routers, the router obtains IP routes after a short period (up to 60 seconds). These routes show the network from the point of view of the router. You can check the route table to verify the correct operation of the IP module, by using the following command on either router:

   **show ip route**

   The display (on Router A) should be similar to the following output.

```
IP Routes
--------------------------------------------------------------------------------
Destination        Mask                NextHop              Interface          Age
DLCI/Circ.         Type      Policy    Protocol             Metrics     Preference
--------------------------------------------------------------------------------
172.16.8.0         255.255.255.0       0.0.0.0              eth0              8372
-                  direct    0         static               1                  100
172.16.254.0       255.255.255.0       0.0.0.0              ppp0              8372
-                  direct    0         static               1                  100
192.168.31.16      255.255.255.240     172.16.254.2         ppp0              8369
-                  remote    0         rip                  2                  100
```

   The route table should contain easily verifiable data and should indicate that this router can communicate with other router systems. The **ping** command (common to most TCP/IP implementations) can be used on a host to test that paths to remote hosts are available through the router.

   The router’s **ping** command on page 22-135 can be used to verify that hosts respond on both links. Use the command:

   ping *ipadd*

   If *nickname* has been added to the host name table with the **add ip host** command on page 22-79, use the command:

   ping *nickname*

   ICMP echo request packets are sent to the host IP address and the response time for each is listed when the command is successful.

2. **Check the ARP cache.**

   The ARP cache starts to show binding information (especially from the LAN link) for each active host on the links. The ARP cache can be checked using the command:

   **show ip arp**

   The router should have entries for some known hosts in the ARP cache. This means that it communicates correctly with these hosts.

   Entries appear only in the ARP cache when a local host attempts to access a host on another subnet or when it uses a protocol like BOOTP. It is easy to force this by attempting to ping a host on another subnet from a local host.

3. **Try using Telnet to access the remote router.**

   To Telnet from Router A to Router B, on Router A use the command:

   `telnet 192.168.31.30`

   To Telnet from Router B to Router A, on Router B use the command:

   `telnet 172.16.8.33`

   You can use any of the assigned interface IP addresses as the target for Telnet access.

# Configuring IP Filters

This example shows how to configure IP filtering to perform firewall functions.

With the increase in connections to the Internet, and the interconnection of networks from different organisations, filtering data packets is an important mechanism to ensure that only legitimate connections are allowed. Security can never be perfect while connections to other networks exist, but filters allow network managers to manage the permissible free access, while restricting users without permission.

The router can restrict traffic on the basis of source/destination IP address, source/destination ports, IP protocol type and TCP flags. The choice of filters depends on an organisation's particular requirements. However, extensive filtering and large filter lists reduce the performance of the router, so filtering design needs to ensure that lists are simple, but effective.

In this example, an organisation wishes to allow access to its mainframe for users from another organisation. Access from the remote network is controlled by filters defined on the local router (Figure 22-8 on page 22-58). On the remote network there are three hosts. Host A can connect via Telnet to the mainframe. Host B can connect via Telnet and FTP. Host C can connect via FTP. Table 22-6 on page 22-58 lists parameter values used in the example. A static route exists for the PPP link between the local and remote routers.

Figure 22-8: Example configuration for IP filtering

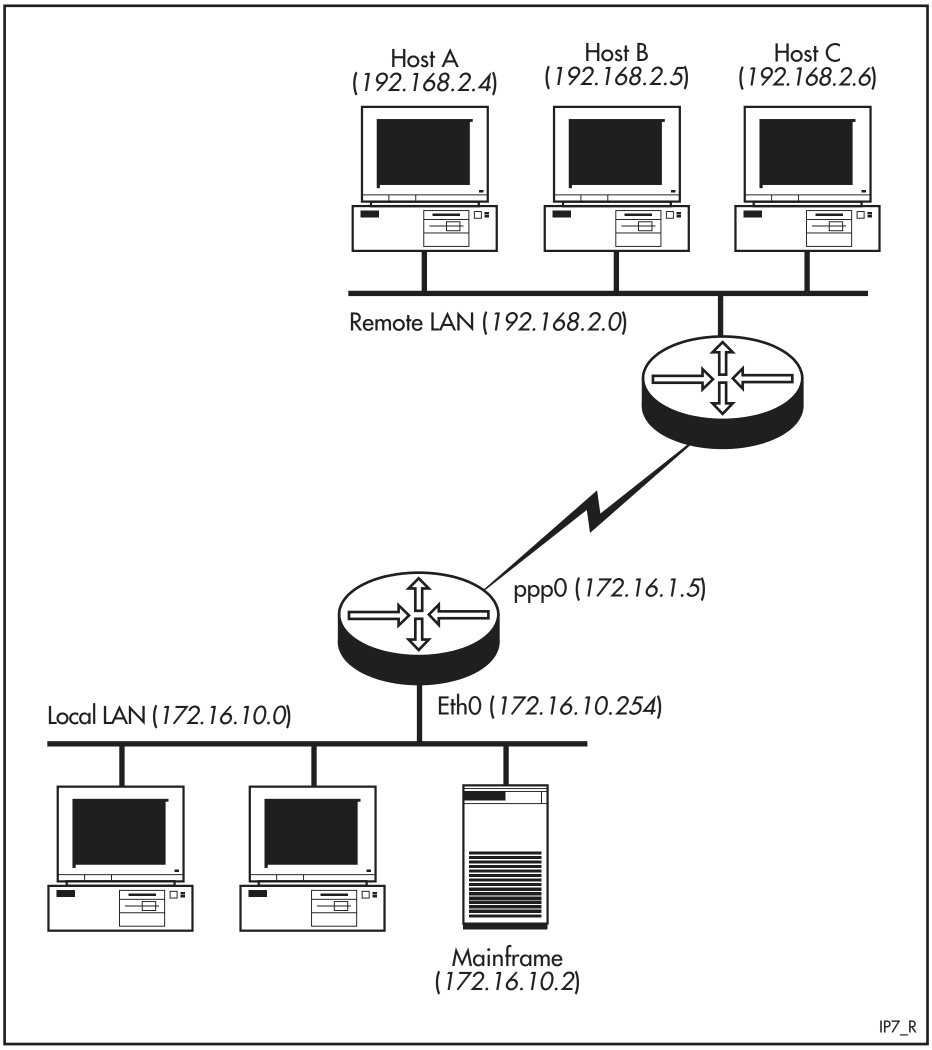


Table 22-6: Example configuration parameters for IP filtering

| **Site** | **Local LAN** | **Remote LAN** |
|---|---|---|
| LAN subnet | 172.16.10.0 | 192.168.2.0 |
| LAN network mask | 255.255.255.0 | 255.255.255.0 |
| Eth0 interface IP address | 172.16.10.254 | - |
| ppp0 interface IP address | 172.16.1.5 | - |
| ppp0 network mask | 255.255.255.0 | 255.255.255.0 |
| Mainframe IP address | 172.16.10.2 | - |
| Remote Host A IP address | - | 192.168.2.4 |
| Remote Host B IP address | - | 192.168.2.5 |
| Remote Host C IP address | - | 192.168.2.6 |

### To configure IP filters

1. **Create a filter to control the access of hosts A, B and C to the mainframe.**

   Create filter 1 for interface ppp0 to control the access of hosts A, B and C on the remote network to the mainframe on the local network. To enable Telnet connections from host A, use the command:

```
enable ip
```

```
add ip filter=1 type=traf so=192.168.2.4
   sm=255.255.255.255 destination=172.16.10.2
   dm=255.255.255.255 dport=telnet protocol=tcp sess=any
   action=include
```

   To enable Telnet and FTP access from host B, use the commands:

```
add ip filter=1 type=traf so=192.168.2.5
   sm=255.255.255.255 destination=172.16.10.2
   dm=255.255.255.255 dp=ftpdata protocol=tcp sess=esta
   action=include

add ip filter=1 type=traf so=192.168.2.5
   sm=255.255.255.255 destination=172.16.10.2
   dm=255.255.255.255 dp=ftp protocol=tcp sess=any
   action=include

add ip filter=1 type=traf so=192.168.2.5
   sm=255.255.255.255 destination=172.16.10.2
   dm=255.255.255.255 dp=telnet protocol=tcp sess=any
   action=include
```

   To enable FTP access from host C, use the commands:

```
add ip filter=1 type=traf so=192.168.2.6
   sm=255.255.255.255 destination=172.16.10.2
   dm=255.255.255.255 dp=ftp protocol=tcp sess=esta
   action=include

add ip filter=1 type=traf so=192.168.2.6
   sm=255.255.255.255 destination=172.16.10.2
   dm=255.255.255.255 dp=ftpdata protocol=tcp sess=esta
   action=include
```

   The last entry in a filter is always an implicit entry (one that you need not enter) to exclude all sources, destinations, and ports. It is equivalent to the command:

```
add ip filter=1 type=traf so=0.0.0.0 smask=0.0.0.0
   destination=0.0.0.0. dmask=0.0.0.0 sport=all
   action=exclude
```

2. **Create a filter to allow replies from the mainframe to reach hosts A, B and C.**

   Create filter 2 for interface eth0 to allow the replies from the mainframe to remote hosts A, B and C, but prevent other users on the local network from accessing remote hosts A, B and C:

```
add ip filter=2 type=traf so=172.16.10.2
   sm=255.255.255.255 sp=telnet destination=192.168.2.4
   dm=255.255.255.255 protocol=tcp sess=esta
   action=include

add ip filter=2 type=traf so=172.16.10.2
   sm=255.255.255.255 sp=telnet destination=192.168.2.5
   dm=255.255.255.255 protocol=tcp sess=esta
   action=include

add ip filter=2 type=traf so=172.16.10.2
   sm=255.255.255.255 sp=ftpdata destination=192.168.2.5
   dm=255.255.255.255 protocol=tcp sess=any
   action=include

add ip filter=2 type=traf so=172.16.10.2
   sm=255.255.255.255 sp=ftp destination=192.168.2.5
   dm=255.255.255.255 protocol=tcp sess=esta
   action=include

add ip filter=2 type=traf so=172.16.10.2
   sm=255.255.255.255 sp=ftpdata destination=192.168.2.65
   dm=255.255.255.255 protocol=tcp sess=any
   action=include

add ip filter=2 type=traf so=172.16.10.2
   sm=255.255.255.255 sp=ftp destination=192.168.2.6
   dm=255.255.255.255 protocol=tcp sess=esta
   action=include
```

   The explicit exclusion is not required. Other hosts on the local network are not able to communicate with hosts on the remote network.

3. **Add the filters to the interfaces.**

   The filters that have been defined must be assigned to interfaces in order for them to take affect. Assign filter 1 to interface ppp0 and filter 2 to interface eth0, using the commands:

```
create ppp=0 over=syn0

add ip interface=ppp0 ip=172.16.10.54 mask=255.255.255.0
   filter=1

add ip interface=eth0 ip=172.16.1.5 mask=255.255.255.0
   filter=2
```

4. **Test the configuration.**

   The definitions of the filters can be checked with the command:

```
show ip filter
```

   This command produces output similar to Figure 22-9 on page 22-61.

   To display details of the IP interfaces defined, including the filter assigned to each interface (Figure 22-10 on page 22-62), use the command:

```
show ip interface
```

Figure 22-9: Example output from the **show ip filter** command for IP filtering

```
IP Filters
--------------------------------------------------------------------------------
No.  Filter Type
  Ent. Source Port   Source Address    Source Mask       Session         Size
       Dest. Port    Dest. Address     Dest. Mask        Prot.(C/T)      Options
       Pattern Type  Act/Pol/Pri                         Logging         Matches
--------------------------------------------------------------------------------
 1   Traffic
    1  Any           192.168.2.4       255.255.255.255   Start           Any
       23:23         172.16.10.2       255.255.255.255   TCP             Any
       General       Include                             Off             0
    2  Any           192.168.2.5       255.255.255.255   Established     Any
       20:20         172.16.10.2       255.255.255.255   TCP             Any
       General       Include                             Off             0
    3  Any           192.168.2.5       255.255.255.255   Any             Any
       21:21         172.16.10.2       255.255.255.255   TCP             Any
       General       Include                             Off             0
    4  Any           192.168.2.5       255.255.255.255   Start           Any
       23:23         172.16.10.2       255.255.255.255   TCP             Any
       General       Include                             Off             0
    5  Any           192.168.2.6       255.255.255.255   Start           Any
       21:21         172.16.10.2       255.255.255.255   TCP             Any
       General       Include                             Off             0
    6  Any           192.168.2.6       255.255.255.255   Established     Any
       20:20         172.16.10.2       255.255.255.255   TCP             Any
       General       Include                             Off             0
Requests: 0 Passes: 0 Fails: 0
--------------------------------------------------------------------------------
 2   Traffic
    1  23:23         172.16.10.2       255.255.255.255   Established     Any
       Any           192.168.2.4       255.255.255.255   TCP             Any
       General       Include                             Off             0
    2  23:23         172.16.10.2       255.255.255.255   Established     Any
       Any           192.168.2.5       255.255.255.255   TCP             Any
       General       Include                             Off             0
    3  20:20         172.16.10.2       255.255.255.255   Any             Any
       Any           192.168.2.5       255.255.255.255   TCP             Any
       General       Include                             Off             0
    4  21:21         172.16.10.2       255.255.255.255   Established     Any
       Any           192.168.2.5       255.255.255.255   TCP             Any
       General       Include                             Off             0
    5  20:20         172.16.10.2       255.255.255.255   Any             Any
       Any           192.168.2.65      255.255.255.255   TCP             Any
       General       Include                             Off             0
    6  21:21         172.16.10.2       255.255.255.255   Established     Any
       Any           192.168.2.6       255.255.255.255   TCP             Any
       General       Include                             Off             0
Requests: 0 Passes: 0 Fails: 0
--------------------------------------------------------------------------------
```

Figure 22-10: Example output from the **show ip interface** command for IP filtering

```
Interface     Type     IP Address       Bc Fr PArp  Filt RIP Met.   SAMode  IPSc
Pri. Filt     Pol.Filt Network Mask     MTU   VJC   GRE  OSPF Met.  DBcast  Mul.
--------------------------------------------------------------------------------
eth0          Static   172.16.10.254    1  n  On    002  01         Pass    --
---           ---      255.255.255.0    1500  -     ---  0000000000 No      ---
ppp0          Static   172.16.1.5       1  n  -     001  01         Pass    --
---           ---      255.255.255.0    1500  Off   ---  0000000001 No      ---
--------------------------------------------------------------------------------
```

# Troubleshooting

## No Route Exists to the Remote Router

1. **If using RIP, wait for a RIP update.**

   Wait for at least one minute to ensure that a RIP update has been received (see Chapter 25, Routing Information Protocol (RIP)).

2. **Try using Telnet to access the remote router.**

   To Telnet from the local router to the remote router, and from the remote router to the local router, enter the command:

   ```
   telnet {ipadd|host}
   ```

3. **Check the underlying layer 2 interface.**

   Make sure all ports are enabled and functional.

   For PPP, to check that the PPP link has been established for both LCP and IP, enter the command:

   ```
   show ppp
   ```

   The display should be similar to the following output.

```
Name            Enabled  ifIndex  Over                  CP           State
-----------------------------------------------------------------------------
ppp0              YES    4                              IPCP         OPENED
                                  syn0                  LCP          OPENED
-----------------------------------------------------------------------------
```

   For information on how to check the PPP link, see Chapter 15, Point-to-Point Protocol (PPP).

4. **Make sure IP is in forwarding mode.**

   By default, IP operates in forwarding mode but an alternative server mode is also available. In server mode, the router does not route IP packets, but provides Telnet services, responds to SNMP requests, and uses TFTP to download software upgrades. In forwarding mode, the router routes IP packets, as well as performing all the functions of server mode. To see if the router has been set to server mode, use the command:

   ```
   show ip
   ```

   A router in server mode has DISABLED in the “IP Packet Forwarding” entry. To return the router to forwarding mode, use the command:

   ```
   enable ip forwarding
   ```

5. **Restart IP.**

   To restart the IP routing software (warm restart), enter the command:

   ```
   reset ip
   ```

6. **Contact your authorised distributor or reseller for assistance if necessary.**

   If the route still does not appear, contact your authorised distributor or reseller for assistance.

## Getting an IP Address from DHCP

**Problem** **You have set the router to get its IP address using DHCP but the router hasn't been given an IP address.**

**Solution**

- Check that the router's domain and host name are correct. To do this using the GUI, select Configuration > System > General.
- Check that the DHCP server can reach the router, by pinging the router from the DHCP server. Note that you do not receive a response if **Respond to ping** is unchecked on the Firewall Policy Options page (Configuration > Firewall > Interfaces > Policy options tab). This option is checked by default.
- Check that IP remote assignment is enabled. The router cannot obtain its address on an unnumbered PPP interface by DHCP if remote assignment is disabled. To enable it, use the **enable ip remoteassign** command on page 22-131.

## Telnet Fails

If telnet into the remote router fails, work logically through the following possible reasons for the failure.

1. Make sure that you have the correct IP address for the remote host.
2. Make sure that IP routing is configured correctly all the way from you to the remote host, by checking if you can **ping** the remote host. If you can, then routing is configured correctly. If you cannot, then routing is not configured correctly.
3. If step 2. determines that routing **is not** correctly configured, then try to work out where the routing is going wrong. Use trace route (the **trace** command) to determine the path that packets are trying to take to the remote host. This lets you identify the device that is not forwarding packets correctly. Then check the following:
    - route tables on that device
    - if that device has a firewall that is blocking packets
    - if that device is forwarding any IP traffic
4. If step 2. determines that routing **is** correctly configured, then check the following:
    - if the destination device is configured to allow telnet connections (for example, a router does not accept telnet connections when it is in secure mode)
    - if the destination device is configured to only allow telnet connections from certain addresses
    - if a firewall between you and the destination device is blocking telnet but allowing ping.

# Command Reference

This section describes the commands available on the router to configure and manage the IP routing module. The extended **ping** command on page 22-135 requires that the IPX and AppleTalk routing modules be enabled and correctly configured if IPX or AppleTalk addresses are used. See Chapter 36, Novell IPX for a detailed description of the commands required to enable and configure IPX or AppleTalk routing.

Some interface and port types mentioned in this chapter may not be supported on your router. The interface and port types that are available vary depending on your product's model, and whether an expansion unit (PIC) is installed. For more information, see the Hardware Reference for the router.

The shortest valid command is denoted by capital letters in the Syntax section. See "Conventions" on page lxv of About this Software Reference in the front of this manual for details of the conventions used to describe command syntax. See Appendix A, Messages for a complete list of error messages and their meanings.

# activate ddns update

**Syntax**

```
ACTivate DDNS UPdate
```

**Description** This command activates a Dynamic DNS update.

**Examples** To force a router to send a Dynamic DNS update, use the command:

```
act ddns up
```

**Related Commands**

**disable ddns**
**disable ddns debug**
**enable ddns**
**enable ddns debug**
**set ddns**
**show ddns**

# add ip arp

**Syntax**

```
ADD IP ARP=ipadd INTerface=atm-interface

ADD IP ARP=ipadd INTerface=ppp-interface

ADD IP ARP=ipadd INTerface=x25-interface
   CIRCuit=miox-circuit

ADD IP ARP=ipadd INTerface=fr-interface DLCI=dlci

ADD IP ARP=ipadd INTerface=eth-interface [ETHernet=macadd]

ADD IP ARP=ipadd INTerface=vlan ETHernet=macadd
   POrt=port-number
```

where:

- *ipadd* is an IP address in dotted decimal notation.
- *atm-interface* is an ATM interface name such as atm0.1
- *ppp-interface* is a PPP interface name such as ppp0 or ppp0-1
- *x25-interface* is an X.25 DTE interface name such as x25t0 or x25t0-1
- *miox-circuit* is the name of a MIOX circuit defined for an X.25 interface 1 to 15 characters long. The name is not case-sensitive.
- *fr-interface* is a Frame Relay interface name such as fr0 or fr0-1
- *dlci* is the Data Link Connection Identifier (DLCI) of a Frame Relay DLC (circuit).
- *eth-interface* is an Eth port name such as eth0 or eth0-1
- *vlan* is a VLAN name such as vlan1 or vlan1-1
- *macadd* is the physical Ethernet (MAC) address of a host.
- *port-number* is the physical switch port number. Port numbers start at 1 and end at m, where m is the highest numbered Ethernet switch port, including uplink ports.

**Description**

This command adds a static ARP entry to the ARP cache. This is typically used to add entries for hosts that do not support ARP or to speed up the address resolution function for a host. The ARP entry must not already exist.

The **arp** parameter specifies the IP address of the host.

The **interface** parameter specifies the interface over which the host can be reached. The specified interface must already exist. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces.

If the interface type is ETH, FR, VLAN, or X.25 DTE, then one of the following parameters must be present on the command line. If the interface type is ATM or PPP, the following parameters do not apply and must not be present. If the interface type is ATM inverse ARP must be turned off for the interface. This can be done with the **add ip interface** command on page 22-80 or the **set ip interface** command on page 22-155.

The **circuit** parameter specifies the MIOX circuit on an X.25 interface.

The **dlci** parameter specifies the physical address for the host on a Frame Relay interface.

The **ethernet** parameter specifies the physical (MAC) address for the host on the following:

- ETH interfaces
- VLAN interfaces

The **port** parameter specifies the physical switch port number in a VLAN. If the **interface** parameter specifies a VLAN interface, both the **ethernet** and **port** parameters are required. Otherwise, the **port** parameter is invalid. When configuring VLAN interfaces, both the **ethernet** and **port** parameters must be used.

**Examples**

To add a static ARP entry for a host with an Ethernet address of 00-00-00-08-31-9F and an IP address of 172.16.9.197 on interface eth0, use:

```
add ip arp=172.16.9.197 int=eth0 eth=00-00-00-08-31-9F
```

To add a static ARP entry for a host with an Ethernet address of 00-00-00-08-31-9F, port 3, and an IP address of 192.168.4.101 on interface vlan40, use:

```
add ip arp=192.168.4.101 int=vlan40 po=3
   eth=00-00-00-08-31-9F
```

To add a static ARP entry for a Frame Relay station with an IP address of 172.16.249.5 at the remote end of DLC 23, use:

```
add ip arp=172.16.249.5 int=fr0 dlci=23
```

To add a static ARP entry for a host with an Ethernet address of 00-00-00-08-31-9F and an IP address of 172.16.9.197 on interface vlan1, port 1, use:

```
add ip arp=172.16.9.197 int=vlan1 eth=00-00-00-08-31-9F po=1
```

**Related Commands**

**delete ip arp**
**set ip arp**
**show ip arp**

# add ip dns

**Syntax**

```
ADD IP DNS [DOMain={ANY|domain-name}]
   {INTerface=interface|PRIMary=ipadd [SECOndary=ipadd]}
```

where:

- *domain-name* is a string of up to 255 characters. Valid characters are uppercase and lowercase letters, digits (0-9), and the underscore.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.
- *ipadd* is an IP address in dotted decimal notation.

**Description**

This command adds a DNS server to the list of DNS servers used to resolve host names into IP addresses.

The **domain** parameter specifies the domain for which this DNS server is to be used to resolve host names. DNS requests for hosts in this domain are sent to this server. If **any** is specified, the name server is the default name server, and is used for domains not otherwise matched by another DNS entry. The default is **any**.

The default name server must be configured before domain-specific name servers can be configured. The maximum number of domain-specific name servers is 10.

The **interface** parameter specifies the interface over which the router learns the address of a primary and/or a secondary name server. The primary and secondary name server's addresses can be either statically configured using the **primary** and **secondary** parameters, or learned dynamically over an interface. Name servers can be learned via DHCP over an Ethernet or VLAN interface or via IPCP over a PPP interface. If the **interface** parameter is specified, the **primary** and **secondary** parameters are not required. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces.

The **primary** parameter specifies the IP address of the name server to be used as the primary name server for resolving hosts in the specified domain. If the **primary** parameter is specified, the **interface** parameter must not be specified.

The **secondary** parameter specifies the IP address of the name server to be used as the secondary name server for resolving hosts in the specified domain. If the **secondary** parameter is specified, the **interface** parameter must not be specified.

**Examples** To add primary and secondary name servers, with IP addresses of 192.168.20.1 and 192.168.20.2 respectively, for use as default name servers when the domain to be resolved does not match any of the domain suffixes specifically configured, use the command:

```
add ip dns prim=192.168.20.1 seco=192.168.20.2
```

The name servers are used when the name being resolved does not match any DNS domain suffixes specifically configured by commands such as in the following example.

To add primary and secondary name servers, with IP addresses of 192.168.10.1 and 192.168.10.2 respectively, for use when resolving host names in the domain "oranges.com", use the command:

```
add ip dns dom=oranges.com prim=192.168.10.1
   seco=192.168.10.2
```

**Related Commands**

**delete ip dns**
**set ip dns**
**show ip dns**

# add ip filter

**Syntax** Traffic filter:

```
ADD IP FILter=0..999 ACtion={INCLude|EXCLude} SOurce=ipadd
    [TYPE=TRAFfic] [SMask=ipadd]
    [SPort={port-name|port-id}] [DEStination=ipadd
    [DMask=ipadd]] [DPort={port-name|port-id}]
    [ICMPCode={icmp-code-name|icmp-code-id}]
    [ICmptype={icmp-type-name|icmp-type-id}]
    [LOG={4..1600|DISabled|Dump|ENabled|Full|Header|None|OF
    f|ON|Yes}] [OPtions={False|OFF|ON|NO|True|YES}]
    [PROTocol={protocol|Any|Icmp|Ospf|Tcp|Udp}]
    [SEssion={Any|Established|Start}] [SIze=size]
    [ENTry=1..255]
```

Policy filter:

```
ADD IP FILter=0..999 POLIcy=0..15 SOurce=ipadd
    [TYPE=POLIcy] [SMask=ipadd] [SPort={port-name|port-id}]
    [DEStination=ipadd [DMask=ipadd]]
    [DPort={port-name|port-id}]
    [ICMPCode={icmp-code-name|icmp-code-id}]
    [ICmptype={icmp-type-name|icmp-type-id}]
    [LOG={4..1600|DISabled|Dump|ENabled|Full|Header|None|OF
    f|ON|Yes}] [OPtions={False|OFF|ON|NO|True|YES}]
    [PROTocol={protocol|Any|Icmp|Ospf|Tcp|Udp}]
    [SEssion={Any|Established|Start}] [SIze=size]
    [ENTry=1..255]
```

Priority filter:

```
ADD IP FILter=0..999 PRIOrity=P0..P7 SOurce=ipadd
    [TYPE=PRIOrity] [SMask=ipadd]
    [SPort={port-name|port-id}] [DEStination=ipadd
    [DMask=ipadd]] [DPort={port-name|port-id}]
    [ICMPCode={icmp-code-name|icmp-code-id}]
    [ICmptype={icmp-type-name|icmp-type-id}]
    [LOG={4..1600|DISabled|Dump|ENabled|Full|Header|None|OF
    f|ON|Yes}] [OPtions={False|OFF|ON|NO|True|YES}]
    [PROTocol={protocol|Any|Icmp|Ospf|Tcp|Udp}]
    [SEssion={Any|Established|Start}] [SIze=64..65535]
    [ENTry=1..255]
```

Routing filter:

```
ADD IP FILter=0..999 ACtion={INCLude|EXCLude} SOurce=ipadd
    [TYPE=ROUting] [ENTry=1..255] [SMask=ipadd]
```

**Tip** Use of IP filters for route filtering has been superseded by route maps and prefix lists. However, it is still available for backwards compatibility.

where:

- *ipadd* is an IP address in dotted decimal notation.
- *port-name* is the predefined name for an IP port.
- *port-id* is an IP port number or a range in the format *low:high*.
- *icmp-code-name* is the predefined name for an ICMP reason code.
- *icmp-code-id* is the number of an ICMP reason code.
- *icmp-type-name* is the predefined name of an ICMP message type.
- *icmp-type-id* is the number of an ICMP message type.
- *protocol* is an IP protocol number.

**Description** This command adds a pattern to an IP traffic filter, policy filter, routing filter, or priority filter. The exact pattern should not already exist in the filter.

The **filter** parameter specifies the filter number, from 0 to 999, that the pattern is added to. When the **type** parameter is not specified, the router may use the filter number to help determine the filter type. See the description of the **type** parameter for further details.

The **source** parameter specifies the source IP address, in dotted decimal notation, for the pattern.

The **action** parameter is used for traffic and routing filters and specifies the action to take when the pattern is matched. If **include** is specified, the IP packet is processed and forwarded for traffic filters, or the IP route is selected, for routing filters. If **exclude** is specified, the IP packet is discarded for traffic filters, or the IP route is excluded for routing filters. The **action**, **policy**, and **priority** parameters are mutually exclusive so only one may be specified.

The **policy** parameter is used for policy-based routing and specifies the policy to use when the pattern is matched.

- For policy numbers from 0 to 7, routes with a matching policy are considered first.
- For policy numbers from 8 to 15, routes with a policy of *n*-8 (where *n* is the filter policy) are considered first, and the policy value *n*-8 is written into the TOS field of the packet.

The policy number is assigned to incoming packets but employed during forwarding (transmission). When no route is matched to the policy, the packet is routed as if no policies are present; only routes with no policy are considered. The **action**, **policy**, and **priority** parameters are mutually exclusive so only one may be specified.

The **priority** parameter is used for priority routing and specifies the priority when the pattern is matched. The priority number is assigned to incoming packets but employed during forwarding (transmission). Packets can be assigned a priority from p3 (highest) to p7 (lowest). The default is p5. Priority levels p0, p1, and p2 should not be used because they may conflict with the router’s system activities. The **action**, **policy**, and **priority** parameters are mutually exclusive so only one may be specified.

The **type** parameter specifies the type of filter the router creates. Four filter types are supported: **traffic**, **policy**, **priority** and **routing**. When **type** is not specified, the router determines the filter type based on the IP filter number and the specified parameters:

- Filters with a specified **policy** parameter are policy filters.
- Filters with a specified **priority** parameter are priority filters.
- Filters with a specified **action** parameter are either traffic or routing filters. If the filter number set is:
    - between 0 to 99, they are traffic filters
    - between 100 to 999, they are routing filters, as long as the only other parameters specified are the **source**, **entry** and **smask** parameters. If any other parameter is specified the filter is a traffic filter.

We recommend always defining this parameter because a traffic filter created without specifying **type=traffic**, and with a filter number between 100 and 999, can default to a routing filter. Routing filters are used in conjunction with Border Gateway Protocol (BGP) only. However, you can still create a routing filter even if BGP is not on your router.

See these sections in the IP chapter of the Software Reference for more information about using traffic, policy and priority filters:

- "Traffic Filters" on page 22-36
- "Policy-Based Routing" on page 22-26
- "Priority-Based Routing" on page 22-27

An interface may have a maximum of one traffic filter, one policy filter, and one priority filter, but the same traffic, policy or priority filter can be assigned to more than one interface.

The **entry** parameter specifies the entry number in the filter that this new pattern occupies. Existing patterns with the same or higher entry numbers are pushed down the filter. The default is to add the new pattern to the end of the filter.

The **smask** parameter specifies the mask, in dotted decimal notation to apply to source addresses for this pattern. The mask is used to determine the portion of the source IP address in the IP packet that is significant for comparison with this pattern. The values of **source** and **smask** must be compatible. For each bit in **smask** that is set to zero, the equivalent bit in **source** must also be zero. If either **source** or **smask** is 0.0.0.0, then both must be 0.0.0.0. The default is 255.255.255.255, unless **source** is 0.0.0.0, in which case the default **smask** is 0.0.0.0.

The **sport** parameter specifies the source port to check against for this pattern as the recognised name of a well-known UDP or TCP port (Table 22-7 on page 22-73), a decimal value from 0 to 65535, or a range of numbers formatted *low:high*. If *low* is omitted, 0 is assumed; if *high* is omitted, the maximum port number is assumed. If a port other than **any** is specified, the **protocol** parameter is required and must be TCP or UDP. The default is **any**.

The **destination** parameter specifies the destination IP address for the pattern in dotted decimal notation. The default is 0.0.0.0.

The **dmask** parameter specifies the mask in dotted decimal notation to apply to the destination address for this pattern. The mask determines the portion of the destination IP address in the IP packet that is significant for comparison with this pattern. If **dmask** is specified, **destination** must also be specified. The values of **destination** and **dmask** must be compatible. For each bit in **dmask** that is set to zero, the equivalent bit in **destination** must also be zero. If either **destination** or **dmask** is 0.0.0.0, then both must be 0.0.0.0. If **destination** is not specified or is 0.0.0.0, the default **dmask** is 0.0.0.0. If **destination** is specified and is not 0.0.0.0, the default **dmask** is 255.255.255.255.

The **dport** parameter specifies the destination port to check against for this pattern as the recognised name of a well-known UDP or TCP port (Table 22-7 on page 22-73), a decimal value from 0 to 65535, or a range of numbers formatted *low:high*. If *low* is omitted, 0 is assumed; if *high* is omitted, the maximum port number is assumed. If a port other than **any** is specified, the **protocol** parameter is required and must be TCP or UDP. The default is **any**.

If a pattern for Telnet is not explicitly added to a filter assigned to an interface, all Telnet traffic received over the specified interface is discarded. This prevents Telnet connections to the router itself via the interface. To enable access to the router’s command prompt via Telnet, a pattern for Telnet must be added to the filter for the interface.

Table 22-7: Predefined port names used by the IP filtering process

| Port Name | Number | Protocol[1] | Description |
|---|---|---|---|
| ANY | - | - | Any port |
| BOOTPC | 68 | UDP | Bootstrap Protocol Client |
| BOOTPS | 67 | UDP | Bootstrap Protocol Server |
| DOMAIN | 53 | TCP/UDP | Domain Name Server |
| FINGER | 79 | TCP | Finger |
| FTP | 21 | TCP | File Transfer [Control] |
| FTPDATA | 20 | TCP | File Transfer [Default Data] |
| GOPHER | 70 | TCP | Gopher |
| HOSTNAME | 101 | TCP/UDP | NIC Host Name Server |
| IPX | 213 | TCP/UDP | IPX |
| KERBEROS | 88 | UDP | Kerberos |
| LOGIN | 49 | UDP | Login Host Protocol |
| MSGICP | 29 | TCP/UDP | MSG ICP |
| NAMESERVER | 42 | UDP | Host Name Server |
| NEWS | 144 | TCP | NewS |
| NNTP | 119 | TCP | Network News Transfer Protocol |
| NTP | 123 | TCP | Network Time Protocol |
| RTELNET | 107 | TCP/UDP | Remote Telnet Service |
| SFTP | 115 | TCP/UDP | Simple File Transfer Protocol |
| SMTP | 25 | TCP | Simple Mail Transfer |
| SNMP | 161 | UDP | SNMP |
| SNMPTRAP | 162 | UDP | SNMPTRAP |
| SYSTAT | 11 | TCP | Active Users |

Table 22-7: Predefined port names used by the IP filtering process (cont.)

| Port Name | Number | Protocol[1] | Description |
|---|---|---|---|
| TELNET | 23 | TCP | Telnet |
| TFTP | 69 | UDP | Trivial File Transfer |
| TIME | 37 | TCP/UDP | Time |
| UUCP | 540 | TCP | uucpd |
| UUCPRLOGIN | 541 | TCP/UDP | uucp-rlogin |
| WWWHTTP | 80 | TCP | World Wide Web HTTP |
| XNSTIME | 52 | TCP/UDP | XNS Time Protocol |

[1] *The protocol typically used with the port.*

The **icmptype** and **icmpcode** parameters specify the ICMP message type and ICMP message reason code to match against the ICMP type and code fields in an ICMP packet. The **icmptype** parameter specifies the ICMP message type to match as a decimal value from 0 to 255, or the recognised name of an ICMP type (Table 22-8 on page 22-74). The **icmpcode** parameter specifies the ICMP message reason code to match as a decimal value from 0 to 255, or the recognised name of an ICMP reason code (Table 22-9 on page 22-75). Both parameters are valid when the **protocol** parameter is set to **icmp**.

Table 22-8: Predefined ICMP type names used by the IP filtering process

| ICMP Type Name | ICMP Type Value | ICMP Type Description | ICMP Codes Supported |
|---|---|---|---|
| ECHORPLY | 0 | Echo reply messages | No |
| UNREACHABLE | 3 | Unreachable messages | Yes |
| QUENCH | 4 | Source quench messages | No |
| REDIRECT | 5 | Redirect messages | Yes |
| ECHO | 8 | Echo request messages | No |
| ADVERTISEMENT | 9 | Router advertisement messages | No |
| SOLICITATION | 10 | Router solicitation messages | No |
| TIMEEXCEED | 11 | Time exceeded messages | Yes |
| PARAMETER | 12 | Parameter problem messages | Yes |
| TSTAMP | 13 | Timestamp request messages | No |
| TSTAMPRPLY | 14 | Timestamp reply messages | No |
| INFOREQ | 15 | Information request messages | No |
| INFOREP | 16 | Information reply message | No |
| ADDRREQ | 17 | Address mask request messages | No |
| ADDRREP | 18 | Address mask reply messages | No |
| NAMEREQ | 37 | Name request messages | No |
| NAMERPLY | 38 | Name reply messages | No |

Table 22-9: Predefined ICMP code names used by the IP filtering process

| ICMP Code Name | ICMP Code Value | ICMP Code Description | Applies to ICMP Type Name... |
|---|---|---|---|
| ANY | (any) | Any ICMP code | (any) |
| NETUNREACH | 0 | Network unreachable | UNREACHABLE |
| HOSTUNREACH | 1 | Host unreachable | UNREACHABLE |
| PROTUNREACH | 2 | Protocol unreachable | UNREACHABLE |
| PORTUNREACH | 3 | Port unreachable | UNREACHABLE |
| FRAGMENT | 4 | Fragmentation is needed but "do not fragment" flag is set | UNREACHABLE |
| SOURCEROUTE | 5 | Source route failed | UNREACHABLE |
| NETUNKNOWN | 6 | Destination network unknown | UNREACHABLE |
| HOSTUNKNOWN | 7 | Destination host unknown | UNREACHABLE |
| HOSTISOLATED | 8 | Source host isolated | UNREACHABLE |
| NETCOMM | 9 | Communication with destination network administratively prohibited | UNREACHABLE |
| HOSTCOMM | 10 | Communication with destination host administratively prohibited | UNREACHABLE |
| NETTOS | 11 | Network unreachable for selected TOS | UNREACHABLE |
| HOSTTOS | 12 | Host unreachable for selected TOS | UNREACHABLE |
| FILTER | 13 | Communication administratively prohibited due to filtering | UNREACHABLE |
| HOSTPREC | 14 | Host precedence violation | UNREACHABLE |
| PRECEDENT | 15 | Precedence cutoff in effect | UNREACHABLE |
| NETREDIRECT | 0 | Redirect datagrams for the network | REDIRECT |
| HOSTREDIRECT | 1 | Redirect datagram for the host | REDIRECT |
| NETRTOS | 2 | Redirect datagrams for the TOS and network | REDIRECT |
| HOSTRTOS | 3 | Redirect datagrams for the TOS and host | REDIRECT |
| TTL | 0 | TTL exceeded in transit | TIMEEXCEED |
| FRAGREASSM | 1 | Fragment reassembly time exceeded | TIMEEXCEED |
| PTRPROBLEM | 0 | Pointer value referencing the octet in the original IP packet caused problem | PARAMETER |
| NOPTR | 1 | No pointer present | PARAMETER |

The **log** parameter specifies whether matches to a filter entry result in a message being sent to the router's Logging facility, and the content of the log messages. This parameter enables logging of the IP packet filtering process down to the level of an individual filter entry. The default is **none**.

If you specify a number from 4 to 1600, the filter number, entry number, and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are logged with a message type/subtype of IPFIL/PASS (for patterns with an **include** action) or IPFIL/FAIL (for patterns with an **exclude** action). In addition, the first 4 to 1600 octets of the data portion of TCP, UDP, and ICMP packets or the first 4 to 1600 octets after the IP header of other protocol packets are logged with a message type/subtype of IPFIL/DUMP.

If you specify **dump** or **full**, the filter number, entry number, and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are logged with a message type/subtype of IPFIL/PASS (for patterns with an **include** action) or IPFIL/FAIL (for patterns with an **exclude** action). In addition, the first 32 octets of the data portion of TCP, UDP, and ICMP packets or the first 32 octets after the IP header of other protocol packets are logged with a message type/subtype of IPFIL/DUMP.

If you specify **enabled**, **header**, **on**, or **yes**, the filter number, entry number, and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are logged with a message type/subtype of IPFIL/PASS (for patterns with an **include** action) or IPFIL/FAIL (for patterns with an **exclude** action).

If you specify **disabled**, **none**, or **off**, matches to the filter entry are not logged.

The **options** parameter specifies the IP options field to use to check against the pattern. If **yes**, the pattern matches IP packets with options set; if **no**, the pattern matches packets without options set. The default is to match IP packets with or without IP options set.

The **protocol** parameter specifies the protocol to check against for this pattern as a decimal value from 0 to 65534. Valid protocol names are:

- Internet Control Message Protocol (ICMP)
- Open Shortest Path First Protocol (OSPF)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

If either **sport** or **dport** is specified, **protocol** must be defined as TCP or UDP. Specifying TCP or UDP filters packets from companion protocols, for example ICMP, RIP and OSPF, that do not use TCP or UDP as a transport mechanism. The default is **any**.

The **session** parameter specifies the type of TCP packet to match, and can be used when the **protocol** parameter specifies TCP. If **start** is specified, the pattern matches TCP packets with the SYN bit set and the ACK bit clear. If **established** is specified, the pattern matches TCP packets with either the SYN bit clear or the ACK bit set. If **any** is specified, the pattern matches any TCP packet. The default is **any**.

The **size** parameter specifies the maximum reassembled size to match against for each IP fragment. If the fragment’s offset plus size is greater than the value specified, the fragment is discarded.

**Examples** To create filters to allow only FTP traffic between two hosts with IP addresses 172.16.10.2 and 192.168.2.6, use the commands:

```
add ip fil=1 type=traf so=192.168.2.6 sm=255.255.255.255
    des=172.16.10.2 dp=ftp prot=t ac=incl

add ip fil=1 type=traf so=192.168.2.6 sm=255.255.255.255
    des=172.16.10.2 dp=ftpdata prot=t ac=incl

add ip fil=2 type=traf so=172.16.10.2 sm=255.255.255.255
    sp=ftp des=192.168.2.6 prot=t ac=incl

add ip fil=2 type=traf so=172.16.10.2 sm=255.255.255.255
    sp=ftpdata des=192.168.2.6 prot=t ac=incl
```

**Related Commands**

**add bgp peer** in Chapter 27, Border Gateway Protocol version 4 (BGP-4)
**add ip route filter** in Chapter 28, Filtering IP Routes
**delete ip filter**
**delete ip route filter** in Chapter 28, Filtering IP Routes
**set bgp peer** in Chapter 27, Border Gateway Protocol version 4 (BGP-4)
**set ip filter**
**show ip filter**
**show ip route filter** in Chapter 28, Filtering IP Routes

# add ip helper

**Syntax**

```
ADD IP HElper DEStination=ipadd INTerface=interface
   POrt=port-number
```

where:

- *ipadd* is an IP address in dotted decimal notation.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.
- *port-number* is a UDP port number from 1 to 65535, or one of the predefined UDP port names DNS (port 53), NT or NETBIOS (ports 137 and 138), TACACS (port 49), TIME (port 37) or TFTP (port 69).

**Description**

This command adds a port or a set of named ports to the list of UDP ports to listen for on the specified interface. When a broadcast UDP packet is received on the specified interface with the specified destination port number it is redirected to the destination IP address. This allows all network broadcast packets to be delivered across the internet to appropriate servicing hosts. Multiple invocations of this command can be used for forward packets for several UDP ports to the same IP address, to forward packets for a single UDP port to multiple IP addresses.

The **destination** parameter specifies the IP address to which the UDP broadcast traffic is forward.

The **interface** parameter specifies the interface to which the UDP port list is assigned. UDP broadcasts are forwarded that are received for the specified interface for one of the UDP ports in the UDP port list. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces.

The **port** parameter specifies the UDP port, as a decimal number from 1 to 65535, or the recognised name of a UDP port set. Broadcast traffic received by the router on the specified port or set of ports is redirected to the IP host at the destination address. Up to 32 ports can be specified.

**Examples**

To forward all NETBIOS broadcasts received via interface eth0 to IP address 192.168.202.3, use the command:

```
add ip he po=netbios des=192.168.202.3 int=eth0
```

To forward all broadcasts to UDP port 3001 received via interface eth0 to IP address 192.168.100.2, use the command:

```
add ip he po=3001 int=eth0 des=192.168.100.2
```

**Related Commands**

**delete ip helper**
**disable ip helper**
**enable ip helper**
**show ip helper**

# add ip host

**Syntax**

```
ADD IP HOst=name IPaddress=ipadd
```

where:

- *name* is a string up to 60 characters long. If the string contains spaces, it must be in double quotes.
- *ipadd* is an IP address in dotted decimal notation.

**Description**

This command adds a user-defined name for an IP host to the host name table. The host name table makes it easier to Telnet to commonly accessed hosts by enabling the user to enter a shorter, easier to remember name for the host rather than the host's full IP address or domain name. The name can also be used with the **ping** command on page 22-135.

The **host** parameter specifies the user-defined name for the IP host. A host with the same name must not already exist in the host name table. When a host name is specified in the Telnet command, the entire name is used to match a name in the host name table. All characters are used in the comparison, including nonalphabetic characters if they are present.

The **ipaddress** parameter specifies the IP address of the host.

**Examples**

To add the host name "zaphod" to the host name table for an IP host with an IP address of 172.16.1.5 and the domain name "zaphod.company.com", use the command:

```
add ip host=Zaphod IP=172.16.1.5
```

To Telnet to the host, use any of the following commands:

```
telnet zaphod
telnet zaphod.company.com
telnet 172.16.1.5
```

**Related Commands**

**delete ip host**
**set ip host**
**set ip nameserver**
**set ip secondarynameserver**
**show ip host**

# add ip interface

**Syntax**

```
ADD IP INTerface=interface IPaddress={ipadd|DHCP}
   [BROadcast={0|1}]
   [DIRectedbroadcast={False|NO|OFF|ON|True|YES}]
   [FILter={0..999|NONE}] [FRAgment={NO|OFF|ON|YES}]
   [GRAtuitousarp={ON|OFF}] [GRE={0..100|NONE}]
   [IGMPProxy={OFF|UPstream|DOWNstream}]
   [INVersearp={ON|OFF}] [MASK=ipadd] [METric=1..16]
   [MULticast={BOTH|NO|OFF|ON|RECeive|SENd|YES}]
   [NOTIfyospfdown={False|NO|OFF|ON|True|YES}]
   [OSPFmetric=1..65534] [POLicyfilter={0..999|NONE}]
   [PRIorityfilter={0..999|NONE}]
   [PROxyarp={False|NO|OFF|ON|True|YES|LOCal|STrict|
   DEFRoute}] [RIPMetric=1..16]
   [SAMode={Block|Passthrough}]
   [VJC={False|NO|OFF|ON|True|YES}]
   [VLANPRiority={0..7|None}] [VLantag={1..4094|None}]
```

where:

- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.
- *ipadd* is an IP address in dotted decimal notation.

**Description**

This command adds a logical interface to the IP module. A maximum of 1000 interfaces can be added.

This command requires a user with security officer privilege when the router is in security mode.

The **interface** parameter specifies the name of the logical interface, and implicitly, the attached Layer 2 interface. The Layer 2 interface must already be configured. The IP interface must not already be assigned to the IP module. At least two interfaces must be defined before the router can route IP packets, but only one interface (usually eth0) needs to be defined when the router is acting as a server. When an interface is added, it is automatically enabled. Only one logical interface may be configured to the same IP network or subnet. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces. If you are using the GUI to view interfaces and have configured a large number, the page may take several minutes to display.

The **broadcast** parameter specifies whether to use a broadcast address with all 1s or all 0s. The default is 1. An all 0s setting contradicts current RFCs and is only provided for backwards compatibility with some older UNIX systems.

The **directedbroadcast** parameter specifies whether the router allows network or subnet broadcasts to be forwarded to the network directly attached to the logical interface. The default is **no**.

The **filter** parameter specifies the traffic filter to apply to IP packets transmitted or received over the logical interface. The filter must already have been defined with the **add ip filter** command on page 22-70. A logical interface may have a maximum of one traffic filter, one policy filter and one priority filter, but the same traffic, policy or priority filter can be assigned to more than one interface. Traffic filters are applied to packets received via the logical interface. The default is to not apply a filter.

The **fragment** parameter specifies whether the "*Do not fragment*" bit is obeyed for outgoing IP packets that are larger than the MTU of the interface. If **yes**, the "*Do not fragment*" bit is ignored and outgoing IP packets larger than the MTU of the interface are fragmented. This is particularly useful for interfaces configured with GRE, SA and/or IPsec encapsulation, which can potentially increase packet sizes beyond the MTU of the interface. If **no**, the "*Do not fragment*" bit is obeyed and IP packets larger than the MTU are discarded. This is normal behaviour for IP. The **fragment** parameter has no effect on packets smaller than the interface MTU. The default is **no**.

The **gratuitousarp** parameter enables or disables the acceptance of gratuitous ARP requests or replies. The default is **on**.

The **gre** parameter specifies the GRE (Generic Routing Encapsulation) entity associated with the logical interface. The GRE entity must have been created previously with the **add gre** command on page 30-12 of Chapter 30, Generic Routing Encapsulation (GRE). The default is **none**.

The **igmpproxy** parameter specifies the status of IGMP proxying for the specified interface. If you specify **off**, the interface does not do IGMP Proxy. If you specify **upstream**, the interface passes IGMP messages in the upstream direction. A router can have only one interface when the IGMP proxy direction is upstream. If you specify **downstream**, the interface can receive IGMP messages from the downstream direction. The default is **off**. To display information about IGMP and multicast group membership for each IP interface, use the **show ip igmp** command on page 24-99 of Chapter 24, IP Multicasting.

The **inversearp** parameter enables or disables the operation of the Inverse Address Resolution Protocol (INVARP) on ATM interfaces. The **inversearp** parameter must be set to **on** for IPoA configurations, and to **off** for RFC 1483 Routed configurations. The default is **off**. (Inverse ARP is always on for Frame Relay interfaces.)

The **ipaddress** parameter specifies the IP address of the logical interface. If **dhcp** is specified, the router acts as a DHCP client and obtains the configuration of the IP interface via DHCP. Table 22-10 on page 22-82 lists the parameters from the DHCP reply that the router uses. If an IP interface is configured to use DHCP to obtain its IP address and subnet mask, the interface does not take part in IP routing until the IP address and subnet mask have been set by DHCP.

If different interfaces on a device need to be uniquely distinguished, then extended DHCP identification is needed, and the **extendid** parameter in the **set dhcp** command must be on before DHCP clients are created. See the **set dhcp** command on page 23-38 of Chapter 23, Dynamic Host Configuration Protocol (DHCP).

Table 22-10: DHCP reply parameters used by the router for configuring IP

| DHCP Parameter | Purpose |
|---|---|
| IP address and mask | IP address and subnet mask for the IP interface. |
| DNS Servers | DNS server addresses added to the list of IP name servers. A primary name server and a secondary name server are supported. Name servers are normally added with the **set ip nameserver** command on page 22-162 and the **set ip secondarynameserver** command on page 22-170. |
| Gateway | Default route is added over the specified interface with the next hop set to the gateway address.<br>If a default route does already exist on the router, the gateway parameter in the DHCP reply is ignored. |
| Domain Name | Domain name of the router. |

Remote address assignment must be enabled using the **enable ip remoteassign** command on page 22-131 before IP interfaces accept addresses dynamically assigned by DHCP.

The **mask** parameter specifies the subnet mask for the logical interface. The value must be consistent with the value specified for the **ipaddress** parameter. The default is the network mask for the address class of the IP address (for example, 255.255.0.0 for a Class B address, 255.255.255.0 for a Class C address). If **ipaddress** is set to **dhcp**, the **mask** parameter should not be set because the subnet mask received from the DHCP server is used.

The **multicast** parameter specifies whether the interface receives and forwards multicast packets when DVMRP and PIM are not enabled. If **both** or **on** is specified, the router both sends and receives multicast packets. If **off**, the router neither sends nor receives multicast packets. If **receive** is specified, the router receives but does not send multicast packets. If **send** is specified, the router sends but does not receive multicast packets. Note that this parameter applies to the entire IP interface, not an individual logical interface. Setting it on one logical interface sets it on all other logical interfaces associated with the same IP interface. This parameter determines the interface’s static behaviour for multicast packets. When DVMRP or PIM-SM is enabled, it determines the forwarding behaviour of interfaces dynamically, and this parameter has no effect (Chapter 22, IP Multicasting). The default is **receive**.

The **notifyospfdown** parameter specifies whether IP generates a notification to OSPF when this interface goes down if it is a PPP on-demand interface. IP always sends notifications for all other interfaces, even if this parameter is set to **no**. If **on**, **true** or **yes** is specified, then IP notifies OSPF when the interface goes down and OSPF sets the interface state to Down. OSPF does not send Hello messages to the interface, and OSPF is inactive on the interface until it receives an Up notification. If **false**, **no** or **off** is specified, IP does not notify OSPF when the interface goes down. OSPF continues to consider the interface state as Up, and keeps sending Hello packets. These Hello packets allow OSPF to bring the link up for on-demand PPP links. Note that this parameter applies to the entire IP interface, not an individual logical interface. Setting it on one logical interface sets it on all other logical interfaces associated with the same IP interface. The default value for this parameter is **yes**.

The **ospfmetric** parameter specifies the cost of crossing the logical interface for OSPF. The default is 1.

The **policyfilter** parameter specifies the policy filter to apply to IP packets received over the logical interface. The filter must already have been defined with the **add ip filter** command on page 22-70. A logical interface may have a maximum of one traffic filter, one policy filter, and one priority filter. However, the same traffic, policy or priority filter can be assigned to more than one interface. Policy filters are applied to packets when they are transmitted. The default is not to apply a filter.

The **priorityfilter** parameter specifies the priority filter to apply to IP packets transmitted over the logical interface. The filter must already have been defined with the **add ip filter** command on page 22-70. A logical interface may have a maximum of one traffic filter, one policy filter, and one priority filter. However, the same traffic, policy or priority filter can be assigned to more than one interface. Priority filters are applied to packets as they are transmitted. The default is not to apply a filter.

The **proxyarp** parameter enables or disables proxy ARP responses to ARP requests. This parameter is valid for Eth and VLAN interfaces. The default is **on**.

If **no**, **off**, or **false** is specified, the router does not act as a proxy for ARP. For all other options the router transmits proxy ARP responses to requests for addresses that can be reached via specific routes if they exist. If you specify **yes**, **on**, **true** or **strict**, the router only proxy responds if it has a specific route to the address being requested, excluding the interface route that the ARP request arrived from. It ignores all other APR requests. You can increase the range of ARP requests that the router proxy responds to by using either the **local** or **defroute** parameter. When you specify **defroute**, the router responds to all proxy ARP Requests to remote locations. For addresses where there is no specific route, the router transmits a proxy ARP response containing its default route (0.0.0.0), if it exists. When you specify **local**, the router makes proxy responses to ARP requests for the following types of addresses:

- addresses to which it has a specific route
- addresses within the same subnet as the interface on which the ARP request was received - referred to as *local* ARP requests

For local ARP requests, the router responds with its own MAC address to all ARP requests within the local subnet. This means that all hosts communicating within the subnet must send their packets to the MAC address of the router.This prevents the hosts from using MAC address resolution to communicate directly with one another, and enables the router to have control over which hosts may communicate with one another. When local proxy ARP is enabled on an interface, the router does not send ICMP redirect messages on that interface (for information about ICMP redirect messages, see “ICMP” on page 22-19).

⚠ Setting the router to **defroute** mode is a departure from RFC 1027.

The **ripmetric** parameter specifies the cost of crossing the logical interface for RIP. The default is 1. The **metric** parameter is also accepted for backwards compatibility.

The **samode** parameter specifies how the logical interface handles IP packets that do not belong to one of the security associations assigned to the interface. If **block** is specified, IP packets that do not belong to a security association assigned to the logical interface are blocked from transiting the interface and are discarded. If **passthrough** is specified, IP packets that do not belong to a security association assigned to the logical interface are allowed to transit the interface and are forwarded as normal by the IP routing software. The default is **block**. This parameter has affect when one or more security associations have been assigned to the logical interface with the **add ip sa** command on page 22-95.

The **vjc** parameter is valid for Point-to-Point Protocol (PPP) and X25T interfaces, and specifies whether Van Jacobson header compression is to be used on the Layer 2 interface. The **vjc** parameter applies to all logical interfaces attached to the same Layer 2 interface. Changing the setting on one logical interface alters the setting on the others attached to the Layer 2 interface. Compression provides the most advantage on slower link speeds (up to 48 kbps). At speeds of 64 kbps and higher, compression actually reduces efficiency and so should be disabled. Van Jacobson's TCP/IP header compression should not be enabled on a multilink PPP interface. The default is **off**.

The **vlanpriority** parameter specifies the value of the 802.1p User Priority field of the VLAN tag. This priority is written into all VLAN-tagged frames sent out the interface. Downstream routers may use this priority to determine the quality of service the frame receives. This parameter is only valid when **vlantag** is specified. The default is **none** when **vlantag**=**none**, and 0 when **vlantag** specifies a value.

The **vlantag** parameter specifies the VID (VLAN Identifier) to be included in the header of each frame that is transmitted over the logical interface. This parameter is valid for Eth interfaces only. Multiple logical interfaces on the same physical interface can share the same VLAN tag. The default is **none**, which means no VID is included. For more information, see "VLAN Tagging on Eth Interfaces" on page 22-29.

**Examples** To add PPP interface 0 (logical interface ppp0-0) with an IP address of 172.16.248.33, a subnet mask of 255.255.255.0, a metric of 5 and Van Jacobson's header compression, use the command:

```
add ip int=ppp0 ip=172.16.248.33 mask=255.255.255.0 ripm=5
    vjc=on
```

To add a second logical interface to PPP interface 0 (logical interface ppp0-1) with an IP address of 172.16.200.1, a subnet mask of 255.255.255.0, a metric of 5 and Van Jacobson's header compression, use the command:

```
add ip int=ppp0-1 ip=172.16.200.1 mask=255.255.255.0 ripm=5
    vjc=on
```

To add Ethernet interface 0 with an IP address of 202.36.163.1, a mask of 255.255.255.192 and associate the IP interface with GRE entity 1, use the command:

```
add ip int=eth0 ip=202.36.163.1 mask=255.255.255.192 gre=1
```

To create two logical interfaces on the eth0 interface, tag them with different VLAN tags, and give traffic on one a higher priority, use the commands:

```
add ip int=eth0-0 ip=192.168.1.1 vl=2 vlanpr=2
```

```
add ip int=eth0-1 ip=192.168.2.1 vl=3 vlanpr=3
```

You could use these two interfaces to separate and prioritise traffic destined for two different users.

**Related Commands**

**delete ip interface**
**disable ip interface**
**enable ip interface**
**reset ip interface**
**set ip interface**
**show ip igmp**
**show ip interface**

# add ip local

**Syntax**

```
ADD IP LOCal=1..15 IPaddress=ipadd [FILter={0..999|None}]
    [GRE={0..100|None}] [POLicyfilter={0..999|None}]
    [PRIorityfilter={0..999|None}]
```

where *ipadd* is an IP address in dotted decimal notation

**Description**

This command adds a local interface to the router. Up to fifteen local interfaces can be added to a single router. These are in addition to the default local interface that is automatically added at start up, and can be configured through the **set ip local** command.

The **local** parameter specified is a unique identifying number that is used to identify a particular local interface. The naming convention for this interface is the concatenation of the word *local* along with this identifying number.

The **ipaddress** parameter specifies the IP address of the interface. This can be any valid IP address. Note that specifying an IP address of 0.0.0.0 effectively "unsets" the IP address of the local interface specified.

The **filter** parameter specifies which traffic filter is to be applied to IP packets transmitted or received over the interface. The filter must already have been defined with the **add ip filter** command on page 22-70. An interface may have a maximum of one traffic filter, one policy filter and one priority filter, but the same traffic, policy or priority filter can be assigned to more than one interface. Traffic filters are applied to packets received via the interface. The default is not to apply a filter.

The **gre** parameter specifies the GRE (Generic Routing Encapsulation) entity associated with the interface. The specified GRE entity must have been created previously using the **add gre** command on page 30-12 of Chapter 30, Generic Routing Encapsulation (GRE). The default is **none**.

The **policyfilter** parameter specifies the policy filter to be applied to IP packets received over the interface. The filter must already have been defined with the **add ip filter** command on page 22-70. Although an interface can only have one traffic filter, one policy filter and one priority filter; each of these filters can be assigned to more than one interface. Policy filters are applied to packets as they are transmitted. The default is **none;** that is, not to apply a filter.

The **priorityfilter** parameter specifies the filter to be applied to IP packets received over the interface. The filter must have already been defined with the **add ip filter** command on page 22-70. Although an interface can only have one traffic filter, policy filter, and priority filter; each of these filters can be assigned to more than one interface. Priority filters are applied to packets as they are transmitted. The default is **none**; that is, not to apply a filter.

**Examples**

To add the local interface 3 with an IP address of 192.168.33.1, use:

```
add ip loc=3 ip=192.168.33.1
```

**Related Commands**

**delete ip local**
**set ip local**
**show ip interface**

# add ip nat

**Syntax**

```
ADD IP NAT IP=ipadd [MASK=ipadd] [GBLIPaddress=ipadd]
    [GBLMask=ipadd] [GBLPort=port] [GBLINterface=interface]
    [POrt=port]
    [PROTocol={protocol|ALL|EGP|GRE|ICmp|OSPF|SA|TCp|UDp}]
```

where:

- *ipadd* is an IP address in dotted decimal notation.
- *port* is an IP port number or the predefined name for an IP service.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.
- *protocol* is an IP protocol number.

**Description** This command adds a local private IP network to the address translation table that NAT uses. The method of NAT translation depends on parameters in this command.

- To create a static NAT, use the **ip** and **gblipaddress** parameters.
- To create a dynamic NAT, use the **ip**, **mask**, **gblipaddress**, and **gblmask** parameters.
- To create a dynamic ENAT, use the **ip**, **mask**, and **gblipaddress**.
- To create a static ENAT, use the **ip**, **protocol**, **port**, **gblipaddress**, and **gblport** parameters.
- To create an interface ENAT, use the parameters **ip**, **mask**, and **gblinterface**.

NAT must be explicitly enabled with the **enable ip nat** command on page 22-130 before these translations take effect.

The **ip** parameter specifies either a host or network IP address for the private network. This parameter can be used with the **mask** parameter to specify a range of IP address for the private network.

The **protocol** parameter specifies the IP protocol number or the name of a predefined protocol type to be used with a static ENAT entry. If **tcp** or **udp** is specified, then **port** must also be specified.

The **port** parameter specifies the port number or service name (Table 22-11) for the port used on the private IP host when specifying a static ENAT entry.

Table 22-11: Service names for use with Network Address Translation (NAT)

| Service Names | Value |
|---|---|
| ECHO | 7 |
| DISCARD | 9 |
| FTP | 21 |
| TELNET | 23 |
| SMTP | 25 |
| TIME | 37 |
| DOMAIN | 53 |
| BOOTPS | 67 |
| BOOTPC | 68 |
| TFTP | 69 |
| GOPHER | 70 |
| FINGER | 79 |
| WWW | 80 |
| KERBEROS | 88 |
| RTELNET | 107 |
| POP2 | 109 |
| POP3 | 110 |
| SNMP | 161 |
| SNMPTRAP | 162 |
| BGP | 179 |
| RIP | 520 |
| PPTP | 1723 |

The **gblipaddress** parameter specifies either an officially assigned global IP address or the start of a range of officially assigned global IP addresses. This parameter can be used with the **gblmask** parameter to specify the range of global IP addresses for an entry.

The **gblport** parameter specifies the port number or service name (Table 22-11 on page 22-88) for the port available to global Internet access when creating a static ENAT.

The **gblinterface** parameter specifies the interface that has or will dynamically obtain an officially assigned global IP address. Valid interfaces are:

- eth (such as eth0, eth0-1)
- PPP (such as ppp0, ppp1-1)
- VLAN (such as vlan1, vlan1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces.

A static ENAT entry can be added to an existing dynamic ENAT entry only.

**Examples** To configure a static NAT to translate between the private IP address 10.1.1.2 and the officially assigned global IP address 202.1.1.1, use the command:

```
add ip nat IP=10.1.1.2 gblip=202.1.1.1
```

To configure a dynamic NAT entry to allow an entire private network with the IP address 10.1.1.0 to use a block of 32 officially assigned global IP addresses starting at 202.1.1.1, use the command:

```
add ip nat ip=10.1.1.0 mask=255.0.0.0 gblip=202.1.1.1
    gblm=255.255.255.224
```

To configure a dynamic ENAT entry to allow an entire private network with the IP address 192.168.100.0 to use the single officially assigned global IP address 202.1.1.1, use the command:

```
add ip nat ip=192.168.100.0 mask=255.255.255.0
    gblip=202.1.1.1
```

To add a static ENAT entry to the above example to allow access to a WWW server at IP address 192.168.100.54 on the private network, use the command:

```
add ip nat ip=192.168.100.54 prot=TCP po=80 gblip=202.1.1.1
    gblp=80
```

To re-map the WWW port on the server (192.168.100.54) to port 8001 in the private network, use the command:

```
add ip nat ip=192.168.100.54 prot=tcp po=8001 gblip=202.1.1.1
    gblp=80
```

To allow two different TELNET servers on the private network at IP addresses 192.168.100.54 and 192.168.100.92 to be accessed from the Internet, use the commands:

```
add ip nat ip=192.168.100.54 prot=tcp po=23 gblip=202.1.1.1
    gblp=23
add ip nat IP=192.168.100.92 prot=tcp po=23 gblip=202.1.1.2
    gblp=23
```

To add an interface ENAT entry to allow an entire private network with the IP address 10.1.1.0 to use a single dynamically assigned global IP addresses on interface ppp0, use the command:

```
add ip nat ip=10.1.1.0 mask=255.0.0.0 gblin=PPP0
```

**Related Commands**

**delete ip nat**
**disable ip nat**
**enable ip nat**
**show ip nat**

# add ip route

**Syntax**

```
ADD IP ROUte=ipadd INTerface=interface NEXThop=ipadd
   [CIRCuit=miox-circuit] [DLCi=dlci] [MASK=ipadd]
   [METric=1..16] [METRIC1=1..16] [METRIC2=1..65535]
   [POLIcy=0..7] [PREFerence=0..65535] [TAG=1..65535]
```

where:

- *ipadd* is an IP address in dotted decimal notation.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.
- *miox-circuit* is the name of a MIOX circuit defined for an X.25 interface 1 to 15 characters long. The name is not case-sensitive.
- *dlci* is the Data Link Connection Identifier (DLCI) of a Frame Relay DLC (circuit).

**Description**

This command adds a static route to the IP route table. Static routes can be used to define default routes to external routers or networks. A default route is one with a network address of 0.0.0.0. When the router receives data and cannot find a route for it, it sends the data to the default route. To define a default route, **route** is set to 0.0.0.0 and **nexthop** points to the network (router) where default packets are to be directed. The static route must not already exist. However, if the route exists as a dynamic route (such as RIP-derived), the static route can still be added. A recommended limit of 300 static routes applies to devices with 16MB of DRAM or less.

This command also defines subnets. Multiple routes can be defined for a single interface (usually a LAN). This is useful for configuring more than one network or subnet on a particular interface. A common problem is when hosts exceed the capacity of a single subnet. Additional subnets can be assigned by adding static routes. In this case **route** is set to the new subnet, **nexthop** is set to 0.0.0.0, and **metric** set to 1.

The **route** parameter specifies the IP address of the static route.

The **interface** parameter specifies the IP interface with which the route is associated. The interface must already exist and be assigned to the IP module. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces. If the interface is a Frame Relay interface, the **dlci** parameter is required and specifies the DLC to use on the Frame Relay interface. If the interface is an X.25 DTE interface, the **circuit** parameter is required and specifies the name of a MIOX circuit already defined for the X.25 DTE interface. If a logical interface is specified, the route is only added to the specified logical interface. If a logical interface is not specified, 0 is assumed and the route is only added to logical interface 0. The interface cannot be a local interface.

The **nexthop** parameter specifies the IP address of the next hop (router) for the route. The default is the IP address of the interface specified by the **interface** parameter. For a PPP link, **nexthop** should be the IP address of the remote end of the PPP link.

The **mask** parameter specifies the subnet mask for the route. The default mask is determined using the following:

1. If **mask** is specified, use the specified mask.
2. If the route is the default route, use a mask of 0.0.0.0.
3. If the route is for a network to which the router is not attached, use the unsubnetted mask for the network class (A, B or C).
4. Otherwise, use the subnet mask of the specified interface. The subnet mask does not need to be specified in most cases.

In all cases a check is performed on the route and mask to verify that the route is the same before and after masking. This ensures that a static route is not specified to more than its subnet mask.

The **metric1** parameter specifies the cost of traversing the route for RIP. The default is 1. The normal range is from 2 to 16. A metric of 1 should be used if adding a subnet to an interface. The **metric** parameter is also accepted for backwards compatibility.

The **metric2** parameter specifies the cost of traversing the route. The default is 1.

If static routes are redistributed into OSPF, **metric1** and **metric2** may be used as metric values for the redistributed routes. If **metric2** is configured, the route is redistributed into OSPF as a Type-2 route. For more information see the **add ospf redistribute** command on page 26-38 of Chapter 26, Open Shortest Path First (OSPF).

The **policy** parameter specifies the type of service for the route. The default is 0.

The **preference** parameter specifies the preference for the route. When more than one route in the route table matches the destination address in an IP packet, the route with the lowest preference value is used to route the packet. If two or more routes have the same preference, the route with the lowest metric is used. Interface routes have a preference of 0 and RIP routes have a preference of 100. The default preference for static routes other than 0.0.0.0 is 60. The default for the default static route 0.0.0.0 is 360.

The **tag** parameter specifies an integer to tag the route with. You can then match against this number in a route map and only import the appropriately-tagged routes into BGP or OSPF.

**Examples** To create a default route that points to a router at the remote end of a PPP link attached to interface ppp0 with the IP address 172.16.8.82, use the command:

```
add ip rou=0.0.0.0 mask=0.0.0.0 int=ppp0 next=172.16.8.82
   met=1
```

To add the subnet 172.16.9.0 to the existing routes on interface eth0, use the command:

```
add ip rou=172.16.9.0 mask=255.255.255.0 int=eth0
   next=172.16.8.82 met=1
```

Adding static routes to get more local address space can cause problems with PC-based TCP/IP software. You may need to change the subnet mask on the PC so that it recognises hosts on other subnets.

**Related Commands**

**delete ip route**
**set ip route**
**show ip route**

# add ip route template

**Syntax**

```
ADD IP ROUte TEMPlate=name INTerface=interface
   NEXThop=ipadd [CIRCuit=miox-circuit] [DLCi=dlci]
   [METric=1..16] [METRIC1=1..16] [METRIC2=1..65535]
   [POLIcy=0..7] [PREFerence=0..65535]
```

where:

- *name* is a string 1 to 31 characters long, and is not case-sensitive. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.
- *ipadd* is an IP address in dotted decimal notation.
- *miox-circuit* is the name of a MIOX circuit defined for an X.25 interface 1 to 15 characters long. The name is not case-sensitive.
- *dlci* is the Data Link Connection Identifier (DLCI) of a Frame Relay DLC (circuit).

**Description**

This command adds an IP route template. IP route templates are used by the router to add IP routes to IP subnetworks discovered during normal operation by other protocols, such as IPsec. This is required when IP traffic to the discovered IP subnetwork needs to be routed via a route other than the default route.

The **interface** parameter specifies the IP interface with which any route added using this template is associated. The interface must already exist and be assigned to the IP module. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces. If the interface is a Frame Relay interface, the **dlci** parameter is required and specifies the DLC to use on the Frame Relay interface. If the interface is an X.25 DTE interface, the **circuit** parameter is required and specifies the name of a MIOX circuit already defined for the X.25 DTE interface. If a logical interface is specified, the route is only added to the specified logical interface. If a logical interface is not specified, 0 is assumed and the route is only added to logical interface 0. The interface cannot be a local interface.

The **nexthop** parameter specifies the IP address of the next hop (router) for routes added with this template. The default is the IP address specified by the **interface** parameter. For a PPP link, **nexthop** should be the IP address of the remote end of the PPP link.

The **metric1** parameter specifies the cost of traversing routes added with this template for RIP. The default is 1. The normal range is from 2 to 16. A metric of 1 should be used when adding a subnet to an interface.The **metric** parameter is also accepted for backwards compatibility.

The **metric2** parameter specifies the cost of traversing any route added with this template for OSPF. The default is 1.

The **policy** parameter specifies the type of service for any route added using this template. The default is 0.

The **preference** parameter specifies the preference for routes added with this template. When more than one route in the route table matches the destination address in an IP packet, the route with the lowest preference value is used. If two or more routes have the same preference, the route with the longest subnet mask is used. Interface routes have a preference of 0 and RIP routes have a preference of 100. The default preference for static routes other than 0.0.0.0 is 60. The default for the default static route 0.0.0.0 is 360.

**Examples**

To add an IP route template named “branch_office”, use the command:

```
add ip rou temp=branch_office int=vlan1 next=192.168.23.3
```

**Related Commands**

**create ipsec policy**
**delete ip route template**
**set ip route template**
**show ip route template**

# add ip sa

**Syntax**

```
ADD IP SA=sa-id INTerface=interface
```

where:

- *sa-id* is a number from 0 to 100.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. If a logical interface is not specified, 0 is assumed.

**Description**

This command adds a security association to the list of security associations for an IP interface. IP SA commands provide support for RFCs 1825, 1827, and 1829, which have been superseded by IP Security. See Chapter 48, IP Security (IPsec) and RFCs 2401–2412 for more information about IPsec. This command requires a user with security officer privilege when the router is in security mode.

The **sa** parameter specifies the identifier of the security association. The security association must have been created previously using the **create sa** command on page 48-70 of Chapter 48, IP Security (IPsec).

The **interface** parameter specifies the name of the interface. The interface must already be assigned to the IP routing module. Valid interfaces are:

- eth (such as eth0, eth0-1)
- PPP (such as ppp0, ppp1-1)
- VLAN (such as vlan1, vlan1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces.

**Examples**

To add security association 3 to the IP interface ppp0, use the command:

```
add ip sa=3 int=ppp0
```

**Related Commands**

**create sa**
**delete ip sa**
**set ip interface**
**show ip interface**
**show ip sa**

# create ip pool

**Syntax**

```
CREate IP POOL=pool-name IP=ipadd[-ipadd]
```

where:

- *pool-name* is a string 1 to 15 characters long. Valid characters are any printable characters. If *pool-name* contains spaces, it must be in double quotes.
- *ipadd* is an IP address in dotted decimal notation.

**Description**

This command creates a pool of IP addresses that can be used by **ACC,** PPP, and other modules to assign IP addresses.

The **pool** parameter specifies a name for the IP address pool. The name is used in other commands to identify the pool.

The **ip** parameter specifies a range of IP addresses or a single one assigned to the pool. They should not overlap with IP address or ranges in other pools.

**Examples**

To create an IP pool named "dialin" with the IP addresses 192.168.1.1 to 192.168.1.16, use the command:

```
cre ip pool=dialin ip=192.168.1.1-192.168.1.16
```

**Related Commands**

**destroy ip pool**
**show ip pool**

# delete ip arp

**Syntax**

```
DELete IP ARP={ipadd|ALLDynamic}
```

where *ipadd* is an IP address in dotted decimal notation

**Description**

This command deletes dynamic or static ARP entries from the ARP cache.

To display the current contents of the router's ARP cache, use the **show ip arp** command on page 22-181.

The **arp** parameter specifies the entry to delete. If you specify an IP address, the router deletes the entry that corresponds to that address. If you specify **alldynamic**, the router deletes all dynamic ARP entries in the ARP cache.

**Examples**

To delete an ARP entry for a host with an IP address of 172.16.9.197, use the command:

```
del ip arp=172.16.9.197
```

**Related Commands**

**add ip arp**
**set ip arp**
**show ip arp**

# delete ip dns

**Syntax**

```
DELete IP DNS [DOMain={ANY|domain-name}]
```

where *domain-name* is a string of up to 255 characters. Valid characters are uppercase and lowercase letters, digits (0-9), and the underscore.

**Description**

This command deletes name server information from the DNS servers used by the router to resolve host names. When name server information is deleted, all DNS cache entries that were learned from those servers are removed from the cache.

The **domain** parameter specifies a domain name suffix for the name server configuration information to be deleted. If the **domain** parameter is not specified, the default DNS server configuration is deleted.

You cannot delete the default name server configuration while domain-specific name servers are configured.

**Examples**

To delete name server configuration information used for hosts in the domain “oranges.com”, use the command:

```
del ip dns dom=oranges.com
```

To delete the default name server configuration, use the command:

```
del ip dns
```

**Related Commands**

add ip dns
set ip dns
show ip dns

# delete ip filter

**Syntax**

```
DELete IP FILter=0..999 ENTry={entry-number|ALL}
```

where *entry-number* is the position of this entry in the filter

**Description**

This command deletes an existing pattern from an IP traffic filter, policy filter, or priority filter.

The **filter** parameter specifies the number of the filter where the pattern is to be deleted. Routing filters cannot be deleted if a BGP peer is using them.

The **entry** parameter specifies the entry number in the filter that is to be deleted. If **all** is specified, all entries in the filter are deleted. Existing patterns with the same or higher entry numbers are pushed up the filter to occupy the vacant entry.

**Examples**

To delete entry 3 from filter 2, use the command:

```
del ip fil=2 ent=3
```

**Related Commands**

**add ip filter**
**set ip filter**
**show ip filter**

# delete ip helper

**Syntax**

```
DELete IP HElper DEStination=ipadd INTerface=interface
   POrt=port-number
```

where:

- *ipadd* is an IP address in dotted decimal notation.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.
- *port-number* is a UDP port number from 1 to 65535, or one of the predefined UDP port names DNS (port 53), NT or NETBIOS (ports 137 and 138), TACACS (port 49), TIME (port 37) or TFTP (port 69).

**Description**

This command deletes either a port from the list of UDP ports to be forwarded or a destination IP address to which UDP broadcasts are being forwarded.

The **destination** parameter specifies the IP address to which the UDP broadcast traffic is forwarded.

The **interface** parameter specifies the interface to which the UDP port list is assigned. UDP broadcasts are forwarded that are received for the specified interface for one of the UDP ports in the UDP port list. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202.

The **port** parameter specifies the UDP port, as a decimal number from 1 to 65535, or the recognised name of a UDP port set. All broadcast traffic received by the router on the specified port or set of ports is redirected to the IP host at the destination address.

**Examples**

To stop forwarding all NETBIOS broadcasts received via interface eth0 to IP address 192.168.202.3, use the command:

```
del ip he po=netbios des=192.168.202.3 int=eth0
```

To stop forwarding all broadcasts to UDP port 3001 received via interface eth0 to IP address 192.168.100.2, use the command:

```
del ip he po=3001 int=eth0 des=192.168.100.2
```

**Related Commands**

**add ip helper**
**disable ip helper**
**enable ip helper**
**show ip helper**

# delete ip host

**Syntax**

```
DELete IP HOst=name
```

where *name* is a string up to 60 characters long. If the string contains spaces, it must be in double quotes.

**Description**

This command deletes a user-defined name for an IP host from the host name table. The host name table makes it easier to Telnet to commonly accessed hosts by enabling the user to enter a shorter, easier to remember name for the host rather than the host's full IP address or domain name.

The **host** parameter specifies the user-defined name to be deleted. The specified host name must exist in the host name table.

**Examples**

To delete the host name "zaphod" from the host name table, use the command:

```
del ip ho=Zaphod
```

**Related Commands**

**add ip host**
**set ip host**
**set ip nameserver**
**set ip secondarynameserver**
**show ip host**

# delete ip interface

**Syntax**

```
DELete IP INTerface=interface
```

where *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.

**Description**

This command deletes a logical interface from the IP module so that the logical interface is no longer used by the IP routing module.

The **interface** parameter specifies the name of the logical interface to be deleted. The interface must already be assigned to the IP routing module. At least two interfaces must be assigned to the IP module for the router to route IP packets, but only one interface (usually Ethernet) needs to be assigned when the router is acting as a server. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202. If you are using the GUI to view interfaces and have configured a large number, the page may take several minutes to display.

When an IP interface is deleted, static routes and ARP entries related to the interface are also deleted.

**Examples**

To delete PPP interface 2, use the command:

```
del ip int=ppp2
```

To delete the third logical interface attached to PPP0, use the command:

```
del ip int=ppp0-2
```

**Related Commands**

**add ip interface**
**disable ip interface**
**enable ip interface**
**reset ip interface**
**set ip interface**
**show ip interface**

# delete ip local

**Syntax**

```
DELete IP LOCal=1..15
```

**Description**

This command deletes a local interface from the IP module. The selected local interface will no longer be used by the IP routing module.

Note that when an IP interface is deleted, static routes and ARP entries specific to the interface are also deleted.

**Examples**

To delete local interface 5, use the command:

```
del ip local=5
```

**Related Commands**

add ip local
set ip local
show ip interface

# delete ip nat

**Syntax**

```
DELete IP NAT IP=ipadd MASK=ipadd GBLINterface=interface
    [GBLMask=ipadd] [GBLPort=port] [POrt=port]
    [PROTocol={protocol|ALL|EGP|GRE|ICmp|OSPF|SA|TCp|UDp}]

DELete IP NAT IP=ipadd GLBIPaddress=ipadd [MASK=ipadd]
    [GBLMask=ipadd] [GBLPort=port] [POrt=port]
    [PROTocol={protocol|ALL|EGP|GRE|ICmp|OSPF|SA|TCp|UDp}]
```

where:

- *ipadd* is an IP address in dotted decimal notation.
- *port* is an IP port number or the predefined name for an IP service.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.
- *protocol* is an IP protocol number.

**Description**

This command deletes a NAT or ENAT mapping. The first variant deletes an interface NAT. The **gblinterface** parameter is required and the **gblipaddress** parameter is not valid. The second variant deletes any other type of NAT. The **gblipaddress** parameter is required and the **gblinterface** parameter is not valid.

The **ip** parameter specifies either a host or network IP address for the private network. This parameter can be used with the **mask** parameter to specify a range of IP addresses for a private network.

The **protocol** parameter specifies the IP protocol number or the name of a predefined protocol type to be used with the static ENAT entry. If TCP or UDP is specified, then the **port** parameter must also be specified.

The **port** parameter specifies the port number or service name (Table 22-11 on page 22-88) for the port used on the private IP host when specifying a static ENAT entry.

The **gblipaddress** parameter specifies either an officially assigned global IP address or the start of a range of addresses. This parameter can be used with the **gblmask** parameter to specify the range of global IP addresses for an entry.

The **gblport** parameter specifies the port number or service name (Table 22-11 on page 22-88) for the port available to global Internet access when creating a static ENAT.

The **gblinterface** parameter specifies the interface that has or will dynamically obtain an officially assigned global IP address. Valid interfaces are:

- eth (such as eth0, eth0-1)
- PPP (such as ppp0, ppp1-1)
- VLAN (such as vlan1, vlan1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202.

**Examples** To delete a static NAT mapping between the private IP address 10.1.1.2 and the officially assigned global IP address 202.1.1.1, use the command:

```
del ip nat IP=10.1.1.2 gblip=202.1.1.1
```

To delete a static ENAT entry to allow access to a WWW server at IP address 192.168.100.54 on the private network, use the command:

```
del ip nat ip=192.168.100.54 prot=tcp po=80 gblip=202.1.1.1
   gblp=80
```

**Related Commands**

**add ip nat**
**disable ip nat**
**enable ip nat**
**show ip nat**

# delete ip route

**Syntax**

```
DELete IP ROUte=ipadd MASK=ipadd INTerface=interface
   NEXThop=ipadd
```

where:

- *ipadd* is an IP address in dotted decimal notation.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.

**Description**

This command deletes an existing static route from the IP route table. However, if the route exists as a dynamic route (such as RIP-derived), the static route may not be deleted. A maximum of 300 static routes can be defined.

The **route** parameter specifies the IP address of the static route.

The **interface** parameter specifies the IP interface with which the route is associated. The interface must already exist and be assigned to the IP module. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202.

The **nexthop** parameter specifies the IP address of the next hop (router) for the route. The default is the IP address of the interface specified by the **interface** parameter. For a PPP link, **nexthop** should be the IP address of the remote end of the PPP link.

The **mask** parameter specifies the subnet mask for the route. A check is performed on the route and mask to verify that the route is the same before and after masking. This ensures that a static route is not specified to more than its subnet mask.

**Examples**

To delete a default route that points to a router at the remote end of a PPP link attached to interface ppp0, with the IP address 172.16.8.82, use the command:

```
del ip rou=0.0.0.0 mask=0.0.0.0 int=pp0 next=172.16.8.82
```

**Related Commands**

**add ip route**
**set ip route**
**show ip route**

# delete ip route template

**Syntax**

```
DELete IP ROUte TEMPlate=name
```

where *name* is a string 1 to 31 characters long, and is not case-sensitive. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.

**Description**

This command deletes the specified IP route template.

**Examples**

To delete an IP route template named “branch_office”, use the command:

```
del ip rou temp=branch_office
```

**Related Commands**

**add ip route template**
**create ipsec policy**
**set ip route template**
**show ip route template**

# delete ip sa

**Syntax**

```
DELete IP SA=sa-id INTerface=interface
```

where:

- *sa-id* is a number from 0 to 100.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. If a logical interface is not specified, 0 is assumed.

**Description**

This command deletes a security association from the list of security associations for an IP interface. This command requires a user with security officer privilege when the router is in security mode

The **ip sa** commands provide support for RFCs 1825, 1827, and 1829, which have been superseded by IP Security. See Chapter 48, IP Security (IPsec) and RFCs 2401–2412 for more information about IPsec.

The **sa** parameter specifies the identifier of the security association. The security association must have been created previously using the **create sa** command on page 48-70 of Chapter 48, IP Security (IPsec), and must currently be assigned to the interface.

The **interface** parameter specifies the name of the interface. The interface must already be assigned to the IP routing module. Valid interfaces are:

- eth (such as eth0, eth0-1)
- PPP (such as ppp0, ppp1-1)
- VLAN (such as vlan1, vlan1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202.

**Examples**

To delete security association 3 from the IP interface ppp0, use the command:

```
del ip sa=3 int=ppp0
```

**Related Commands**

**add ip sa**
**create sa**
**set ip interface**
**show ip interface**
**show ip sa**

# delete tcp

**Syntax** `DELete TCP=`*`tcb`*

where *tcb* is the index of a TCP connection in the TCP connection table

**Description** This command deletes an active TCP session. The TCP parameter specifies the index in the TCP connection table of the TCP connection to be deleted. The index can be obtained from the output of the **show tcp** command. TCP sessions in the listen state cannot be deleted.

**Examples** To delete TCP session number 6, use the command:

```
del tcp=6
```

**Related Commands** show tcp

# destroy ip pool

**Syntax** `DESTroy IP POOL=`*`pool-name`*

where *pool-name* is a string 1 to 15 characters long. Valid characters are any printable characters. If *pool-name* contains spaces, it must be in double quotes.

**Description** This command destroys an existing pool of IP addresses. An IP address pool can be destroyed when there are no IP addresses in use. The **pool** parameter specifies the name of the IP address pool.

**Examples** To destroy the IP pool named "dialin", use the command:

```
dest ip pool=dialin
```

**Related Commands** create ip pool
show ip pool

# disable ddns

**Syntax** `DISable DDNS`

**Description** This command disables the Dynamic DNS feature.

**Examples** To disable the DDNS feature, use the command:

```
dis ddns
```

**Related Commands** activate ddns update
disable ddns debug
enable ddns
enable ddns debug
set ddns
show ddns

# disable ddns debug

**Syntax**

```
DISable DDNS DEbug
```

**Description**

This command disables the Dynamic DNS debug facility.

**Examples**

To disable DDNS debugging, use the command:

```
dis ddns de
```

**Related Commands**

**activate ddns update**
**disable ddns**
**enable ddns**
**enable ddns debug**
**set ddns**
**show ddns**

# disable ip

**Syntax**

```
DISable IP
```

**Description**

This command disables the IP routing module when it is enabled. The router no longer routes IP packets, responds to SNMP requests, uses TFTP to download software upgrades, or provides Telnet services. By default the IP module is disabled. The current operational mode of the IP module is retained so it can be restored when the IP module is enabled again.

The IP module operates in server mode or forwarding mode. In server mode, the router does not route IP packets, but provides Telnet services, responds to SNMP requests, and uses TFTP to download software upgrades. In forwarding mode, the router routes IP packets, as well as performing all the functions of server mode. The default is forwarding.

**Related Commands**

**disable ip forwarding**
**disable ip srcroute**
**enable ip**
**enable ip forwarding**
**enable ip srcroute**
**show ip**

# disable ip arp agepoll

**Syntax**

```
DISable IP ARP AGEPoll
```

**Description**

This command stops the router from checking whether devices are still attached to the network before deleting their dynamic ARP entries. Once a dynamic ARP entry has reached its timeout period, the router deletes the entry without sending an ARP request for the address. The agepoll feature is disabled by default.

**Related Commands**

**delete ip arp**
**enable ip arp agepoll**
**set ip arp refresharp**
**set ip arp timeout**
**show ip arp**

# disable ip arp log

**Syntax**

```
DISable IP ARP LOG
```

**Description**

This command disables logging of all MAC and IP addresses of all equipment connected to the router LAN interfaces accessing the WAN interface.

**Related Commands**

**enable ip arp log**

# disable ip debug

**Syntax**

```
DISable IP DEBug[={ARP|PACket|UPNP|ALL}]
```

**Description**

This command disables the IP debugging facility, or disables one or more currently-enabled debug options. The debugging facility is disabled by default.

**Related Commands**

**enable ip debug**
**show ip debug**
**show ip**
**disable debug active** in Chapter 4, Configuring and Monitoring the System
**show debug active** in Chapter 4, Configuring and Monitoring the System

# disable ip dnsrelay

**Syntax**

```
DISable IP DNSRelay
```

**Description** This command disables the DNS relay agent. The router stops forwarding DNS requests from hosts to the router's own configured DNS server. The DNS relay agent is disabled by default.

**Related Commands**

enable ip dnsrelay
set ip dnsrelay
show ip

# disable ip echoreply

**Syntax**

```
DISable IP ECHoreply
```

**Description** This command disables the generation of ICMP echo reply messages in response to ICMP echo request messages. Echo reply messages are enabled by default.

**Related Commands**

enable ip echoreply

# disable ip fofilter

**Syntax** `DISable IP FOFilter`

**Description** This command disables the filtering (discarding) of IP packets with a fragment offset of 1, and is intended for use in secure environments to prevent attacks by intruders using *tiny fragments* or *overlapping fragments* (see RFC 1858 for a detailed description). By default, the filter is enabled.

In the *tiny fragment* attack, the attacker transmits IP packets of the minimum fragment size. The first packet contains the IP header and 8 octets of data, which is insufficient to hold a complete TCP header. The TCP flags field is forced into the second fragment. Filters that attempt to discard connection requests (TCP datagrams with the SYN bit set and the ACK bit clear) are unable to test these flags in the first fragment and typically ignore them in subsequent fragments. As a result, the IP packet is not discarded. The fragment offset filter discards all fragments with a fragment offset of one, preventing reassembly of the packet at the receiving host.

In the *overlapping fragment* attack, the attacker transmits IP packets in fragments that overlap in an attempt to circumvent filters that discard connection requests. The first fragment contains a complete TCP header (so it avoids filters that discard fragments with a fragment offset of one) with the SYN bit clear and the ACK bit set (so it passes filters that discard connection requests). The second fragment has an offset of eight octets and contains another set of TCP flags, this time with the SYN bit set and the ACK bit clear. Typically, this fragment is passed by the filter, and at the receiving host the reassembly process results in the second fragment partially overwriting the first fragment.

The fragment offset filter discards all fragments with a fragment offset of one, preventing reassembly of the packet at the receiving host and effectively preventing both tiny fragment and overlapping fragment attacks.

If IP traffic filters have been created to drop connection requests (with the **session=start** parameter of the **add ip filter** command on page 22-70 or the **set ip filter** command on page 22-150), the fragment offset filter should be enabled to prevent tiny fragment and offset fragment attacks from circumventing the IP traffic filters.

**Related Commands**

**add ip filter**
**delete ip filter**
**enable ip fofilter**
**set ip filter**
**show ip filter**

# disable ip forwarding

**Syntax**

```
DISable IP FORwarding
```

**Description**

This command sets the IP module's operational mode to server, which disables the routing function. This flushes all dynamic routes, ARPs, and L3 table entries so that forwarding stops. The IP module must already be enabled and in forwarding mode.

The IP module operates in one of two modes: server or forwarding. In server mode, the router does not route IP packets, but provides Telnet services, responds to SNMP requests, and uses TFTP to download software upgrades. In forwarding mode, the router routes IP packets as well as performing all functions of the server mode. The default is forwarding.

**Related Commands**

**disable ip**
**disable ip srcroute**
**enable ip**
**enable ip forwarding**
**enable ip srcroute**
**show ip**

# disable ip helper

**Syntax**

```
DISable IP HElper
```

**Description**

This command disables the forwarding of broadcast UDP traffic on specific UDP ports to specific destination IP addresses.

**Examples**

To disable broadcast forwarding, use the command:

```
dis ip he
```

**Related Commands**

**add ip helper**
**delete ip helper**
**enable ip helper**
**show ip helper**

# disable ip icmpreply

**Syntax**

```
DISable IP
   ICMPreply[={ALL|NETunreach|HOSTunreach|REDirect}]
```

**Description** This command disables ICMP reply messages.

If **all** is specified, all configurable ICMP message replies are disabled.

If **netunreach** is specified, all network unreachable message replies are disabled (RFC 792 Type 3 Code 0).

If **hostunreach** is specified, all host unreachable message replies are disabled (RFC 792 Type 3 Code 1).

If **redirect** is specified, all ICMP redirect message replies are disabled (RFC 792 Type 5 Code 0, 1, 2, 3).

**Example** To disable all configurable ICMP messages, use the command:

```
dis ip icmp=all
```

**Related Commands** enable ip icmpreply
show ip icmpreply

# disable ip interface

**Syntax**

```
DISable IP INTerface=interface
```

where *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.

**Description**

This command temporarily disables a logical IP interface. The logical interface is not used by the IP routing module. The effect is equivalent to physically disconnecting the router from the attached IP network. Routes associated with a disabled interface are not explicitly deleted. However, routes learned via a routing protocol such as RIP are eventually deleted by the routing protocol's aging mechanism. Static routes are retained until explicitly removed by deleting the specific static route entry or by deleting the IP interface.

The **interface** parameter specifies the name of the logical interface to be disabled. The interface must be assigned to the IP routing module and currently enabled. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202. If you are using the GUI to view interfaces and have configured a large number, the page may take several minutes to display.

**Examples**

To disable the first logical IP interface attached to PPP0, use the command:

```
dis ip int=ppp0-0
```

**Related Commands**

**add ip interface**
**delete ip interface**
**enable ip interface**
**reset ip interface**
**set ip interface**
**show ip interface**

# disable ip macdisparity

**Syntax**

```
DISable IP MACdisparity
```

**Description** This command stops ARP entries from being configured with discrepancies in their address. When disabled, the router does not allow an ARP entry with a multicast MAC address to be added, and the routerdiscards packets received with address discrepancies. This feature is disabled by default.

**Examples** To ensure that ARP entries with unicast IP addresses do not get assigned a multicast MAC address, use the command:

```
dis ip mac
```

**Related Commands**

**add ip arp**
**enable ip macdisparity**
**show ip**

# disable ip nat

**Syntax**

```
DISable IP NAT [FRAgment={ICMP|UDP|OTHER}[,...]]
    [LOG={ALL|FAILS|INTCP|INUDP|OUTTCP|OUTUDP}[,...]]
```

**Description** This command disables NAT, disables enhanced packet fragment handling when NAT is in use, or disables the logging of a specific class of NAT events.

The **fragment** parameter specifies that IP packets for a specific protocol be dropped when NAT is used when they have been fragmented into more than 8 fragments, or have a total payload of more than 1780 bytes of protocol data. This restores the default functionality where fragmented packets are permitted when there are no more than 8 fragments and the combined payload consists of a maximum of 1780 bytes. There is no default.

The **log** parameter specifies the class of NAT event to log. The **fails** option logs IP traffic received by the global interface of the router that could not be delivered because a service had not been specified on the private network. The **intcp** option logs TCP session opens to servers on the private network. The **inudp** option logs UDP flows initiated to a server on the private network. The **outtcp** option logs TCP sessions originating on the private network destined for the Internet. The **outudp** option logs UDP flows originating on the private network destined for the Internet. Messages are logged by the router's Logging facility. Logging multiple classes of events can be disabled by entering a comma-separated list of event classes.

NAT is automatically disabled when the firewall is enabled because the firewall provides NAT services. However, the NAT configuration is retained so that it can be manually enabled again if the firewall is disabled.

**Examples** To disable NAT, use the command:

```
dis ip nat
```

To disable the logging of inward and outward TCP connections to a server, use the command:

```
dis ip nat log=intcp,outtcp
```

**Related Commands** add ip nat
delete ip nat
enable ip nat
show ip nat

# disable ip remoteassign

**Syntax**

```
DISable IP REMoteassign
```

**Description**

This command disables the remote assignment of IP addresses for unnumbered PPP interfaces. The router does not allow a remote PPP peer to set the IP address of the local PPP interface

**Examples**

To disable remote IP address assignment, use the command:

```
dis ip rem
```

**Related Commands**

enable ip remoteassign
show ip

# disable ip route

**Syntax**

```
DISable IP ROUte [CAChe|COUnt|MULtipath|DEBug]
```

**Description**

This command disables route caching, route counters, or equal cost multipath routing.

The **cache** parameter disables route caching. The cache is enabled by default.

The **count** parameter disables counting of octets sent and received to and from a network. It is disabled by default.

The **multipath** parameter disables equal cost multipath routing (ECMP). The router still learns multiple routes, but only forwards packets over the best route to the destination. ECMP is enabled by default.

The **debug** parameter disables the IP route debugging facility.

**Examples**

To disable equal cost multipath routing, use the command:

```
dis ip rou mul
```

**Related Commands**

enable ip route
show ip route

# disable ip spoofcheck

**Syntax**

```
DISable IP SPoofcheck
```

**Description**

This command stops the router from checking for packets with spoofed IP addresses. The router determines that a packet has a spoofed IP address if its source address matches the address of a local interface.

**Related Commands**

enable ip spoofcheck
show ip

# disable ip srcroute

**Syntax**

```
DISable IP SRCRoute[={LOOSE|STrict|ALL}]
```

**Description**

This command disables the forwarding of source-routed IP packets. If a specific type of source-routed IP packet is entered, as defined in RFC 791, forwarding of that type is disabled. Otherwise, forwarding of all source-routed IP packets is disabled.

When forwarding is enabled, source-routed IP packets are forwarded by the router as normal. When forwarding is disabled, source-routed packets are discarded by the router. The default is to disable forwarding.

Source routing is rarely used for legitimate purposes, and is a common method used to circumvent packet-filtering firewalls and to masquerade as a trusted host inside the destination network. This command is therefore an extra security feature, which is why source routed packets are discarded by default.

When forwarding IP source routed datagrams is disabled, all source routed packets are logged by the Logging facility with a message type/subtype of IPFIL/SRCRT.

**Related Commands**

disable ip
enable ip
enable ip forwarding
enable ip srcroute
show ip

# disable tcp debug

**Syntax** `DISable TCP DEBug={ALL|EVent|IN|MAIn|OUT|STAte}]`

**Description** This command enables the TCP debug facility.

| Parameter | Description | |
|---|---|---|
| DEBug | The type of TCP debugging mode you are disabling.<br>Default: no default | |
| | ALL | Disables all TCP debugging modes. |
| | EVent | Disables debugging of the events occurring to the TCP state machine. |
| | IN | Disables debugging of the TCP packets that the router is receiving. |
| | OUT | Disables debugging of the TCP packets that the router is sending. |
| | STAte | Disables debugging of the state changes occurring to the TCP state machine. |

**Examples** To disable all TCP debugging modes, use the command:

```
dis tcp deb=all
```

**Related Commands** **enable tcp debug**
**show debug active** in Chapter 4, Configuring and Monitoring the System
**show tcp**

# disable telnet server

**Syntax** `DISable TELnet SErver [LOGINBAnner]`

**Description** This command blocks telnet access to the router or disables the login banner. The telnet access and the login banner are enabled by default. For security reasons, you have the option to disable telnet access to the router.

The **loginbanner** parameter disables the telnet server from displaying the system name when you initiate a telnet session.

**Example** To disable telnet access to the router, use the command:

```
dis tel se
```

. **Related Commands** **enable telnet server**
**set telnet**
**show telnet**
**telnet**

# enable ddns

**Syntax** `ENAble DDNS`

**Description** This command enables the Dynamic DNS feature.

**Examples** To enable the DDNS feature, use the command:

```
ena ddns
```

**Related Commands**

activate ddns update
disable ddns
disable ddns debug
enable ddns debug
set ddns
show ddns

# enable ddns debug

**Syntax** `ENAble DDNS DEBug`

**Description** This command enables the Dynamic DNS debug facility.

**Examples** To enable DDNS debugging, use the command:

```
ena ddns deb
```

**Related Commands**

activate ddns update
disable ddns
disable ddns debug
enable ddns
set ddns
show ddns

# enable ip

**Syntax**

```
ENAble IP
```

**Description**

This command enables the IP routing module when it has been disabled. It requires a user with security officer privilege when the router is in security mode. The IP module is disabled by default.

The IP module operates in server mode or forwarding mode, and the operational mode is restored from when the IP module was last disabled. In server mode, the router does not route IP packets, but provides Telnet services, responds to SNMP requests, and uses TFTP to download software upgrades. In forwarding mode, the router routes IP packets, as well as performing all the functions of server mode. The default is forwarding.

**Related Commands**

disable ip
disable ip forwarding
disable ip srcroute
enable ip forwarding
enable ip srcroute
show ip

# enable ip arp agepoll

**Syntax**

```
ENAble IP ARP AGEPoll
```

**Description**

This command enables the router to check whether devices are still attached to the network before deleting their dynamic ARP entries. Once a dynamic ARP entry has reached its timeout period, the router sends an ARP request for the address. If the router receives a response, then it resets the timeout period for the dynamic ARP entry. If the router does not receive a response, then it deletes the dynamic ARP entry. The agepoll feature is disabled by default.

**Related Commands**

delete ip arp
disable ip arp agepoll
set ip arp refresharp
set ip arp timeout
show ip arp

# enable ip arp log

**Syntax**

```
ENAble IP ARP LOG
```

**Description**

This command enables logging of all MAC and IP addresses of all equipment connected to the router's LAN interfaces accessing the WAN interface. This occurs at the time these addresses are added to or deleted from the router ARP table.

**Related Commands**

disable ip arp log

# enable ip debug

**Syntax**

```
ENAble IP DEBug

ENAble IP DEBug={ARP|PACket|UPNP|ALL}
[TIMEOut={NONE|1..2400}]
```

**Description** This command enables the IP debugging facility, which is disabled by default. It is available in two modes:

- storing incorrectly formatted packets in a buffer

  To do this, enter **enable ip debug** without specifying any other options. The router stores the header and the reason for rejection of up to 40 incorrectly formatted IP packets. Once the buffer is full, the router discards the oldest packets. To display the stored information, use the **show ip debug** command on page 22-192. Note that the **timeout** parameter has no effect in this mode.

- displaying information about IP packets on screen

  To do this, enter **enable ip debug={arp | packet | upnp | all}**. You can also limit the length of time for which information displays by using the **timeout** parameter.

The **debug** parameter specifies the debugging option to enable. The following table lists the available debugging options.

| Option | Description |
|---|---|
| ARP | Displays summaries of ARP requests and replies along with any changes to the ARP table. |
| PACKet | Displays summaries of all packet headers coming in and going out of IP interfaces. |
| UPNP | Displays a trace of the internal UPnP functions as they execute. |
| ALL | Enables all debugging options. |

The **timeout** parameter specifies the time period, in seconds, for which IP debugging is enabled. Setting a timeout reduces the risk of overloading the router and the display with too much debugging information. The value set by the **timeout** parameter overrides any previous IP debugging timeout values, even if they were specified for other debugging options. The default is the timeout value used the last time that this command was run, or **none**.

Note that the **timeout** parameter only applies if you specify an option on the **debug** parameter.

To change the current timeout value, re-enter the command **enable ip debug={arp | packet | upnp | all} timeout={none | 1..2400}**. A value of **none** turns the timeout off.

To see the current timeout value, use the **show debug active[=ip]** command. The current timeout is shown above the types of IP debugging that are currently enabled.

**Example** To display ARP debugging information on screen for the next 25 seconds, use the command:

```
enable ip debug=arp timeout=25
```

To enable all debug options indefinitely, use the command:

```
enable ip debug=all timeout=none
```

**Related Commands**

**disable ip debug**
**show ip debug**
**show ip**
**disable debug active** in Chapter 4, Configuring and Monitoring the System
**show debug active** in Chapter 4, Configuring and Monitoring the System

# enable ip dnsrelay

**Syntax** `ENAble IP DNSRelay`

**Description** This command enables the DNS relay agent so that the router forwards DNS requests from hosts to the router’s own configured DNS server. The DNS relay agent is disabled by default.

**Related Commands**

**disable ip dnsrelay**
**set ip dnsrelay**
**show ip**

# enable ip echoreply

**Syntax** `ENAble IP ECHoreply`

**Description** This command enables the generation of ICMP echo reply messages in response to ICMP echo request messages. You cannot ping the router if ICMP echo reply messages are disabled. Echo reply messages are enabled by default.

**Related Commands**

**disable ip echoreply**

# enable ip fofilter

**Syntax**

```
ENAble IP FOFilter
```

**Description**

This command enables the filtering (discarding) of IP packets with a fragment offset of 1, and is intended for use in secure environments to prevent attacks by intruders using *tiny fragments* or *overlapping fragments* (see RFC 1858 for a detailed description). By default, the filter is enabled.

In the *tiny fragment* attack, the attacker transmits IP packets of the minimum fragment size. The first packet contains the IP header and 8 octets of data, which is insufficient to hold a complete TCP header. The TCP flags field is forced into the second fragment. Filters that attempt to discard connection requests (TCP datagrams with the SYN bit set and the ACK bit clear) are unable to test these flags in the first fragment and typically ignore them in subsequent fragments. As a result, the IP packet is not discarded. The fragment offset filter discards all fragments with a fragment offset of one, preventing reassembly of the packet at the receiving host.

In the *overlapping fragment* attack, the attacker transmits IP packets in fragments that overlap in an attempt to circumvent filters that discard connection requests. The first fragment contains a complete TCP header (so it avoids filters that discard fragments with a fragment offset of one) with the SYN bit clear and the ACK bit set (so it passes filters that discard connection requests). The second fragment has an offset of eight octets and contains another set of TCP flags, this time with the SYN bit set and the ACK bit clear. Typically, this fragment is passed by the filter, and at the receiving host the reassembly process results in the second fragment partially overwriting the first fragment.

The fragment offset filter discards all fragments with a fragment offset of one, preventing reassembly of the packet at the receiving host and effectively preventing both tiny fragment and overlapping fragment attacks.

IP datagrams discarded by the fragment offset filter are logged by the Logging facility with a message type/subtype of IPFIL/FRAG.

If IP traffic filters have been created to drop connection requests (using the **session=start** parameter of the **add ip filter** command on page 22-70 or the **set ip filter** command on page 22-150), the fragment offset filter should be enabled to prevent tiny fragment and offset fragment attacks from circumventing the IP traffic filters.

**Related Commands**

**add ip filter**
**delete ip filter**
**disable ip fofilter**
**set ip filter**
**show ip filter**

# enable ip forwarding

**Syntax**

```
ENAble IP FORwarding
```

**Description**

This command sets the IP module's operational mode to a routing function. The IP module must already be enabled and in server mode.

The IP module operates in one of two modes: server or forwarding. In server mode, the router does not route IP packets, but provides Telnet services, responds to SNMP requests, and uses TFTP to download software upgrades. In forwarding mode, the router routes IP packets as well as performing all functions of the server mode. The default is forwarding.

**Related Commands**

disable ip
disable ip forwarding
disable ip srcroute
enable ip
enable ip srcroute
show ip

# enable ip helper

**Syntax**

```
ENAble IP HElper
```

**Description**

This command enables the forwarding of broadcast UDP traffic on specified UDP ports to specified destination IP addresses.

**Examples**

To enable broadcast forwarding, use the command:

```
ena ip he
```

**Related Commands**

add ip helper
delete ip helper
disable ip helper
show ip helper

# enable ip icmpreply

**Syntax**

```
ENAble IP
   ICMPreply[={ALL|NETunreach|HOSTunreach|REDirect}]
```

**Description**

This command enables ICMP reply messages.

If **all** is specified, all configurable ICMP message replies are enabled. If **netunreach** is specified, all network unreachable message replies are enabled (RFC 792 Type 3 Code 0). If **hostunreach** is specified, all host unreachable message replies are enabled (RFC 792 Type 3 Code 1). If **redirect** is specified, all ICMP redirect message replies are enabled (RFC 792 Type 5 Code 0, 1, 2, 3).

**Example**

To enable all configurable ICMP messages, use the command:

```
ena ip icmp=all
```

**Related Commands**

**disable ip icmpreply**
**show ip icmpreply**

# enable ip interface

**Syntax**

```
ENAble IP INTerface=interface
```

where *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.

**Description**

This command enables a logical IP interface for use by the IP routing module.

The **interface** parameter specifies the name of the logical interface to be enabled. The interface must be assigned to the IP routing module and currently disabled. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202. If you are using the GUI to view interfaces and have configured a large number, the page may take several minutes to display.

**Examples**

To enable the first logical IP interface attached to PPP0, use the command:

```
ena ip int=ppp0-0
```

**Related Commands**

**add ip interface**
**delete ip interface**
**disable ip interface**
**reset ip interface**
**set ip interface**
**show ip interface**

# enable ip macdisparity

**Syntax**

```
ENAble IP MACdisparity
```

**Description**

This command allows you to add static ARP entries with multicast MAC addresses, and allows packets with conflicting IP and MAC addresses to pass through the router. Normally these packets are discarded as being invalid by the router.

Conflicting IP and MAC addresses include the following:

- A multicast IP address with a unicast MAC address
- A unicast IP address with a multicast MAC address

ARP entries cannot be configured with multicast MAC addresses unless MACdisparity is enabled using this command.

This feature is disabled by default. When disabled, you can only add ARP entries with unicast MAC addresses, and the router discards packets with conflicting IP and MAC addresses.

Routers further downstream may not accept unicast IP addresses with multicast MAC addresses.

**Examples**

To allow static entries with multicast MAC addresses to be configured on the router, use the command:

```
ena ip mac
```

**Related Commands**

add ip arp
disable ip macdisparity
show ip

# enable ip nat

**Syntax**

```
ENAble IP NAT [FRAgment={ICMP|UDP|OTHER}[,...]]
    [LOG={ALL|FAILS|INTCP|INUDP|OUTTCP|OUTUDP}[,...]]
```

**Description**

This command enables NAT, enables enhanced packet fragment handling when NAT is in use, or enables the logging of a class of specific NAT events.

The **fragment** parameter specifies that IP packets for a specific protocol be permitted when NAT is used when they have been fragmented into more than 8 fragments. There is no limit on total data within this number of fragments other than the MTU restrictions of the interfaces involved in forwarding the packets. If **icmp** is specified, IP NAT permits ICMP ping (echo) requests and replies that have been fragmented into more than 8 fragments. The default behaviour is that fragmented packets are permitted when there are no more than 8 fragments and the combined protocol data consists of a maximum of 1780 bytes. The number of fragments that can be handled is configured by the **set ip nat maxfragments** command on page 22-163.

The **log** parameter specifies the class of NAT event to log. The **fails** option logs IP traffic received by the global interface of the router that could not be delivered because a service had not been specified on the private network. The **intcp** option logs TCP session opens to servers on the private network. The **inudp** option logs all UDP flows initiated to a server on the private network. The **outtcp** option logs TCP sessions originating on the private network destined for the Internet. The **outudp** option logs UDP flows originating on the private network destined for the Internet. Messages are logged by the router's Logging facility. Logging multiple classes of events can be disabled by entering a comma-separated list of event classes.

NAT is automatically disabled when the firewall is enabled because the firewall provides NAT services. However, the NAT configuration is retained so that it can be manually enabled again if the firewall is disabled.

**Examples**

To enable NAT, use the command:

```
ena ip nat
```

To enable the logging of inward and outward TCP connections to a server, use the command:

```
ena ip nat log=intcp,outtcp
```

**Related Commands**

**add ip nat**
**delete ip nat**
**disable ip nat**
**show ip nat**

# enable ip remoteassign

**Syntax**

```
ENAble IP REMoteassign
```

**Description** This command enables the remote assignment of IP addresses for unnumbered PPP interfaces.

If a PPP interface is created with an IP address of 0.0.0.0, and remote IP address assignment is enabled, then during the IP control protocol (IPCP) negotiation process the router allows the remote PPP peer to set the IP address of the local PPP interface. If the local PPP interface has an IP number other than 0.0.0.0, or if remote IP address assignment is disabled, the router does not allow the remote PPP peer to set the IP address of the local PPP interface.

**Examples** To enable remote IP addresses to be assigned, use the command:

```
ena ip rem
```

**Related Commands**

**disable ip remoteassign**

**show ip**

# enable ip route

**Syntax**

```
ENAble IP ROUte [CAChe|COUnt|MULtipath|DEBug]
```

**Description** This command enables route caching, route counters, or equal cost multipath routing. Route caching is enabled by default.

The **cache** parameter enables route caching. The cache is enabled by default.

The **count** parameter enables counting of octets sent and received to and from a network. It is disabled by default.

The **multipath** parameter enables equal cost multipath routing (ECMP). If the router learns multiple routes of the same cost to a destination, it distributes the packets across all the routes. You can have up to 16 individual routes to a destination. For more information see “Equal Cost Multipath Routing” on page 22-24. ECMP routing is enabled by default.

The **debug** parameter enables the IP route debugging facility. Incorrectly formatted IP packet headers are captured for later analysis.

**Examples** To enable byte counting for routes, use the command:

```
ena ip rou cou
```

**Related Commands**

**disable ip route**

**show ip route**

# enable ip spoofcheck

**Syntax**

```
ENAble IP SPoofcheck
```

**Description**

This command enables the router to check for and discard any packets with spoofed IP addresses. The router determines that a packet has a spoofed IP address if its source address matches the address of a local interface. IP spoof checking only applies to packets routed by the CPU. This feature is enabled by default.

**Related Commands**

**disable ip spoofcheck**
**show ip**

# enable ip srcroute

**Syntax**

```
ENAble IP SRCRoute[={LOOSE|STrict|ALL}]
```

**Description**

This command enables the forwarding of source-routed IP packets. If a specific type of source-routed IP packet is specified, as defined in RFC 791, forwarding of that type will be enabled. Otherwise, forwarding of all source-routed IP packets will be enabled.

When forwarding is enabled, source-routed IP packets are forwarded by the router as normal. When forwarding is disabled, source-routed packets are discarded. The default is to disable forwarding.

Source routing is rarely used for legitimate purposes, and is a common method used to circumvent packet-filtering firewalls and to masquerade as a trusted host inside the destination network. This command is therefore an extra security feature, which is why source routed packets are discarded by default.

**Related Commands**

**disable ip**
**disable ip srcroute**
**enable ip**
**enable ip forwarding**
**show ip**

# enable tcp debug

**Syntax**

```
ENAble TCP DEBug={ALL|EVent|IN|MAIn|OUT|STAte}]
```

**Description** This command enables the TCP debug facility.

| Parameter | Description | |
|---|---|---|
| DEBug | The type of TCP debugging mode you are enabling.<br>Default: no default | |
| | ALL | Enables all TCP debugging modes. |
| | EVent | Enables debugging of the events occurring to the TCP state machine. |
| | IN | Enables debugging of the TCP packets that the router is receiving. |
| | OUT | Enables debugging of the TCP packets that the router is sending. |
| | STAte | Enables debugging of the state changes occurring to the TCP state machine. |

**Examples** To enable debugging of the input packets TCP receives, use the command:

```
ena tcp deb=in
```

**Related Commands**

**disable tcp debug**
**show debug active** in Chapter 4, Configuring and Monitoring the System
**show tcp**

# enable telnet server

**Syntax**

```
ENAble TELnet SErver [LOGINBAnner]
```

**Description** This command enables remote users to telnet to the router or enables the **loginbanner**. Telnet access and the login banner are enabled by default. For security reasons, you have the option to disable telnet access to the router.

**Examples** To enable telnet access to the router, use the command:

```
ena tel se
```

**Related Commands**

**disable telnet server**
**set telnet**
**show telnet**
**telnet**

# finger

**Syntax** FINGer [*username*]@*host*[@*host*]... [DETail={HIgh|LOW}]

where:

- *username* is the account name from 1 to 20 characters long to be fingered. Valid characters are uppercase and lowercase letters, digits (0–9), and the underscore. Wildcards are not allowed.
- *host* is the finger server to be queried, an IP address in dotted decimal notation or a host name from the host name table.

**Description** This commands sends a finger query to the finger server on the specified host or hosts. The response from the finger server is sent to the terminal or telnet session from which the command was entered. When more than one host is given, finger forwarding is attempted, and each host refers to each step in the chain of finger servers.

The **detail** parameter specifies the level or details to request from the finger server. The finger server either interprets the command or ignores it, depending on whether the server supports it. The format of the information varies from server to server. The default is **low**.

Figure 22-11 shows a typical response from a finger query to a remote host requesting information about a specific user. A plan is a type of file that is appended to a finger response, and acts in a similar manner as signature files in email messages.

Figure 22-11: Example output of the response to a finger query.

```
> finger root@admin
Login: root                                 Name: root
Directory: /root                           Shell: /bin/tcsh
Last login Wed Jul 14 12:12 (NZDT) on ttyp0 from 192.168.11.17
New mail received Wed Jul 14 01:02 1999 (NZDT)
     Unread since Tue Jun  1 12:23 1999 (NZDT)
No Plan.
```

**Examples** To send a finger query to the host `admin` requesting detailed information about user “root”, use the command:

```
fing root@admin det=hi
```

To send a finger query to the host `admin` (at IP address 192.168.11.5) requesting a list of all logged in users, use either of the commands:

```
fing @admin
```

```
fing @192.168.11.5
```

# ping

**Syntax**

```
PING
   [IPaddress=]{ipadd|ipv6add[%interface]|host|domain-name
   } [DElay=seconds] [Length=number]
   [NUMber={number|CONTinuous}] [PATTern=hexnum]
   [SIPAddress={ipadd|ipv6add}]
   [SCReenoutput={OFf|ON|NO|YES}] [TIMEOut=1..60]
   [TOS=0..255]

PING [IPXAddress=]network:station [DElay=seconds]
   [LENGTH=number] [NUMBER={number|CONTinuous}]
   [PATTern=hexnum] [SIPXaddress=network:station]
   [SCReenoutput={OFf|ON|NO|YES}] [TIMEOut=1..60]

PING [APPLEAddress=]network.node [DElay=seconds]
   [Length=number] [NUMber={number|CONTinuous}]
   [PATTern=hexnum] [SAPpleaddress=network.node]
   [SCReenoutput={OFf|ON|NO|YES}] [TIMEOut=1..60]

PING [OSIAddress=]nsap [DElay=seconds] [Length=number]
   [NUMber={number|CONTinuous}] [PATTern=hexnum]
   [SOSIaddress=nsap] [SCReenoutput={OFf|ON|NO|YES}]
   [TIMEOut=1..60]
```

where:

- *ipadd* is an IPv4 address in dotted decimal notation.
- *ipv6add* is a valid IPv6 address.
- *interface* is the interface the ping request is sent for a request to ping an IPv6 link-local address, e.g. vlan1. Separate the address and interface with a % sign.
- *host* is a host name from the host name table.
- *domain-name* is a valid domain name.
- *network:station* is a valid Novell network number and station MAC address, expressed as hexadecimal numbers. Leading zeros may be omitted.
- *network.node* is an AppleTalk network number from 0 to 65279 or an AppleTalk network number range in the format “nnnnn-nnnnn”, and an AppleTalk node number from 0 to 127.
- *nsap* is a valid OSI NSAP address in dotted hexadecimal notation.
- *seconds* is a decimal number from 0 to 4294967295.
- *hexnum* is an 8-digit hexadecimal number, optionally proceeded by the characters “0x”.

**Description** This command can be used to test that a valid path (route) exists to a destination. Packets are sent to the address specified; if the destination host replies, the time taken for the response to be received is recorded, and optionally displayed. The parameters of this command override the defaults set with the **set ping** command on page 22-171.

The router’s extended **ping** command supports IPv4, IPv6, OSI, IPX, and AppleTalk protocols, using the protocol’s native echo request message.

Pinging an IPv6 link-local address requires interface information as well as the address because a single link-local address can belong to several interfaces. To ping a link-local address, specify the interface out which the router is to send the ping request, as well as the address to which the router is to send the ping request (Figure 31-2 on page 31-19 in Chapter 31, Internet Protocol version 6 (IPv6)). For example:

```
ping fe80::7c27%vlan1
```

The **ipaddress**, **ipxaddress**, **osiaddress**, and **appleaddress** parameters specify the destination address for ping packets for IP, IPX, OSI, and AppleTalk networks, respectively.

You can specify a valid IP address, a host name defined in the host name table, or a domain name as the destination IP address. To add a host to the host name table, use the **add ip host** command on page 22-79. To configure a DNS for the router to use to resolve domain names, use the **add ip dns** command on page 22-68.

The **delay** parameter specifies the time interval, in seconds, between ping packets. If **delay** is not specified, the default is used.

The **length** parameter specifies the number of data bytes of the specified pattern to include in the data portion of the ping packet. If **length** is not specified, the default is used.

The **number** parameter specifies the number of ping packets to transmit. If **number** is not specified, the default is used. If **continuous** is specified, the **timeout** parameter must be set to a value greater than 0, and packets are sent continuously until the **stop ping** command on page 22-233 is issued.

The **pattern** parameter specifies the data to use to fill the data portion of the ping packet. If **pattern** is not specified, the default is used.

The **sipaddress**, **sipxaddress**, **sosiaddress**, and **sappleaddress** parameters specify the source address to use in ping packets for IP, IPX, OSI, and AppleTalk networks, respectively.

If the source address is not specified, and has not been set with the **set ping** command on page 22-171, the default is to use the address of the interface from which the ping packets are transmitted. For IP addresses, the router's local interface IP address is used, if set. Otherwise, the IP address of the interface from which the ping packets are transmitted is used. If the ping request is to an IPv6 link-local address, **sipaddress** must be on the outgoing interface and cannot be a link-local address.

The **screenoutput** parameter specifies whether the output is sent to the terminal. If **yes** is specified, the response time for each echo reply packet is displayed to the terminal as the reply is received from the destination host (Figure 22-12 on page 22-137).

Figure 22-12: Example output from the **ping** command when **screenoutput** is **yes**

```
Echo reply 1 from 172.16.8.2 time delay 20 ms

Echo reply 2 from 172.16.8.2 time delay 40 ms

Echo reply 3 from 172.16.8.2 time delay 0 ms

Echo reply 4 from 172.16.8.2 time delay 0 ms

Echo reply 5 from 172.16.8.2 time delay 60 ms
```

If **no** is specified, the results are stored and not displayed. To view the results, use the **show ping** command on page 22-224. If **screenoutput** is not specified, the default is used.

The **timeout** parameter specifies the length of time in seconds to wait for a response to a ping packet, and cannot be zero. If **timeout** is not specified, the default is used.

The **tos** parameter specifies the value of the TOS (Type Of Service) field in the IP header of the ping packet. This parameter is valid for IP addresses and is ignored for IPv6 addresses. If **tos** is not specified, the default is used.

**Related Commands**

**add ip host**
**set ping**
**show ping**
**stop ping**

# purge ip

**Syntax** `PURge IP`

**Description** This command purges all configuration information relating to the IP routing module, and reinitialises the data structures used by the IP module. It should be used when first setting up the IP module or when a major change is required.

**Caution** All current configuration information will be lost. Use with extreme caution!

Minor changes, such as changing the IP address of an interface, can be done without using the **purge ip** command. The configuration information is kept in non-volatile storage so that information is retained after a power down.

**Related Commands** **reset ip**

# reset ip

**Syntax**

```
RESET IP
```

**Description**

This command reinitialises dynamic IP data structures. It restarts routing timers and clears the ARP cache and IP cache, the route table, and IP counters. It only affects the IP module; protocols that operate with and use IP (such as OSPF and BGP-4) are not affected. It does not make the router operational if incorrect or incomplete information has been entered, for example, if no IP address has been assigned to an interface. The command also sends a message to the Logging facility if one has been configured. Note that all dial-in SLIP/PPP connections are disconnected when this command is executed.

This command is not typically necessary during normal router operations. However, some occasions require that the IP module be restarted. For example, when an underlying interface (such as a PPP interface) has changed, the IP module must be restarted to discover changes.

**Related Commands**

**purge ip**
**reset ip counter**
**reset ip interface**
**show ip counter**

# reset ip counter

**Syntax**

```
RESET IP
    COUnter={ALL|ARP|CAChe|ICmp|INTerface|IP|MULticast|ROUt
    e|SNmp|UDP}
```

**Description**

This command sets IP counters to zero. The **counter** parameter specifies particular counters depending on the option, and **all** resets all of them.

**Examples**

To reset the IP route counters to zero, use the command:

```
reset ip cou=rou
```

**Related Commands**

**reset ip**
**reset ip interface**
**show ip counter**

# reset ip interface

**Syntax** `RESET IP INTerface=`*`interface`*

where *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.

**Description** This command resets a logical IP interface. All routes associated with the interface, except interface and static routes, are purged from the routing table. All ARPs associated with this interface are purged from the ARP cache, and all counters for this interface are reset to zero.

The **interface** parameter specifies the name of the logical interface to be reset. The interface must currently be assigned to the IP routing module. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202. If you are using the GUI to view interfaces and have configured a large number, the page may take several minutes to display.

**Examples** To reset the first logical IP interface attached to vlan1, use the command:

```
reset ip int=vlan1
```

**Related Commands** **add ip interface**
**delete ip interface**
**disable ip interface**
**enable ip interface**
**reset ip**
**reset ip counter**
**set ip interface**
**show ip interface**

# set ddns

**Syntax**

```
SET DDNS [SERVER=server] [PORT=port] [USer=userid]
    [PASSword=password] [DYNamichost=hostnames]
    [CUSTomhost=hostnames] [STAtichost=hostnames]
    [PRImaryint=interface] [SECOndaryint=interface]
    [WILdcard={ON|OFF|NOCHG}]
    [MAIlexchanger={mailexchanger|NOCHG}]
    [BAckmx={YES|NO|NOCHG}] [OFFline={YES|NO}]
```

**Description** This command sets the parameters used for updating the Dynamic DNS.

| Parameter | Description |
|---|---|
| SERVER | The name of the DDNS server. It is a string of up to 31 characters, having a URL name format.<br>Default: **members.dyndns.org** |
| PORT | Specifies the remote port number to be used by a local HTTP client in order to convey the Dynamic DNS Update. Only port numbers 80, 8245 and 443 can be used. Ports 80 and 8245 are for HTTP, and port 443 is for HTTPS. Port 8245 may be used to bypass transparent HTTP proxies.<br>Default: **80** |
| USer | A user name used for an account with Dynamic DNS. It may contain up to 15 printable characters and is case sensitive. If the name contains spaces, these must be enclosed in double quotes.<br>Default: no default. |
| PASSword | A password to be used with Dynamic DNS account. It may contain up to 15 printable characters and is case sensitive. If the password contains spaces, these must be enclosed in double quotes.<br>Default: no default. |
| DYNamichost | Host names to be used with Dynamic DNS. Each name may contain up to 31 printable characters having a URL format. If a name contains spaces, these must be enclosed in double quotes. Up to four comma separated names can be entered. All names must be Internet registered.<br>Defaults: no default. |
| CUSTomhost | Host names for Custom DNS. Each name may contain up to 31 printable characters having a URL format. If a name contains spaces, these must be enclosed in double quotes. Up to four comma separated names can be entered. All names must be Internet registered.<br>Defaults: no default. |
| STAtichost | Host names to be used for Static DNS. Each name can be up to 31 printable characters long and is case sensitive. Up to four comma separated names can be entered.<br>Default: no default. |
| PRImaryint | An interface to be used for the primary WAN connection. The IP interface must be pre-defined.<br>Default: no default. |
| SECOndaryint | The interface to be used for the secondary WAN connection. The IP interface must be pre-defined.<br>Default: no default. |

| **Parameter** | **Description** (cont.) | |
|---|---|---|
| WILdcard | Whether to use the wildcard * feature when matching host names.<br>Default: **off** | |
| | ON | Turns on the wildcard option |
| | OFF | Turns off the wildcard option |
| | NOCHG | Retains the previous wildcard value |
| MAILexchanger | A mail exchanger to be used with the hostname. The mail exchanger is a string of up to 31 characters.<br>Although the router accepts any printable character string, DynDNS requires a specific format such as a URL address.<br>Default: no default | |
| | *mailexchanger* | The mail exchanger name. It must be one that can be resolved to an IP address or it is ignored.<br>By not providing a mail exchanger name or entering one that does not resolve to an A record, you will remove the record of the hostname. |
| | NOCHG | Preserves the current **mailexchanger** setting. |
| BAckmx | Whether the mail exchanger specified by the **mailexchanger** parameter is set up as a backup.<br>Default: **no** | |
| | Yes | The **mailexchanger** is set as the backup. |
| | No | The **mailexchanger** is not set as the backup. |
| | NOCHG | Retains the existing setting. |
| OFFline | Whether to set the hostnames to offline mode.<br>Default: **no** | |

**Examples** To set the DDNS operation for a user called Oliver, use the command:

```
set ddns us=oliver pas=fagin pri=ppp0
```

**Related Commands**

activate ddns update
disable ddns
disable ddns debug
enable ddns
enable ddns debug
show ddns

# set ip arp

**Syntax**

```
SET IP ARP=ipadd INTerface=atm-interface

SET IP ARP=ipadd INTerface=ppp-interface

SET IP ARP=ipadd INTerface=x25-interface
   CIRCuit=miox-circuit

SET IP ARP=ipadd INTerface=fr-interface DLCI=dlci

SET IP ARP=ipadd INTerface=eth-interface [ETHernet=macadd]

SET IP ARP=ipadd INTerface=vlan ETHernet=macadd
   POrt=port-number
```

where:

- *ipadd* is an IP address in dotted decimal notation.
- *atm-interface* is an ATM interface name such as atm0.1
- *ppp-interface* is a PPP interface name such as ppp0 or ppp0-1
- *x25-interface* is an X.25 DTE interface name such as x25t0 or x25t0-1
- *miox-circuit* is the name of a MIOX circuit defined for an X.25 interface 1 to 15 characters long. The name is not case-sensitive.
- *fr-interface* is a Frame Relay interface name such as fr0 or fr0-1
- *dlci* is the Data Link Connection Identifier (DLCI) of a Frame Relay DLC (circuit).
- *eth-interface* is an Eth port name such as eth0 or eth0-1
- *macadd* is the physical Ethernet (MAC) address of a host.
- *port-number* is the physical switch port number. Port numbers start at 1 and end at m, where m is the highest numbered Ethernet switch port, including uplink ports.

**Description**

This command modifies a static ARP entry in the ARP cache. The ARP entry must already exist.

The **arp** parameter specifies the IP address of the host.

The **interface** parameter specifies an existing interface over which the host can be reached. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202.

The **circuit** parameter specifies the MIOX circuit on an X.25 interface. If **circuit** is specified, **dlci** and **ethernet** cannot be specified.

The **dlci** parameter specifies the physical address for the host on a Frame Relay interface. If **dlci** is specified, **circuit** and **ethernet** cannot be specified.

The **ethernet** parameter specifies the physical (MAC) address for the host on an Eth or VLAN interface. If **ethernet** is specified, **dlci** and **circuit** cannot be specified.

The **port** parameter specifies the physical switch port number in a VLAN. If **interface** is a VLAN, the **port** parameter is valid; otherwise it is invalid.

**Examples**

To change the ARP entry for host 172.16.9.197 on interface eth0 (because, for example, the Ethernet interface on the host has been replaced and the host now has an Ethernet address of 00-BC-00-03-2F-9B), use:

```
set ip arp=172.16.9.197 int=eth0 eth=00-bc-00-03-2f-9b
```

To change the ARP entry for host 192.168.4.101 on interface vlan4 (because, for example, the Ethernet interface on the host has been replaced and the host now has an Ethernet address of 00-BC-00-03-2F-9B), use:

```
set ip arp=192.168.4.101 int=vlan4 po=3 eth=00-bc-00-03-2f-9b
```

**Related Commands**

add ip arp
delete ip arp
show ip arp
set ip arp refresharp
set ip arp timeout

# set ip arp refresharp

**Syntax**

```
SET IP ARP REFresharp={ON|YES|True|OFF|NO|False}
```

**Description**

This command specifies whether IP ARP entries are refreshed in the ARP cache as they are used (hit).

The specified **arp** must already exist within the ARP cache. You can view the contents of the ARP cache with the **show ip arp** command.

The **refresharp** parameter specifies whether to refresh IP ARP entries in the cache and restart the aging timer when the entry is used. The default is **on**.

**Related Commands**

add ip arp
delete ip arp
enable ip arp agepoll
set ip arp
show ip arp
set ip arp timeout

# set ip arp timeout

**Syntax**

```
SET IP ARP TIMEOut=multiplier
```

where *multiplier* is an integer from 1 to 1023

**Description**

This command changes the ARP timeout period, by changing the speed at which the router passes through the ARP cache deleting unused entries. For a detailed description of the timeout mechanism, see "Timing Out ARP Entries" on page 22-15.

The "base" value for the ARP timeout period varies from 256 to 512 seconds. This command changes the value by specifying a multiplier. The actual timeout value used by the router is:

```
multiplier x base timeout range
```

If you do not enter this command, the router's implementation uses a multiplier of 4 by default, giving a timeout of 1024 to 2048 seconds.

**Examples**

To delete ARP entries after 256 to 512 seconds without traffic, use the command:

```
set ip arp timeo=1
```

To delete ARP entries after 34 to 68 minutes without traffic, use the command:

```
set ip arp timeo=8
```

**Related Commands**

**add ip arp**
**delete ip arp**
**enable ip arp agepoll**
**set ip arp**
**set ip arp refresharp**
**show ip arp**

# set ip arpwaittimeout

**Syntax**

```
SET IP ARPWaittimeout=1..30
```

**Description**

This command sets the amount of time the router waits for a response after it sends an ARP request message.

The easiest way to test a changed wait timeout period is to ping an unavailable device. The timeout determines the delay between pinging an IP address and receiving the reply that the device is unreachable.

The **arpwaittimeout** parameter specifies the number of seconds that the router waits for a response to an ARP request message. If it does not receive a reply after that number of seconds, it notifies the sending device that the destination is unknown. You may need to increase the timeout period if you are communicating with devices that are slow to respond. The default is 1 second.

**Examples**

To set the router to wait 2 seconds after you ping a device before declaring that the device is unreachable, use the command:

```
set ip arpw=2
```

**Related Commands**

show ip

# set ip dns

**Syntax**

```
SET IP DNS [DOMain={ANY|domain-name}]
   {INTerface=interface|[PRIMary=ipadd] [SECOndary=ipadd]}
```

where:

- *domain-name* is a string of up to 255 characters. Valid characters are uppercase and lowercase letters, digits (0-9), and the underscore.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed (that is, eth0 is equivalent to eth0-0).
- *ipadd* is an IP address in dotted decimal notation.

**Description**

This command sets configuration information for the DNS servers to be used to resolve hosts in a particular domain into IP addresses. DNS servers for this domain must have already been configured with the **add ip dns** command on page 22-68.

The **domain** parameter specifies the domain for which this DNS server is to be used to resolve host names. DNS requests for hosts in this domain are sent to this server. If **any** is specified, the name server is used for domains not otherwise matched by another DNS entry. The default is **any**.

The **interface** parameter specifies the interface over which the router learns the address of a primary and/or a secondary name server. The primary and secondary name server's addresses can be either statically configured using the **primary** and **secondary** parameters, or learned dynamically over an interface. Name servers can be learned via DHCP over an Ethernet or VLAN interface or via IPCP over a PPP interface. If the **interface** parameter is specified, the **primary** and **secondary** parameters are not required. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202.

The **primary** parameter specifies the IP address of the name server to be used as the primary name server for resolving hosts in the specified domain. If the **primary** parameter is specified, the **interface** parameter must not be specified.

The **secondary** parameter specifies the IP address of the name server to be used as the secondary name server for resolving hosts in the specified domain. If the **secondary** parameter is specified, the **interface** parameter must not be specified.

If the router was originally configured to learn name servers dynamically over a particular interface for use in resolving host names in the specified domain, this configuration can be overridden by specifying values for one or both of the static name server parameters (**primary** and **secondary**). Similarly, if static name server addresses were originally configured, using the **interface** parameter causes name server information learned dynamically to overwrite the static name server configuration; static name server addresses are lost.

**Examples** To set the IP addresses of the default primary and secondary name servers to 192.168.20.1 and 192.168.20.2 respectively, use the command:

```
set ip dns prim=192.168.20.1 seco=192.168.20.2
```

To set the IP addresses of the primary and secondary name servers for use when resolving host names in the domain “oranges.com” to 192.168.20.1 and 192.168.20.2 respectively, use the command:

```
set ip dns dom=oranges.com prim=192.168.20.1
   seco=192.168.20.2
```

**Related Commands**

**add ip dns**
**delete ip dns**
**show ip dns**

# set ip dns cache

**Syntax**

```
SET IP DNS CAChe [SIze=0..1000] [TIMeout=1..60]
```

**Description**

This command controls the size of the IP DNS cache, which stores the responses to DNS requests that the router receives. A DNS cache containing 100 entries uses approximately 30 kilobytes of RAM.

The **size** parameter specifies the maximum number of entries allowed in the cache at any time. If the maximum number has been reached when a new DNS request is made, the oldest entry is deleted to make space. The default is 0, meaning the router does not cache responses to DNS requests.

The **timeout** parameter specifies the maximum time in minutes that an entry remains in the cache. Entries with a TTL less than the **timeout** parameter are stored with their own TTL. However, those with a TTL larger than this parameter are stored for the duration of **timeout**. Some types of DNS records are handled differently. For example, when a CNAME record is added to the cache, the TTL of the corresponding Type A record is stored. Cache entries are deleted when their TTL reaches 0. The default is 60 minutes.

**Examples**

To configure the DNS cache so it has a maximum size of 250 entries that are kept for no more than 15 minutes, use the command:

```
set ip dns cac si=250 tim=15
```

**Related Commands**

**add ip dns**
**delete ip dns**
**set ip dns**
**show ip dns**

# set ip dnsrelay

**Syntax**

```
SET IP DNSRelay INTerface={interface|NONE}
```

where *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.

**Description**

This command has been made obsolete by the **add ip dns** command on page 22-68 and **set ip dns** command on page 22-146, and is described for backwards compatibility. It no longer appears in dynamically generated configuration scripts, and router-generated configuration scripts replace **set ip nameserver** commands with **add ip dns** commands.

This command specifies the interface that the router uses to learn remote DNS server addresses. The interface is typically a dial-up connection to an ISP that provides the DNS name server PPP option. The router learns DNS addresses that have not been set using the **set ip nameserver** command. If **none** is specified, the router does not learn DNS server addresses via IPCP addresses. By default, DNS relay is not used to learn DNS server addresses.

**Related Commands**

**disable ip dnsrelay**
**enable ip dnsrelay**
**set ip nameserver**

# set ip filter

**Syntax** Traffic filter:

```
SET IP FILter=0..999 ACtion={INCLude|EXCLude} SOurce=ipadd
   [SMask=ipadd] [SPort={port-name|port-id}]
   [DEStination=ipadd [DMask=ipadd]]
   [DPort={port-name|port-id}]
   [ICMPCode={icmp-code-name|icmp-code-id}]
   [ICmptype={icmp-type-name|icmp-type-id}]
   [LOG={4..1600|DISabled|Dump|ENabled|Full|Header|None|OF
   f|ON|Yes}] [OPtions={False|OFF|ON|NO|True|YES}]
   [PROTocol={protocol|Any|Icmp|Ospf|Tcp|Udp}]
   [SEssion={Any|Established|Start}] [SIze=size]
   [ENTry=1..255]
```

Policy filter:

```
SET IP FILter=0..999 POLIcy=0..15 SOurce=ipadd
   [SMask=ipadd] [SPort={port-name|port-id}]
   [DEStination=ipadd [DMask=ipadd]]
   [DPort={port-name|port-id}]
   [ICMPCode={icmp-code-name|icmp-code-id}]
   [ICmptype={icmp-type-name|icmp-type-id}]
   [LOG={4..1600|DISabled|Dump|ENabled|Full|Header|None|OF
   f|ON|Yes}] [OPtions={False|OFF|ON|NO|True|YES}]
   [PROTocol={protocol|Any|Icmp|Ospf|Tcp|Udp}]
   [SEssion={Any|Established|Start}] [SIze=size]
   [ENTry=1..255]
```

Priority filter:

```
SET IP FILter=0..999 PRIOrity=P0..P7 SOurce=ipadd
   [SMask=ipadd] [SPort={port-name|port-id}]
   [DEStination=ipadd [DMask=ipadd]]
   [DPort={port-name|port-id}]
   [ICMPCode={icmp-code-name|icmp-code-id}]
   [ICmptype={icmp-type-name|icmp-type-id}]
   [LOG={4..1600|DISabled|Dump|ENabled|Full|Header|None|OF
   f|ON|Yes}] [OPtions={False|OFF|ON|NO|True|YES}]
   [PROTocol={protocol|Any|Icmp|Ospf|Tcp|Udp}]
   [SEssion={Any|Established|Start}] [SIze=size]
   [ENTry=1..255]
```

Routing filter:

```
SET IP FILter=0..999 ACtion={INCLude|EXCLude} SOurce=ipadd
   [ENTry=1..255] [SMask=ipadd]
```

where:

- *ipadd* is an IP address in dotted decimal notation.
- *port-name* is the predefined name for an IP port.
- *port-id* is an IP port number, or a range of ports in the form *low:high*.
- *icmp-code-name* is the predefined name for an ICMP reason code.
- *icmp-code-id* is the number of an ICMP reason code.
- *icmp-type-name* is the predefined name of an ICMP message type.

- *icmp-type-id* is the number of an ICMP message type.
- *protocol* is an IP protocol number.
- *size* is a number from 64 to 65535.
- *entry-number* is the position of this entry in the filter.

**Description** This command changes a pattern in an IP traffic filter, policy filter, priority filter or routing filter. It cannot change the type of the filter.

The **filter** parameter specifies the number of the filter where the pattern is to be changed.

The **source** parameter specifies the source IP address, in dotted decimal notation, for the pattern.

The **smask** parameter specifies the mask, in dotted decimal notation to apply to source addresses for this pattern. The mask is used to determine the portion of the source IP address in the IP packet that is significant for comparison with this pattern. The values of **source** and **smask** must be compatible. For each bit in **smask** that is set to zero, the equivalent bit in **source** must also be zero. If either **source** or **smask** is 0.0.0.0, then both must be 0.0.0.0. The default is 255.255.255.255, unless **source** is 0.0.0.0, in which case the default **smask** is 0.0.0.0.

The **sport** parameter specifies the source port to check against for this pattern as the recognised name of a well-known UDP or TCP port (Table 22-7 on page 22-73), a decimal value from 0 to 65535, or a range of numbers formatted *low:high*. If *low* is omitted, 0 is assumed; if *high* is omitted, the maximum port number is assumed. If a port other than **any** is specified, the **protocol** parameter is required and must be TCP or UDP. The default is **any**.

The **destination** parameter specifies the destination IP address for the pattern in dotted decimal notation. The default is 0.0.0.0.

The **dmask** parameter specifies the mask in dotted decimal notation to apply to the destination address for this pattern. The mask determines the portion of the destination IP address in the IP packet that is significant for comparison with this pattern. If **dmask** is specified, **destination** must also be specified. The values of **destination** and **dmask** must be compatible. For each bit in **dmask** that is set to zero, the equivalent bit in **destination** must also be zero. If either **destination** or **dmask** is 0.0.0.0, then both must be 0.0.0.0. If **destination** is not specified or is 0.0.0.0, the default **dmask** is 0.0.0.0. If **destination** is specified and is not 0.0.0.0, the default **dmask** is 255.255.255.255.

The **dport** parameter specifies the destination port to check against for this pattern as the recognised name of a well-known UDP or TCP port (Table 22-7 on page 22-73), a decimal value from 0 to 65535, or a range of numbers formatted *low:high*. If *low* is omitted, 0 is assumed; if *high* is omitted, the maximum port number is assumed. If a port other than **any** is specified, the **protocol** parameter is required and must be TCP or UDP. The default is **any**.

The **icmptype** and **icmpcode** parameters specify the ICMP message type and ICMP message reason code to match against the ICMP type and code fields in an ICMP packet. The **icmptype** parameter specifies the ICMP message type to match as a decimal value from 0 to 255, or the recognised name of an ICMP type (Table 22-8 on page 22-74). The **icmpcode** parameter specifies the ICMP message reason code to match as a decimal value from 0 to 255, or the recognised name of an ICMP reason code (Table 22-9 on page 22-75). Both parameters are valid when the **protocol** parameter is set to **icmp**.

The **log** parameter specifies whether matches to a filter entry result in a message being sent to the router's Logging facility, and the content of the log messages. This parameter enables logging of the IP packet filtering process down to the level of an individual filter entry. The default is **none**.

If you specify a number from 4 to 1600, the filter number, entry number, and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are logged with a message type/subtype of IPFIL/PASS (for patterns with an **include** action) or IPFIL/FAIL (for patterns with an **exclude** action). In addition, the first 4 to 1600 octets of the data portion of TCP, UDP, and ICMP packets or the first 4 to 1600 octets after the IP header of other protocol packets are logged with a message type/subtype of IPFIL/DUMP.

If you specify **dump** or **full**, the filter number, entry number, and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are logged with a message type/subtype of IPFIL/PASS (for patterns with an **include** action) or IPFIL/FAIL (for patterns with an **exclude** action). In addition, the first 32 octets of the data portion of TCP, UDP, and ICMP packets or the first 32 octets after the IP header of other protocol packets are logged with a message type/subtype of IPFIL/DUMP.

If you specify **enabled**, **header**, **on**, or **yes**, the filter number, entry number, and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are logged with a message type/subtype of IPFIL/PASS (for patterns with an **include** action) or IPFIL/FAIL (for patterns with an **exclude** action).

If you specify **disabled**, **off**, or **none**, matches to the filter entry are not logged.

The **options** parameter specifies the IP options field to use to check against the pattern. If **yes**, the pattern matches IP packets with options set; if **no**, the pattern matches packets without options set. The default is to match IP packets with or without IP options set.

The **protocol** parameter specifies the protocol to check against for this pattern as a decimal value from 0 to 65534. Valid protocol names are:

- Internet Control Message Protocol (ICMP)
- Open Shortest Path First Protocol (OSPF)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

If either **sport** or **dport** are specified, **protocol** must be defined as TCP or UDP. Specifying TCP or UDP filters packets from companion protocols, for example ICMP, RIP, and OSPF, that do not use TCP or UDP as a transport mechanism. The default is **any**.

The **session** parameter specifies the type of TCP packet to match, and can be used when the **protocol** parameter specifies TCP. If **start** is specified, the pattern matches TCP packets with the SYN bit set and the ACK bit clear. If **established** is specified, the pattern matches TCP packets with either the SYN bit clear or the ACK bit set. If **any** is specified, the pattern matches any TCP packet. The default is **any**.

The **size** parameter specifies the maximum reassembled size to match against for each IP fragment. If the fragment's offset plus size is greater than the value specified, the fragment is discarded.

The **entry** parameter specifies the entry number in the filter to be changed. Existing patterns with the same or higher entry numbers are pushed down the filter. The default is to add the new pattern to the end of the filter.

The **action** parameter specifies, for traffic filters, the action to take when the pattern is matched. The **action**, **policy**, and **priority** parameters are mutually exclusive so only one may be specified. If **include** is specified, the IP packet is processed and forwarded for traffic filters, or the IP route is selected for routing filters. If **exclude** is specified, the IP packet is discarded for traffic filters, or the IP route is excluded for routing filters.

The **policy** parameter is used for policy-based routing and specifies the policy to use when the pattern is matched. The policy number is assigned to incoming packets but employed during forwarding (transmission). The **action**, **policy**, and **priority** parameters are mutually exclusive so only one may be specified.

- For policy numbers from 0 to 7, routes with a matching policy are considered first.
- For policy numbers from 8 to 15, routes with a policy of *n*-8 (where *n* is the filter policy) are considered first, and the policy value *n*-8 is written into the TOS field of the packet.

The **priority** parameter is used for priority routing and specifies the priority when the pattern is matched. The priority number is assigned to incoming packets but employed during forwarding (transmission). Packets can be assigned a priority from p3 (highest) to p7 (lowest). The default is p5. Priority levels p0, p1, and p2 should not be used because they may conflict with router system activities. The **action**, **policy**, and **priority** parameters are mutually exclusive so only one may be specified.

**Examples** To set the session to be matched by entry 3 of filter 2 to established, use the command:

```
set ip fil=2 ent=3 se=e
```

**Related Commands**

**add ip filter**
**add ip route filter** in Chapter 28, Filtering IP Routes
**delete ip filter**
**delete ip route filter** in Chapter 28, Filtering IP Routes
**show ip filter**
**show ip route filter** in Chapter 28, Filtering IP Routes

# set ip host

**Syntax**

```
SET IP HOst=name IPaddress=ipadd
```

where:

- *name* is a string up to 60 characters long. If the string contains spaces, it must be in double quotes.
- *ipadd* is an IP address in dotted decimal notation.

**Description**

This command modifies the IP address associated with a user-defined name for an IP host in the host name table. The host name table makes it easier to Telnet to commonly accessed hosts by enabling the user to enter a shorter, easier to remember name for the host rather than the host’s full IP address or domain name.

The **host** parameter specifies the user-defined name for the IP host. A host with the same name must already exist in the host name table. When a host name is specified in the Telnet command, the entire name is used to match a name in the host name table. All characters are used in the comparison, including nonalphabetic characters when present.

The **ipaddress** parameter specifies the IP address for the host.

**Examples**

To change the IP address for host name “zaphod” in the host name table from 172.16.1.5 to 172.16.9.8, use:

```
set ip ho=Zaphod ip=172.16.9.8
```

To Telnet to the host, use any of the following commands:

```
telnet zaphod
telnet zaphod.company.com
telnet 172.16.9.8
```

**Related Commands**

**add ip host**
**delete ip host**
**set ip nameserver**
**set ip secondarynameserver**
**show ip host**

# set ip interface

**Syntax**

```
SET IP INTerface=interface [BROadcast={0|1}]
    [DIRectedbroadcast={False|NO|OFF|ON|True|YES}]
    [FILter={0..999|NONE}] [FRAgment={NO|OFF|ON|YES}]
    [GRAtuitousarp={ON|OFF}] [GRE={0..100|NONE}]
    [IGMPProxy={OFF|UPstream|DOWNstream}]
    [INVersearp={ON|OFF}] [IPaddress=ipadd|DHCP]
    [MASK=ipadd] [METric=1..16]
    [MULticast={BOTH|OFF|ON|RECeive|SENd}]
    [NOTIfyospfdown={False|NO|OFF|ON|True|YES}]
    [OSPFmetric=1..65534|DEFAULT]
    [POLicyfilter={0..999|NONE}]
    [PRIorityfilter={0..999|NONE}]
    [PROxyarp={False|NO|OFF|ON|True|YES|LOCal|STrict
    |DEFRoute}] [RIPMetric=1..16]
    [SAMode={Block|Passthrough}]
    [VJC={False|NO|OFF|ON|True|YES}]
    [VLANPRiority={0..7|None}] [VLantag={1..4094|None}]
```

where:

- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.
- *ipadd* is an IP address in dotted decimal notation.

**Description**

This command modifies the configuration of a logical interface that the IP module uses. Only explicitly specified parameters are changed. For example, if vlan1 has the IP address 192.168.10.4 and mask 255.255.255.0, and you enter **set ip int=vlan1 ip=10.5.1.4**, the IP interface changes to 10.5.1.4 but the mask remains 255.255.255.0.

This command requires a user with security officer privilege when the router is in security mode. Note that all dial-in PPP connections will be disconnected when this command is executed.

The IP configuration of an interface cannot be changed while DVMRP or PIM is attached to the interface. The DVMRP or PIM interface must first be deleted, and then re-added after the IP changes have been made. See Chapter 24, IP Multicasting for information about DVMRP and PIM.

The **interface** parameter specifies the name of the logical interface, and implicitly, the attached Layer 2 interface. The interface must currently be assigned to the IP module. At least two interfaces must be defined before the router can route IP packets, but only one interface (usually eth0) needs to be defined when the router is acting as a server. A maximum of 640 interfaces can be added. When an interface is added, it is automatically enabled. Only one logical interface may be configured to the same IP network or subnet. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)

- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202. If you are using the GUI to view interfaces and have configured a large number, the page may take several minutes to display.

The **broadcast** parameter specifies whether a broadcast address with all 1s or all 0s is used. The default is 1. This parameter should not be set to 0 without careful consideration of the consequences. It is provided to allow compatibility with older host implementations that do not meet the current standard.

The **directedbroadcast** parameter specifies whether the router allows network or subnet broadcasts to be forwarded to the network directly attached to the logical interface. The default is **no**.

The **filter** parameter specifies the traffic filter to apply to IP packets transmitted or received over the logical interface. The filter must already have been defined with the **add ip filter** command on page 22-70. A logical interface may have a maximum of one traffic filter, one policy filter and one priority filter, but the same traffic, policy or priority filter can be assigned to more than one interface. Traffic filters are applied to packets received via the logical interface. The default is not to apply a filter.

The **fragment** parameter specifies whether the "*Do not fragment*" bit is obeyed for outgoing IP packets that are larger than the MTU of the interface. If **yes**, the "*Do not fragment*" bit is ignored and outgoing IP packets larger than the MTU of the interface are fragmented. This is particularly useful for interfaces configured with GRE, SA and/or IPsec encapsulation that can potentially increase packet sizes beyond the MTU of the interface. If **no**, the "*Do not fragment*" bit is obeyed and IP packets larger than the MTU of the interface are discarded. This is the normal behaviour for IP. The **fragment** parameter has no effect on processing packets smaller than the interface MTU. The default is **no**.

The **gratuitousarp** parameter enables or disables the acceptance of gratuitous ARP requests or replies. The default is **on**.

The **gre** parameter specifies the GRE (Generic Routing Encapsulation) entity associated with the logical interface. The GRE entity must have been created previously with the **add gre** command on page 30-12 of Chapter 30, Generic Routing Encapsulation (GRE). The default is **none**.

The **igmpproxy** parameter specifies the status of IGMP proxying for the specified interface. If you specify **off**, the interface does not do IGMP Proxy. If you specify **upstream**, the interface passes IGMP messages in the upstream direction. A router can have one interface with the IGMP proxy direction equal to **upstream**. If you specify **downstream**, the interface can receive IGMP messages from the downstream direction. The default is **off**. To display information about IGMP and multicast group membership for each IP interface, use the **show ip igmp** command on page 24-99 of Chapter 24, IP Multicasting.

The **inversearp** parameter enables or disables the operation of the Inverse Address Resolution Protocol (INVARP) on ATM interfaces. The **inversearp** parameter must be set to **on** for IPoA configurations, and to **off** for RFC 1483 Routed configurations. The default is **off**. (Inverse ARP is always on for Frame Relay interfaces.)

The **ipaddress** parameter specifies the IP address of the logical interface. If **dhcp** is specified, the router acts as a DHCP client and obtains the configuration of the IP interface via DHCP. Table 22-10 on page 22-82 lists the parameters from the DHCP reply that the router uses. If an IP interface is configured to use DHCP to obtain its IP address and subnet mask, the interface does not take part in IP routing until the IP address and subnet mask have been set by DHCP. Remote address assignment must be enabled with the **enable ip remoteassign** command on page 22-131 before IP interfaces accept addresses dynamically assigned by DHCP.

The **mask** parameter specifies the subnet mask for the logical interface. The value must be consistent with the value specified for the **ipaddress** parameter. If **ipaddress** is set to **dhcp**, the **mask** parameter should not be set because the subnet mask received from the DHCP server is used.

The **metric** parameter specifies the cost of crossing the logical interface for OSPF. If **default** is specified, the interface is restored to the default metric value. Setting **metric** to a value other than **default** provides a mechanism to provide a metric for an interface that is preferred over the OSPF automatic metric setting (if enabled via **set ospf autocost=on**). If **metric** has been set to a numerical value, it must be set to **default** before **set ospf autocost** can take effect for this interface. The default is 1.

The **multicast** parameter specifies whether the interface receives and forwards multicast packets, when neither DVMRP nor PIM are enabled. If **both** or **on** is specified, the router sends and receives multicast packets. If **off**, the router does not send or receive multicast packets. If **receive** is specified, the router receives but does not send packets. If **send** is specified, the router sends but does not receive them. Note that this parameter applies to the entire IP interface, not an individual logical interface so setting this parameter on one logical interface sets it for all associated with the same IP interface. This parameter determines the interface’s static behaviour for multicast packets. When DVMRP or PIM-SM is enabled, it determines the forwarding behaviour of interfaces dynamically, and this parameter has no effect (Chapter 24, IP Multicasting). The default is **receive**.

The **notifyospfdown** parameter specifies whether IP generates a notification to OSPF when this interface goes down if it is a PPP on-demand interface. IP always sends notifications for all other interfaces, even if this is set to **no**. If **on**, **true** or **yes** is specified, then IP notifies OSPF when the interface goes down and OSPF sets the interface state to Down. OSPF does not send Hello messages to the interface, and OSPF is inactive on the interface until it receives an Up notification. If **false**, **no** or **off** is specified, IP does not notify OSPF when the interface goes down. OSPF continues to consider the interface state as Up, and keeps sending Hello packets. These Hello packets allow OSPF to bring the link up for on-demand PPP links. Note that this parameter applies to the entire IP interface, not an individual logical interface. Setting it on one logical interface sets it on all other logical interfaces associated with the same IP interface. The default value for this parameter is **yes**.

The **policyfilter** parameter specifies the policy filter to apply to IP packets received over the logical interface. The filter must already have been defined with the **add ip filter** command on page 22-70. A logical interface may have a maximum of one traffic filter, one policy filter, and one priority filter. However, the same traffic, policy or priority filter can be assigned to more than one interface. Policy filters are applied to packets when they are transmitted. The default is not to apply a filter.

The **priorityfilter** parameter specifies the priority filter to apply to IP packets transmitted over the logical interface. The filter must already have been defined with the **add ip filter** command on page 22-70. A logical interface may have a maximum of one traffic filter, one policy filter, and one priority filter. However, the same traffic, policy or priority filter can be assigned to more than one interface. Priority filters are applied to packets as they are transmitted. The default is not to apply a filter.

The **proxyarp** parameter enables or disables proxy ARP responses to ARP requests. This parameter is valid for Eth and VLAN interfaces. The default is **on**.

If **no**, **off**, or **false** is specified, the router does not act as a proxy for ARP. For all other options the router transmits proxy ARP responses to requests for addresses that can be reached via specific routes if they exist. If you specify **yes**, **on**, **true** or **strict**, the router only proxy responds if it has a specific route to the address being requested, excluding the interface route that the ARP request arrived from. It ignores all other APR requests. You can increase the range of ARP requests that the router proxy responds to by using either the **local** or **defroute** parameter. When you specify **defroute**, the router responds to all proxy ARP Requests to remote locations. For addresses where there is no specific route, the router transmits a proxy ARP response containing its default route (0.0.0.0), if it exists. When you specify **local**, the router makes proxy responses to ARP requests for the following types of addresses:

- addresses to which it has a specific route
- addresses within the same subnet as the interface on which the ARP request was received - referred to as *local* ARP requests

For local ARP requests, the router responds with its own MAC address to all ARP requests within the local subnet. This means that all hosts communicating within the subnet must send their packets to the MAC address of the router.This prevents the hosts from using MAC address resolution to communicate directly with one another, and enables the router to have control over which hosts may communicate with one another. When local proxy ARP is enabled on an interface, the router does not send ICMP redirect messages on that interface (for information about ICMP redirect messages, see “ICMP” on page 22-19).

Setting the router to **defroute** mode is a departure from RFC 1027.

The **ripmetric** parameter specifies the cost of crossing the logical interface for RIP. The default is 1. The **metric** parameter is also accepted for backwards compatibility.

The **samode** parameter specifies how the logical interface handles IP packets that do not belong to one of the security associations assigned to the logical interface. If **block** is specified, IP packets that do not belong to a security association assigned to the logical interface are blocked from transiting the interface and are discarded. If **passthrough** is specified, IP packets that do not belong to a security association assigned to the logical interface are allowed to transit the interface and are forwarded normally by the IP routing software. The default is **block**. This parameter takes affect when one or more security associations have been assigned to the logical interface with the **add ip sa** command on page 22-95.

The **vjc** parameter is valid for Point-to-Point Protocol (PPP) and X25T interfaces, and specifies whether Van Jacobson header compression is to be used on the Layer 2 interface. The **vjc** parameter applies to all logical interfaces attached to the same Layer 2 interface. Changing the setting on one logical interface alters the setting on the others attached to the Layer 2 interface. Compression provides the most advantage on slower link speeds (up to 48 kbps). At speeds of 64 kbps and higher, compression actually reduces efficiency and should be disabled. Van Jacobson's TCP/IP header compression should not be enabled on a multilink PPP interface. The default is **off**.

The **vlanpriority** parameter specifies the value of the 802.1p User Priority field of the VLAN tag. This priority is written into all VLAN-tagged frames sent out the interface. Downstream routers may use this priority to determine the quality of service the frame receives. This parameter is only valid when **vlantag** is specified. The default is **none** when **vlantag**=**none**, and 0 when **vlantag** specifies a value.

The **vlantag** parameter specifies the VID (VLAN Identifier) to be included in the header of each frame that is transmitted over the logical interface. This parameter is valid for Eth interfaces only. Multiple logical interfaces on the same physical interface can share the same VLAN tag. The default is **none**, which mean no VID is included. For more information, see "VLAN Tagging on Eth Interfaces" on page 22-29.

**Examples** To set the first IP interface attached to PPP2 with an IP address of 172.16.248.33, a subnet mask of 255.255.255.0, and a metric of 5, use the command:

```
set ip int=ppp2-0 ip=172.16.248.33 mask=255.255.255.0 ripm=5
```

To associate the second IP interface attached to Eth2 with GRE entity 3, use:

```
set ip int=eth2-1 gre=3
```

To modify two logical interfaces on the eth0 interface so that they have different VLAN tags and traffic on one has a higher priority, use the commands:

```
set ip int=eth0-0 ip=192.168.1.1 vl=2 vlanpr=2
```

```
set ip int=eth0-1 ip=192.168.2.1 vl=3 vlanpr=3
```

You could use these two interfaces to separate and prioritise traffic destined for two different users.

**Related Commands**

**add ip interface**
**delete ip interface**
**disable ip interface**
**enable ip interface**
**reset ip interface**
**show ip igmp** in Chapter 24, IP Multicasting
**show ip interface**

# set ip local

**Syntax**

```
SET IP LOCal[={DEFault|1..15}] [FILter={0..999|None}]
   [GRE={0..100|None}] [IPaddress=ipadd]
   [POLicyfilter={0..999|None}]
   [PRIorityfilter={0..999|None}]
```

where *ipadd* is an IP address in dotted decimal notation

**Description**

This command modifies the parameters of one the router’s local interfaces. If you do not specify a local interface number, then it will modify the router's default local interface.

The **local** parameter specifies the unique identifying number for the particular local interface, or the default local interface. The naming convention for this interface is the concatenation of the word *local* along with this identifying number. The default is **default**.

The **filter** parameter specifies the traffic filter to apply to IP packets transmitted or received over the interface. The filter must already have been defined with the **add ip filter** command on page 22-70. An interface may have a maximum of one traffic filter, one policy filter and one priority filter, but the same traffic, policy or priority filter can be assigned to more than one interface. Traffic filters are applied to packets received via the interface. The default is not to apply a filter.

The **gre** parameter specifies the GRE (Generic Routing Encapsulation) entity associated with the interface. The specified GRE entity must have been created previously with the **add gre** command on page 30-12 of Chapter 30, Generic Routing Encapsulation (GRE). The default is **none**.

The **ipaddress** parameter specifies the IP address of the interface. The IP address must be the IP address of one of the router’s active IP interfaces. Specifying an IP address of 0.0.0.0 effectively ‘unsets’ the IP address of the local interface.

The **policyfilter** parameter specifies the policy filter to apply to IP packets received over the interface. The filter must already have been defined with the **add ip filter** command on page 22-70. An interface may have a maximum of one traffic filter, one policy filter, and one priority filter, but the same traffic, policy, or priority filter can be assigned to more than one interface. Policy filters are applied to packets as they are transmitted. The default is not to apply a filter.

The **priorityfilter** parameter specifies the priority filter to apply to IP packets transmitted over the interface. The filter must already have been defined with the **add ip filter** command on page 22-70. An interface may have a maximum of one traffic filter, one policy filter, and one priority filter, but the same traffic, policy, or priority filter can be assigned to more than one interface. Priority filters are applied to packets as they are transmitted. The default is not to apply a filter.

**Examples**

To set the IP address of the default local IP interface to 192.168.33.11, use:

```
set ip loc ip=192.168.33.11
```

To set the local interface 3 to have IP address 192.168.33.11, use:

```
set ip local=3 ip=192.168.33.1
```

To remove the IP address of the default local IP interface, use:

```
set ip loc ip=0.0.0.0
```

**Related Commands**

add ip interface
add ip local
delete ip interface
delete ip local
set ip interface
show ip interface

# set ip nameserver

**Syntax**

```
SET IP NAMEserver=ipadd
```

where *ipadd* is an IP address in dotted decimal notation

**Description**

This command has been made obsolete by the **add ip dns** command on page 22-68 and **set ip dns** command on page 22-146, and is described for backwards compatibility. It no longer appears in dynamically generated configuration scripts, and router-generated configuration scripts replace **set ip nameserver** commands with **add ip dns** commands.

This command specifies the IP address of a host able to act as the primary name server for the router. Name servers are used to resolve Telnet requests to host names that are not in the host name table. If the host is entered into the host table, then no access to a name server is required. This may suit installations that have no name server.

A secondary name server can also be specified with the **set ip secondarynameserver** command, another obsolete command. When the router performs a DNS lookup, it firsts sends the request to the primary name server. If a response is not received within 20 seconds the request is sent to the secondary name server.

**Examples**

To specify the host with IP address 172.16.1.5 as a name server, use:

```
set ip name=172.16.1.5
```

The equivalent command to the example given above, using the ADD IP DNS command, is:

```
add ip dns prim=172.16.1.5
```

This command would be used if the default primary name server had not previously been configured. If the primary name server had previously been configured the IP address may be changed with the command:

```
set ip dns prim=172.16.1.5
```

**Related Commands**

**add ip host**
**delete ip host**
**set ip dnsrelay**
**set ip host**
**set ip secondarynameserver**
**show ip**
**show ip host**

# set ip nat maxfragments

**Syntax**

```
SET IP NAT MAXFragments=8..50
```

**Description**

This command sets the maximum number of fragments that a fragmented IP packet may consist of when enhanced fragment handling is enabled for IP NAT.

The **maxfragments** parameter specifies the maximum number of fragments that an IP packet may consist of. The default is 20.

Enhanced fragment handling for IP NAT is disabled by default. When disabled, fragmented IP packets can only be processed by IP NAT if the packet consists of no more than 8 fragments, and the total data contained in all the fragments is 1780 bytes or less. Enhanced fragment handling for IP NAT is enabled with the **enable ip nat** command.

**Examples**

To set the maximum number of fragments in a packet to be processed by IP NAT to 25, use the command:

```
set ip nat maxf=25
```

**Related Commands**

**disable ip nat**
**enable ip nat**
**show ip nat**

# set ip route

**Syntax**

```
SET IP ROUte=ipadd INTerface=interface MASK=ipadd
   NEXThop=ipadd [CIRCuit=miox-circuit] [DLCi=dlci]
   [METRIC=1..16] [METric1=1..16] [METRIC2=1..65535]
   [POLIcy=0..7] [PREFerence=0..65535] [TAG=1..65535]
```

where:

- *ipadd* is an IP address in dotted decimal notation.
- *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.
- *miox-circuit* is the name of a MIOX circuit defined for an X.25 interface 1 to 15 characters long. The name is not case-sensitive.
- *dlci* is the Data Link Connection Identifier (DLCI) of a Frame Relay DLC (circuit).

**Description**

This command modifies an existing static route in the IP route table. Static routes can be used to define default routes to external routers or networks. A default route is one with a network address of 0.0.0.0. When the router receives data and cannot find a route for it, it sends the data to the default route.

The **route** parameter specifies the IP address of the static route.

The **interface** parameter specifies the IP interface with which the route is associated. The interface must already exist and be assigned to the IP module. Valid interfaces are:

- eth (such as eth0, eth0-1)
- ATM (such as atm0.1)
- PPP (such as ppp0, ppp1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)
- VLAN (such as vlan1, vlan1-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202. If the interface is a Frame Relay interface, the **dlci** parameter is required and specifies the DLC to use on the Frame Relay interface. If the interface is an X.25 DTE interface, the **circuit** parameter is required and specifies the name of a MIOX circuit already defined for the X.25 DTE interface. If a logical interface is specified, the route is only added to the specified logical interface. If a logical interface is not specified, 0 is assumed and the route is only added to logical interface 0. The interface cannot be a local interface.

The **nexthop** parameter specifies the IP address of the next hop (router) for the route. The default is the IP address of the interface specified by the **interface** parameter. For a PPP link, **nexthop** should be the IP address of the remote end of the PPP link.

The **mask** parameter specifies the subnet mask for the route. A check is performed on the route and mask to verify that the route is the same before and after masking. This ensures that a static route is not specified to more than its subnet mask.

The **metric1** parameter specifies the cost of traversing the route for RIP. The default is 1. The normal range is from 2 to 16. A metric of 1 should be used when adding a subnet to an interface. The **metric** parameter is also accepted for backwards compatibility.

The **metric2** parameter specifies the cost of traversing the route. The default is 1.

If static routes are redistributed into OSPF, **metric1** and **metric2** may be used as metric values for the redistributed routes. If **metric2** is configured, the route is redistributed into OSPF as a Type-2 route. For more information see the **add ospf redistribute** command on page 26-38 of Chapter 26, Open Shortest Path First (OSPF).

The **policy** parameter specifies the type of service for the route. The default is 0.

The **preference** parameter specifies the preference for the route. When more than one route in the route table matches the destination address in an IP packet, the route with the lowest preference value is used to route the packet. If two or more routes have the same preference, the route with the lowest metric is used. Interface routes have a preference of 0 and RIP routes have a preference of 100. The default preference for static routes other than 0.0.0.0 is 60. The default for the default static route 0.0.0.0 is 360.

The **tag** parameter specifies an integer to tag the route with. You can then match against this number in a route map and only import the appropriately-tagged routes into BGP or OSPF.

**Examples**

To set the subnet 172.16.9.0 on interface eth0 to have a RIP metric of 2, use the command:

```
set ip rou=172.16.9.0 mask=255.255.255.0 int=eth0
   next=0.0.0.0 met=2
```

**Related Commands**

**add ip route**
**delete ip route**
**show ip route**

# set ip route preference

**Syntax**

```
SET IP ROUte PREFerence={DEFault|1..65535}
   PROTocol={BGP-ext|BGP-int|OSPF-EXT1|OSPF-EXT2|
   OSPF-INTEr|OSPF-INTRa|OSPF-Other|RIP|ALL}
```

**Description** This command sets the IP route table preference for routes learned via a specific protocol. When more than one route in the route table matches the destination address in an IP packet, the route with the lowest preference value is used to route the packet. If two or more routes matching the packet's destination have the same preference, then the router selects the route using the following process:

1. The router inspects the metric value of each of these routes and selects the route with the lowest metric.
2. If multiple routes share the lowest preference and metric values, then the router inspects the mask of each of these routes and selects the route with the longest mask.

Existing dynamically learned routes in the routing table and new routes added later are updated with the specified preference value. Packet processing times may be affected for a short time while the routing table is updated with new preference values.

The **preference** parameter sets the preference for routes learned via the specified protocol. A route with a low preference value has priority over a route with a high preference value. If **default** is specified, the preference reverts to the default for this protocol type (Table 22-12). Unless you are setting **protocol** to **all**, you must set the preference values for OSPF protocol types in such a way that **ospf-intra < ospf-inter < ospf-ext1 < ospf-ext2 < ospf-other**.

Table 22-12: Default preference values for each protocol type

| Protocol type | Default preference |
|---|---|
| rip | 100 |
| ospf-intra | 10 |
| ospf-inter | 11 |
| ospf-ext1 | 150 |
| ospf-ext2 | 151 |
| ospf-other | 152 |
| bgp-int | 170 |
| bgp-ext | 170 |

The **protocol** parameter specifies which protocol's routing table is updated with the new preference value. If **all** is specified, all protocol routing tables are updated with the new preference value.

**Examples** To set the router to use OSPF routes in preference to equally-specific RIP routes, give RIP routes a higher preference value than OSPF routes by using the command:

```
set ip rou pref=160 prot=rip
```

To set the router to select routes based on the metrics and mask values only by setting the preference value to the same value for all routes, use the command:

```
set ip rou pref=100 prot=all
```

**Related Commands** **show ip route preference**

# set ip route template

**Syntax**

```
SET IP ROUte TEMPlate=name [NEXThop=ipadd]
   [CIRCuit=miox-circuit] [DLCi=dlci] [METric=1..16]
   [METRIC1=1..16] [METRIC2=1..65535] [POLIcy=0..7]
   [PREFerence=0..65535]
```

where:

- *name* is a string 1 to 31 characters long, and is not case-sensitive. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.
- *ipadd* is an IP address in dotted decimal notation.
- *miox-circuit* is the name of a MIOX circuit defined for an X.25 interface 1 to 15 characters long. The name is not case-sensitive.
- *dlci* is the Data Link Connection Identifier (DLCI) of a Frame Relay DLC (circuit).

**Description**

This command modifies an existing IP route template. IP route templates are used by the router to add IP routes to IP subnetworks discovered during normal operation by other protocols such as IPsec. This is required if IP traffic to the discovered IP subnetwork needs to be routed via a route other than the default route.

The **dlci** parameter specifies the DLC to use on the Frame Relay interface. This parameter is valid when the template was added with a Frame Relay interface.

The **circuit** parameter specifies the name of a MIOX circuit to use on the X.25 DTE interface. This parameter is valid when the template was added with a X.25 DTE interface.

The **nexthop** parameter specifies the IP address of the next hop (router) for any route added using this template. The default is the IP address of the interface specified by the **interface** parameter. For a PPP link, **nexthop** should be the IP address of the remote end of the PPP link.

The **metric1** parameter specifies the cost of traversing any route added using this template for RIP. The default is 1. The normal range is from 2 to 16. A metric of 1 should be used when adding a subnet to an interface.The **metric** parameter is also accepted for backwards compatibility.

The **metric2** parameter specifies the cost of traversing any route added using this template for OSPF. The default is 1.

The **policy** parameter specifies the type of service for any route added using this template. The default is 0.

The **preference** parameter specifies the preference for routes added with this template. When more than one route in the route table matches the destination address in an IP packet, the route with the lowest preference value is used to route the packet. If two or more routes have the same preference, the route with the longest subnet mask is used. Interface routes have a preference of 0 and RIP routes have a preference of 100. The default preference for static routes other than 0.0.0.0 is 60. The default for the default static route 0.0.0.0 is 360.

**Examples** To set the preference of routes created with the IP route template named "branch_office" to 90, use the command:

```
set ip rou temp=branch_office pref=90
```

**Related Commands** **add ip route template**
**create ipsec policy**
**delete ip route template**
**show ip route template**

# set ip secondarynameserver

**Syntax**

```
SET IP SECOndarynameserver=ipadd
```

where *ipadd* is an IP address in dotted decimal notation

**Description**

This command has been made obsolete by the **add ip dns** command on page 22-68 and **set ip dns** command on page 22-146, and is described for backwards compatibility. It no longer appears in dynamically generated configuration scripts, and router-generated configuration scripts replace **set ip secondarynameserver** commands with **add ip dns** commands.

This command sets the IP address of the secondary name server. Name servers are used to resolve Telnet requests to host names that are not in the host name table. If the host is entered into the host table, then no access to a name server is required. This may suit installations that have no name server. When the router performs a DNS lookup, it firsts sends the request to the primary name server. If a response is not received within 20 seconds, the request is sent to the secondary name server.

**Example**

To set the router's secondary name server address to 192.168.2.1, use the command:

```
set ip seco=192.168.2.1
```

**Related Commands**

**add ip host**
**delete ip host**
**set ip host**
**set ip nameserver**
**show ip**
**show ip host**

# set ping

**Syntax**

```
SET PING
    [[IPaddress=]{ipadd|ipv6add[%interface]|host|domain-nam
    e}] [DElay=seconds] [Length=number]
    [NUMber={number|CONTinuous}] [PATTern=hexnum]
    [SIPAddress={ipadd|ipv6add}] [SCReenoutput={YES|NO}]
    [TIMEOut=1..60] [TOS=0..255]

SET PING [[IPXAddress=]network:station] [DElay=seconds]
    [Length=number] [NUMber={number|CONtinuous}]
    [PATTern=hexnum] [SIPXaddress=network:station]
    [SCReenoutput={YES|NO}] [TIMEOut=1..60]

SET PING [[APPLEAddress=]network.node] [DElay=seconds]
    [Length=number] [NUMBER={number|CONtinuous}]
    [PATTern=hexnum] [SAPpleaddress=network.node]
    [SCReenoutput={YES|NO}] [TIMEOut=1..60]

SET PING [[OSIAddress=]nsap] [DElay=seconds]
    [Length=number] [NUMber={number|CONtinuous}]
    [PATTern=hexnum] [SOSIaddress=nsap]
    [SCReenoutput={YES|NO}] [TIMEOut=1..60]
```

where:

- *ipadd* is an IPv4 address in dotted decimal notation.
- *ipv6add* is a valid IPv6 address.
- *interface* is the interface the ping request is sent for a request to ping an IPv6 link-local address, e.g. vlan1. Separate the address and interface with a % sign.
- *host* is a host name from the host name table.
- *domain-name* is a valid domain name.
- *network:station* is a valid Novell network number and station MAC address, expressed as hexadecimal numbers. Leading zeros may be omitted.
- *network.node* is an AppleTalk network number from 0 to 65279 or an AppleTalk network number in the format “nnnnn-nnnnn”, and an AppleTalk node number from 0 to 127.
- *nsap* is a valid OSI NSAP address in dotted hexadecimal notation.
- *seconds* is a decimal number from 0 to 4294967295.
- *hexnum* is an 8-digit hexadecimal number, optionally proceeded by the characters “0x”.

**Description** This command sets the defaults for the **ping** command on page 22-135.

The router’s extended **ping** command on page 22-135 supports IPv4, IPv6, OSI, IPX, and AppleTalk protocols, using the protocol’s native echo request message.

If there is no default destination and a destination is not specified on the **ping** command on page 22-135, a ping is not generated and an error message is displayed.

The **ipaddress**, **ipxaddress**, **osiaddress**, and **appleaddress** parameters specify the destination address for ping packets for IP, IPX, OSI, and AppleTalk networks, respectively. If these parameters have already been set, they can be restored to their default “not set” state by specifying:

- 0.0.0.0 or :: for **ipaddress**
- 0:0 for **ipxaddress**
- 0.0 for **appleaddress**

You can specify a valid IP address, a host name defined in the host name table, or a domain name as the destination IP address. To add a host to the host name table, use the **add ip host** command on page 22-79. To configure a DNS for the router to use to resolve domain names, use the **add ip dns** command on page 22-68.

Pinging an IPv6 link-local address requires interface information as well as the address because a single link-local address can belong to several interfaces. To ping a link-local address, specify the interface out which the router is to send the ping request, as well as the address to which the router is to send the ping request (Figure 31-2 on page 31-19 in Chapter 31, Internet Protocol version 6 (IPv6)). For example:

```
ping fe80::7c27%vlan1
```

The **delay** parameter specifies the time interval in seconds between ping packets. The default is 1 second.

The **length** parameter specifies the number of data bytes of the specified pattern to include in the data portion of the ping packet. The default is 24.

The **number** parameter specifies the number of ping packets to transmit. If this parameter is not specified, the default is used. If **continuous** is specified, the **timeout** parameter must be set to a value greater than 0, and packets are sent continuously until the timeout period expires or the **stop ping** command on page 22-233 is issued. The default is 5.

The **pattern** parameter specifies the data to use to fill the data portion of the ping packet.

The **sipaddress**, **sipxaddress**, **sosiaddress**, and **sappleaddress** parameters specify the source address to use in ping packets for IP, IPX, OSI, and AppleTalk networks, respectively. If these parameters have already been set, they can be restored to their default “not set” state by specifying:

- 0.0.0.0 or :: for **ipaddress**
- 0:0 for **ipxaddress**
- 0.0 for **appleaddress**

If the source address is not specified, and has not been set using the **set ping** command on page 22-171, the default is to use the address of the interface from which the ping packets are transmitted. For IP addresses, the router’s local interface IP address, if set, is used. Otherwise, the IP address of the interface from which the ping packets are transmitted is used. If the ping request is to an IPv6 link-local address, the **sipaddress** must be on the outgoing interface and cannot be a link-local address.

The **screenoutput** parameter specifies whether the output is sent to the terminal. If **yes** is specified, the response time for each echo reply packet is displayed to the terminal as the reply is received from the destination host (Figure 22-12 on page 22-137). If **no** is specified, the results are stored and not displayed. To view the results, use the **show ping** command on page 22-224. The default is **yes**.

The **timeout** parameter specifies how many seconds to wait for a response to a ping packet, and cannot be zero. The default is 1.

The **tos** parameter specifies the value of the TOS (Type Of Service) field in the IP header of the ping packet. The **tos** parameter is valid for IP addresses, and is ignored for IPv6 addresses. The default is 0.

**Related Commands**

**add ip host**
**ping**
**show ping**
**stop ping**

# set trace

**Syntax**

```
SET TRAce [[IPaddress=]ipadd|domain-name]
    [ADDROnly={No|OFf|ON|Yes}] [MAXTtl=number]
    [MINTtl=number] [NUMber=number]
    [POrt=1..65535] [SCReenoutput={No|OFf|ON|Yes}]
    [SOurce=ipadd] [TIMEOut=number] [TOS=0..25]
```

where:

- *ipadd* is an IPv4 address in dotted decimal notation, a valid IPv6 address or a host name from the host name table.
- domain-name is a valid network domain name.
- *number* is a decimal number.

**Description** This command sets default options for the trace route command.

If there is no default destination and a destination is not specified with the **trace** command on page 22-234, then a trace is not performed and an error message is displayed.

The **ipaddress** parameter specifies the destination IP address. The command traces the route to this IP address. To configure a DNS for the router to use to resolve domain names, use the **add ip dns** command on page 22-68.

The **addronly** parameter specifies whether trace output is presented as IP addresses only, as opposed to IP addresses and their DNS name equivalent. If **on**, output is presented as IP addresses. The default is **yes**.

The **maxttl** parameter specifies the maximum value for the TTL (Time To Live) field in the IP packet, and is used to limit the trace route to a maximum number of hops. The default is 30.

The **minttl** parameter specifies the initial value of the TTL (Time To Live) field in the IP packet, and can be used to skip hops at the start of the route. The default is 1.

The **number** parameter specifies the number of packets to send to each hop. A maximum of 100 packets may be transmitted. The default is 3.

The **port** parameter specifies the UDP destination port number that **trace** uses to transmit the packet to the first hop destination. The destination UDP port number is incremented by one for each subsequent hop so that **trace** can match reply packets to transmitted packets. You should set **port** to a value such that all devices along the trace route will be listening on the appropriate UDP port. The default is 33434.

The **screenoutput** parameter specifies whether the output is sent to the terminal. The default is **yes**.

The **source** parameter specifies the IP address to use as a source address in the packets. If this parameter is not set, the default IPv4 address of the interface is set as the source address. Because this IPv4 address causes a conflict when an IPv6 address is specified in the **ipaddress** parameter, this parameter is required when tracing a route to an IPv6 address.

The **timeout** parameter specifies how long to wait for a response before sending packets to the next hop. If ICMP “unreachable” messages are received within the timeout period, packets are transmitted to the next hop immediately. The default is 3.

The **tos** parameter specifies the value of the TOS (Type Of Service) field in the IP header of the packets being transmitted. The default is 0.

**Related Commands**

**add ip host**
**show trace**
**stop trace**
**trace**

# show ddns

**Syntax** `SHow DDNS`

**Description** This command displays information about DDNS configuration and operation (Figure 22-13, Table 22-13 on page 22-177).

Figure 22-13: Example output from the **show ddns** command

```
DDNS Configure Information:
  Client State ...................... ENABLED
  Debug ............................. DISABLED
  Server ............................ members.dyndns.org
  Port .............................. 80
  User .............................. test
  Password .......................... ***
  system name ....................... dyndns
    hosts ........................... test.dyndns.org
                                      test.homeip.net
  system name ....................... dyndns
   hosts ............................ test.custom.dyndns.org
                                      test.custom.homeip.net
  MaileX ............................ alliedtelesis.co.nz
  Backmx ............................ no
  Wildcard .......................... off
  Offline ........................... no
  Primary WAN Interface ............. vlan1
  Secondary WAN Interface ........... none

DDNS Operation Information:
  Server IP ......................... 0.0.0.0
  IP in DynDns ...................... 202.49.72.10
  Current IP ........................ 202.49.72.10
Update failed - Suggested action:
  Config IP DNS or Set DDNS Serve
```

Table 22-13: Parameters in output of the **show ddns** command

| Parameter | Meaning |
|---|---|
| Client State | Whether the DDNS feature is enabled. |
| Debug | The DDNS debug state. |
| Server | The DDNS server name. |
| Port | The TCP or UDP port number being used for the DDNS update. That is, the port that the packet is sent from. |
| system name | The DynDNS.com system that update messages use: one of dyndns, statdns, or custom, depending on the type of hostname you chose. The router automatically sets the system name to an appropriate value for your hostname type. |
| hosts | A list of host names registered in this system, to be resolved to the router's public address. |
| MaileX | The name of the mail exchanger. |
| Backmx | Whether the MaileX is set as a backup mail exchanger. |
| Wildcard | Whether the wildcard (*) is used when matching host name. |
| Offline | Whether the hostnames are set to offline mode.<br>Refer to the DynDNS website for offline redirection options, available at: http://www.dyndns.com/services/dns/dyndns/faq.html |
| Primary WAN Interface | The IP interface used for the main WAN connection. |
| Secondary WAN Interface | The IP interface used for the backup WAN connection. |
| Server IP | The IP address of Dynamic DNS server resolved by DNS. |
| IP in DynDns | The IP address assigned to the listed host names at the last update. |
| Current IP | The latest IP address assigned to local WAN connection. |
| Update Failed - Suggested action | User action to take if the last update failed. |

**Related Commands**

**activate ddns update**
**disable ddns**
**disable ddns debug**
**enable ddns**
**enable ddns debug**
**set ddns**

# show ip

**Syntax** `SHow IP`

**Description** This command displays general configuration information regarding the router (Figure 22-14, Table 22-14 on page 22-179).

Figure 22-14: Example output from the **show ip** command

```
IP Module Configuration
------------------------------------------------------------

Module Status .................. ENABLED
IP Packet Forwarding ........... ENABLED
IP Echo Reply .................. ENABLED
IP Spoof Check ................. ENABLED
Debugging ...................... DISABLED
IP Fragment Offset Filtering ... ENABLED
Default Name Servers
  Primary Name Server .......... 192.168.1.1 (ppp0)
  Secondary Name Server ........ Not Set
Name Server .................... 192.168.1.1 (ppp0)
Secondary Name Server .......... Not Set
Source-Routed Packets .......... Discarded
Remote IP address assignment ... DISABLED
DNS Relay ...................... DISABLED
IP ARP LOG ..................... ENABLED
IP ARP refresh by hit .......... ENABLED
IP/MAC address disparity........ DISABLED
IP ARP agepoll ................. DISABLED

Routing Protocols

RIP Neighbours ................. 1
EGP Status ..................... DISABLED
Autonomous System Number ....... Not Set
Transfer RIP to EGP ............ DISABLED
ARP aging timer multiplier...... 4 (1024-2048 secs)
Arp wait timeout ............... 1 secs
OSPF Status .................... DISABLED
IGMP Status .................... ENABLED
DVMRP Status ................... ENABLED
PIM Status ..................... DISABLED
BGP Status ..................... ENABLED

Active Routes

Static ......................... 0
Interface ...................... 1
RIP ............................ 4
EGP ............................ 0
OSPF ........................... 0
BGP ............................ 0
Other .......................... 0
Multicast ...................... 5

IP Filter Configuration

Filter 200 ..................... 1 pattern
Total filters .................. 0

Dynamic Interfaces ............. 0
```

Table 22-14: Parameters in output of the **show ip** command

| Parameter | Meaning |
|---|---|
| Module Status | Whether the IP module is enabled. |
| IP Packet Forwarding | Whether the IP forwarding function is enabled and forwarding or disabled and in server mode. |
| IP Echo Reply | Whether replies to echo request messages are enabled. |
| IP Spoof Check | Whether IP checks for spoofed packets. When enabled, IP drops any packets received where the source address matches the IP address of a local interface. |
| Debugging | Whether the IP debugging facility is enabled. |
| IP Fragment Offset Filter | Whether the IP fragment offset filtering is enabled. |
| Default Name Servers | Configuration of default name servers. More detailed information about the default and domain specific name servers can be displayed with the **show ip dns** command. |
| Primary Name Server | IP address of the default primary name server if assigned. If the address was learned using IPCP negotiation, the name of the interface used for the IPCP negotiation is also displayed. |
| Secondary Name Server | IP address of the default secondary name server if assigned. If the address was learned using IPCP negotiation, the name of the interface used for the IPCP negotiation is also displayed. |
| Source-Routed Packets | Whether source-routed packets are forwarded or discarded. |
| Remote IP address assignment | Whether remote IP address assignment is enabled. |
| DNS Relay | Whether the DNS relay agent is enabled. |
| IP ARP LOG | Whether ARP logging is enabled. |
| IP ARP refresh by hit | Whether ARP entry refreshing is enabled |
| IP/MAC address disparity | Whether the router accepts packets with conflicting IP and MAC addresses, and allows ARP entries with multicast MAC addresses. One of “enabled” or “disabled”. |
| IP ARP agepoll | How the router treats dynamic ARP entries once they have timed out. When enabled, the router checks whether devices are still attached to the network before deleting their dynamic ARP entries. |
| RIP Neighbours | Number of RIP neighbours defined. |
| EGP Status | Whether the EGP routing module is enabled. |
| Autonomous System Number | The autonomous system number used by the BGP module, or “Not Set” if an autonomous system number is not assigned. |
| Transfer RIP to EGP | Whether RIP information is transferred to EGP broadcasts. |
| ARP aging timer multiplier | The multiplier value applied to ARP aging timers, and the resulting current range of ARP aging timer values. |
| Arp wait timeout | The amount of time the router waits for a response after it sends an ARP request message, in seconds. |
| OSPF Status | Whether the OSPF routing module is enabled. |

Table 22-14: Parameters in output of the **show ip** command (cont.)

| Parameter | Meaning |
|---|---|
| IGMP Status | Whether the IGMP protocol module is enabled. |
| DVMRP Status | Whether the DVMRP routing module is enabled. |
| PIM Status | Whether the PIM-SM multicast routing module is enabled. |
| BGP Status | Whether the BGP-4 (Border Gateway Protocol version 4) module is enabled. |
| Static | Number of static routes in use. |
| Interface | Number of interface-related routes in use. |
| RIP | Number of RIP-derived routes in use. |
| EGP | Number of EGP-derived routes in use. |
| OSPF | Number of OSPF-derived routes in use. |
| BGP | Number of BGP-derived routes in use. |
| Other | Number of other routes in use. |
| Multicast | Number of multicast forwarding table entries. |
| Filter *n* | Number of patterns in filter *n*. |
| Total Filters | Total defined IP filters. |
| Dynamic Interfaces | Number of dynamic interfaces created by the Asynchronous Call Control (ACC) module when a dial-in user initiates a SLIP or asynchronous PPP connection. |

**Related Commands**

**disable ip**
**disable ip debug**
**disable ip dnsrelay**
**disable ip forwarding**
**disable ip srcroute**
**disable snmp**
**enable ip**
**enable ip debug**
**enable ip dnsrelay**
**enable ip forwarding**
**enable ip srcroute**
**enable snmp**
**set ip arp refresharp**
**set ip nameserver**
**set ip secondarynameserver**

# show ip arp

**Syntax** `SHow IP ARP`

**Description** This command displays the contents of the ARP cache. The ARP cache contains mappings of IP addresses to physical addresses for hosts to which the router has recently routed packets. To have an entry in the ARP cache, a host must have attempted to access another host, and it must have found the physical address by using the ARP protocol (Figure 22-15, Table 22-15).

Figure 22-15: Example output from the **show ip arp** command

```
 Interface     IP Address       Physical Address      ARP Type    Status
---------------------------------------------------------------------------
 eth0         172.16.8.1       AA-00-04-00-2D-08     Dynamic     Active
 eth0         172.16.8.2       AA-00-04-00-28-08     Dynamic     Active
 eth0         172.16.8.34      00-00-0C-02-5A-0A     Dynamic     Active
 eth0         172.16.9.185     08-03-50-37-00-00     Dynamic     Active
 fr0          172.16.240.2     20                    Static      Active
 x25t0        172.16.198.1     Remote1               Static      Active
 eth0         192.168.163.47   FF-FF-FF-FF-FF-FF     Other       Active
 eth0         255.255.255.255  FF-FF-FF-FF-FF-FF     Other       Active
---------------------------------------------------------------------------
```

Table 22-15: Parameters in output of the **show ip arp** command

| Field | Meaning | |
|---|---|---|
| Interface | Interface over which the network device is accessed. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number. | |
| IP Address | IP address of the network device. | |
| Physical Address | Physical address of the network device. For an Ethernet, this is the Ethernet address; for a Frame Relay DLC it is the DLCI; for a MIOX (X.25) circuit it is the circuit name. | |
| ARP Type | Type of entry: | |
| | Static | Added with the **add ip arp** command on page 22-66 |
| | Dynamic | Learned from ARP request/reply message exchanges |
| | Invalid | The interface may not exist |
| | Other | Automatically generated by the system, for example, general IP broadcast and IP subnet/network broadcast addresses are added when the IP module is configured |
| Status | Whether the ARP entry is active or inactive. | |

**Related Commands** **add ip arp**
**delete ip arp**
**set ip arp**

# show ip cache

**Syntax** `SHow IP CAChe`

**Description** This command displays information about the IP address cache when troubleshooting.

Figure 22-16: Example output from the **show ip cache** command

```
IP Address Cache
------------------------------------------------------------------
Entries ................. 284
Max Entries ............. 284
Last Addition ........... 13:54:43 on Tuesday 21-Feb-2006
Last Rejection .......... -

Source           Destination      Interface        Type      Age       Count
----------------------------------------------------------------------------
10.1.1.2         192.168.100.3    eth0-1           Forward   1             3
10.1.1.3         192.168.100.3    eth0-2           Forward   1             3
10.1.1.4         192.168.100.3    eth0-3           Forward   1             3
10.1.1.5         192.168.100.3    eth0-4           Forward   1             3
.
.
.
```

Table 22-16: Parameters in output of the **show ip cache** command

| Parameter | Meaning |
|---|---|
| Entries | Current number of entries in the cache. |
| Max Entries | Maximum number of entries in the cache since the router restarted. |
| Last Addition | Time and date that the last entry was added to the cache. |
| Last Rejection | Time and date that an entry failed to be added to the cache (possibly because the cache was full). |
| Source | Source of the IP address. |
| Destination | Destination of the IP address. |
| Interface | Interface that the IP packet was received on. |
| Type | One of the following:<br>Forward<br>Local<br>GenBcast<br>SpcBcast<br>MultOsp<br>MultLmtd<br>MultNorm<br>MultLocl |
| Age | Age of the entry, which increases over time, but is reduced when the entry is used. |
| Count | Number of times the entry was found. |

**Related Commands** reset ip counter
show ip counter

# show ip counter

**Syntax**

```
SHow IP
    COUnter[={ALL|ARP|CAChe|ICmp|INterface|IP|MUlticast|ROu
    tes|SNmp|UDp}]
```

**Description**

This command displays all or selected parts of the IP MIB. If **all** is specified or no option, then all IP counters are displayed. The MIB can be selectively displayed by specifying one of the following options:

- arp (Figure 22-17, Table 22-17)
- cache (Figure 22-18 on page 22-184, Table 22-18 on page 22-184)
- icmp (Figure 22-19 on page 22-184, Table 22-19 on page 22-184)
- interface (Figure 22-20 on page 22-185, Table 22-20 on page 22-185)
- ip (Figure 22-21 on page 22-186, Table 22-21 on page 22-187)
- multicast (Figure 22-22 on page 22-188, Table 22-22 on page 22-188)
- routes (Figure 22-23 on page 22-189, Table 22-23 on page 22-189)
- snmp (Figure 22-24 on page 22-190, Table 22-24 on page 22-190)
- udp (Figure 22-25 on page 22-191, Table 22-25 on page 22-191)

Figure 22-17: Example output from the **show ip counter=arp** command

```
ARP Counters

  arpRxPkts ............... 0   arpTxPkts .............. 0
  arpRxReqPkts ............ 0   arpTxReqPkts ........... 0
  arpRxRespPkts ........... 0   arpTxRespPkts .......... 0
  arpRxDiscPkts ........... 0   arpTxDiscPkts .......... 0
```

Table 22-17: Parameters in output of the **show ip counter=arp** command

| Parameter | Meaning |
|---|---|
| arpRxPkts | Number of ARP packets received. |
| arpRxReqPkts | Number of ARP request packets received. |
| arpRxRespPkts | Number of ARP Response packets received. |
| arpRxDiscPkts | Number of inbound ARP packets discarded. |
| arpTxPkts | Number of ARP packets transmitted. |
| arpTxReqPkts | Number of ARP request packets transmitted. |
| arpTxRespPkts | Number of ARP Response packets transmitted. |
| arpTxDiscPkts | Number of outbound ARP packets discarded. |

Figure 22-18: Example output from the **show ip counter=cache** command

```
Cache Counters

  hits ............... 304   rejects .............. 0
  deletes .............. 0
```

Table 22-18: Parameters in output of the **show ip counter=cache** command

| Parameter | Meaning |
|---|---|
| hits | Number of times that an entry was found in the cache. |
| rejects | Number of times that an entry could not be added to the cache. |
| deletes | Number of entries removed from the cache before they timed out. |

Figure 22-19: Example output from the **show ip counter=icmp** command

```
ICMP Counters

  inMsgs ................ 0       outMsgs ............... 0
  inErrors .............. 0       outErrors ............. 0
  inDestUnreachs ........ 0       outDestUnreachs ....... 0
  inTimeExcds ........... 0       outTimeExcds .......... 0
  inParamProbs .......... 0       outParamProbs ......... 0
  inSrcQuenchs .......... 0       outSrcQuenchs ......... 0
  inRedirects ........... 0       outRedirects .......... 0
  inEchos ............... 0       outEchos .............. 0
  inEchoReps ............ 0       outEchoReps ........... 0
  inTimestamps .......... 0       outTimestamps ......... 0
  inTimestampReps ....... 0       outTimestampReps ...... 0
  inAddrMasks ........... 0       outAddrMasks .......... 0
  inAddrMaskReps ........ 0       outAddrMaskReps ....... 0
```

Table 22-19: Parameters in output of the **show ip counter=icmp** command

| Parameter | Meaning |
|---|---|
| inMsgs | Number of ICMP packets received. |
| inErrors | Number of ICMP packets received that had ICMP-specific errors (bad ICMP checksums, bad length, etc). |
| inDestUnreachs | Number of ICMP Destination Unreachable packets received. |
| inTimeExcds | Number of ICMP Time Exceeded packets received. |
| inParamProbs | Number of ICMP Parameter Problem packets received. |
| inSrcQuenchs | Number of ICMP Source Quench request packets received. |
| inRedirects | Number of ICMP Redirect request packets received. |
| inEchos | Number of ICMP echo request (ping) packets received. |
| inEchoReps | Number of ICMP echo reply packets received. |
| inTimestamps | Number of ICMP Timestamp request packets received. |
| inTimestampReps | Number of ICMP Timestamp reply packets received. |
| inAddrMasks | Number of ICMP Address Mask request packets received. |
| inAddrMaskReps | Number ICMP Address Mask reply packets received. |

Table 22-19: Parameters in output of the **show ip counter=icmp** command (cont.)

| Parameter | Meaning |
|---|---|
| outMsgs | Number of ICMP packets transmitted. |
| outErrors | Number of ICMP packets that should have been transmitted but were not. |
| outDestUnreachs | Number of ICMP Destination Unreachable packets transmitted. |
| outTimeExcds | Number of ICMP Time Exceeded packets transmitted. |
| outParamProbs | Number of ICMP Parameter Problem packets transmitted. |
| outSrcQuenchs | Number of ICMP Source Quench reply packets transmitted. |
| outRedirects | Number of ICMP Redirect packets transmitted. |
| outEchos | Number of ICMP echo request (ping) packets transmitted. |
| outEchoReps | Number of ICMP echo reply packets transmitted. |
| outTimestamps | Number of ICMP Timestamp request packets transmitted. |
| outTimestampReps | Number of ICMP Timestamp reply packets transmitted. |
| outAddrMasks | Number of ICMP Address Mask request packets transmitted. |
| outAddrMaskReps | Number of ICMP Address Mask reply packets transmitted. |

Figure 22-20: Example output from the **show ip counter=interface** command

```
Interface Counters
--------------------------------------------------------------------------------
Interface       ifInPkts    ifInBcastPkts    ifInUcastPkts     ifInDiscards
Type           ifOutPkts   ifOutBcastPkts   ifOutUcastPkts    ifOutDiscards
--------------------------------------------------------------------------------
eth0               23531            23224              307                0
Static               230                0              230                0

eth1                   0                0                0                0
Static             63289            63289                0                0

ppp0                   0                0                0                0
Static                 0                0                0                0
--------------------------------------------------------------------------------
```

Table 22-20: Parameters in output of the **show ip counter=interface** command

| Parameter | Meaning | |
|---|---|---|
| Interface | Name of the interface (such as ppp0), or "local" for the local IP interface. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number. | |
| Type | Type of interface: | |
| | Static | Permanent interface that is active and in use |
| | Dynamic | Non-permanent interface created, for example by Asynchronous Call Control (ACC), when a dial-in user initiates a SLIP or PPP connection. The interface disappears when the user logs off, when the router is restarted, or when the IP module is reset with the **reset ip** command on page 22-138. |

Table 22-20: Parameters in output of the **show ip counter=interface** command (cont.)

| Parameter | Meaning | |
|---|---|---|
| | Inactive | An inactive interface is a permanent interface that could not attach to the lower-layer (FR, ETH) interface for some reason. The interface is not in use but remains configured and becomes active when the lower-layer attachment succeeds on the next **reset ip** or **restart** command. The most common cause of inactive interfaces is the deletion of the lower-layer interface. Inactive interfaces may be deleted by the manager but cannot be modified. |
| ifInPkts | Number of packets received over the interface. | |
| ifOutPkts | Number of packets transmitted over the interface. | |
| ifInBcastPkts | Number of multicast packets received over the interface. | |
| ifOutBcastPkts | Number of multicast packets transmitted over the interface. | |
| ifInUcastPkts | Number of unicast packets received over the interface. | |
| ifOutUcastPkts | Number of unicast packets transmitted over the interface. | |
| ifInDiscards | Number of packets received over the interface that were discarded. | |
| ifOutDiscards | Number of packets to be transmitted over the interface that were discarded. | |

Figure 22-21: Example output from the **show ip counter=ip** command

```
IP Counters

  inReceives ......... 1005       outRequests ........... 0
  inHdrErrors ........... 0       outDiscards ........... 0
  inAddrErrors .......... 0       outNoRoutes ........... 0
  inUnknownProtos ....... 0       forwDatagrams ........ 33
  inDiscards ............ 0       routingDiscards ....... 0
  inDelivers .......... 972
  reasmReqds ............ 0       fragCreates ........... 0
  reasmOKs .............. 0       fragOKs ............... 0
  reasmFails ............ 0       fragFails ............. 0

IP Gateway Discards
  tinyFragments ......... 0       spoofedPkts .......... 12
  invalHdrOption ........ 0       dirBroadcasts ......... 0
  saSpoofedPkts ......... 0       saBlockedPkts ......... 0
  saEncodeFails ......... 0
```

Table 22-21: Parameters in output of the **show ip counter=ip** command

| Parameter | Meaning |
|---|---|
| inReceives | Number of IP packets the router received. |
| inHdrErrors | Number of IP packets received with header errors. |
| inAddrErrors | Number of IP packets received with address errors. |
| inUnKnownProtos | Number of IP packets received with unsupported protocols. |
| inDiscards | Number of IP packets received but discarded due to resource limitations at the IP level. |
| inDelivers | Number of IP packets received and passed on by the IP software to other modules. |
| reasmReqds | Number of IP packets received that needed reassembly. |
| reasmOKs | Number of IP packets successfully reassembled. |
| reasmFails | Number of reassembly failures. |
| outRequests | Number of IP packets requested to be transmitted by higher layers. |
| outDiscards | Number of output IP packets discarded for one of the following reasons:<br>• There was no ARP entry for the destination IP address.<br>• The destination IP address was invalid, for example, a sub network address or unknown broadcast address.<br>• There was no route to the destination address.<br>• The route was over an IP interface that did not exist or was not enabled.<br>• The packet was a directed broadcast or multicast packet received on an interface not configured to receive directed broadcasts.<br>• The broadcast type is not supported.<br>• The packet was addressed to a VRRP IP for which this router was the master, but did not own the IP address.<br>• IPsec processing of the packet failed.<br>• A route existed over an asynchronous interface, but transmission over the asynchronous interface failed. |
| outNoRoutes | Number of output IP packets discarded because no route existed to the destination. |
| forwDatagrams | Number of IP packets forwarded. |
| routingDiscards | Number of routing entries discarded even though they were valid (possibly to free up buffer space). |
| fragCreates | Number of fragments created. |
| fragOKs | Number of IP packets successfully fragmented. |
| fragFails | Number of IP packets that needed fragmenting but the IP flags field indicated not to fragment. |
| tinyFragments | Number of packets discarded because they were part of a tiny fragment attack. |
| invalHdrOption | Number of packets discarded because they contained an invalid header option. |
| saSpoofedPkts | Number of packets discarded because they claimed to be from a Security Association partner but were not encoded correctly. |

Table 22-21: Parameters in output of the **show ip counter=ip** command (cont.)

| Parameter | Meaning |
|---|---|
| saEncodeFails | Number of packets discarded because the Security Association encoding failed. |
| spoofedPkts | Number of packets discarded because they were spoofed packets. |
| dirBroadcasts | Number of packets discarded because directed broadcasts are not allowed. |
| saBlockedPkts | Number of packets a Security Association discards because they originated from addresses that do not belong to the Security Association. |

Figure 22-22: Example output from the **show ip counter=multicast** command

```
Multicast Counters
------------------------------------------------------------------------
Interface  ifInMultPkts ifInMultDiscard ifOutMultPkts  ifOutMultDiscards
------------------------------------------------------------------------
eth0                123               2           321                 1
eth1               1234               2         12321                 3
-------------------------------------------------------------------------
```

Table 22-22: Parameters in output of the **show ip counter=multicast** command

| Parameter | Meaning |
|---|---|
| Interface | Name of the interface (such as PPP0), or "local" for the local IP interface. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number. |
| ifInMultPkts | Number of multicast packets received over the interface. |
| ifInMultDiscard | Number of multicast packets received over the interface that were discarded. |
| ifOutMultPkts | Number of multicast packets transmitted over the interface. |
| ifOutMultDiscards | Number of multicast packets to be transmitted over the interface that were discarded. |

Figure 22-23: Example output from the **show ip counter=routes** command

```
Route Counters
 IP address       NextHop         Interface  Metric  Octets rcvd   Octets sent
--------------------------------------------------------------------------------
 0.0.0.0          202.36.163.21   eth0          4            984             0
 192.168.19.0     202.36.163.21   eth0          3              0             0
 192.168.39.0     202.36.163.21   eth0          4              0             0
 192.168.42.0     202.36.163.21   eth0          4              0             0
 192.168.119.0    202.36.163.21   eth0          4              0             0
 192.168.255.0    202.36.163.21   eth0          3              0             0
 202.36.163.0     0.0.0.0         eth0          1          81504          1468
 202.49.72.0      202.36.163.21   eth0          2              0             0
 202.49.74.0      202.36.163.21   eth0          3              0             0
 203.97.191.0     202.36.163.21   eth0          3              0             0
--------------------------------------------------------------------------------
```

Table 22-23: Parameters in output of the **show ip counter=routes** command

| Parameter | Meaning |
|---|---|
| IP address | IP address of the remote network pointed to by this route – could be any IP address. |
| NextHop | IP address of the next router on the path to the remote network. Always an address on one of the router's interfaces. |
| Interface | Interface over which the next hop is reached. This field is blank when the next hop is over an addressless PPP interface. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number. |
| Metric | Cost to reach the remote network. If there are two paths to the same network, the one with the lowest metric is used. If both have the same metric, then the first occurrence is taken. |
| Octets rcvd | Number of octets of data received over this route. |
| Octets sent | Number of octets of data transmitted over this route. |

Figure 22-24: Example output from the **show ip counter=snmp** command

```
SNMP Counters
  inPkts ................ 0       outPkts .............. 0
  inBadVersions ......... 0       outTooBigs ........... 0
  inBadCommunityNames ... 0       outNoSuchNames ....... 0
  inBadCommunityUses .... 0       outBadValues ......... 0
  inASNParseErrs ........ 0       outGenErrs ........... 0
  inTooBigs ............. 0       outGetRequests ....... 0
  inNoSuchNames ......... 0       outGetNexts .......... 0
  inBadValues ........... 0       outSetRequests ........0
  inReadOnlys ........... 0       outGetResponses ...... 0
  inGenErrs ............. 0       outTraps ............. 0
  inTotalReqVars ........ 0
  inTotalSetVars ........ 0
  inGetRequests ......... 0
  inGetNexts ............ 0
  inSetRequests ......... 0
  inGetResponses ........ 0
  inTraps ............... 0
```

Table 22-24: Parameters in output of the **show ip counter=snmp** command

| Parameter | Meaning |
|---|---|
| inPkts | Number of SNMP packets the router received. |
| inBadVersions | Number of SNMP packets with a bad version field the router received. |
| inBadCommunityNames | Total number of SNMP PDUs delivered to the SNMP agent that used an unknown SNMP community name. |
| inBadCommunityUses | Total number of SNMP PDUs delivered to the SNMP agent that represented an SNMP operation not allowed by the SNMP community name in the PDU. |
| inASNParseErrs | Total number of ASN.1 parsing errors, either in encoding or syntax, encountered by the SNMP agent when decoding received SNMP PDUs. |
| inTooBigs | Total number of valid SNMP PDUs delivered to the SNMP agent for which the value of the errorStatus component was tooBig. |
| inNoSuchNames | Number of SNMP packets received with an error status of nosuchname. |
| inBadValues | Number of SNMP packets received with an error status of badvalue. |
| inReadOnlys | Number of SNMP packets received with an error status of readonly. |
| inGenErrs | Number of SNMP packets received with an error status of generr. |
| inTotalReqVars | Total number of SNMP MIB objects requested. |
| inTotalSetVars | Total number of SNMP MIB objects that were changed. |
| inGetRequests | Number of SNMP get request packets the router received. |
| inGetNexts | Number of SNMP get Next request packets the router received. |
| inSetRequests | Number of SNMP set request packets the router received. |
| inGetResponses | Number of SNMP get Response packets the router received. |

Table 22-24: Parameters in output of the **show ip counter=snmp** command (cont.)

| Parameter | Meaning |
|---|---|
| inTraps | Number of SNMP trap message packets the router received. |
| outPkts | Number of SNMP packets the router transmitted. |
| outTooBigs | Number of SNMP packets transmitted with an error status of toobig. |
| outNoSuchNames | Number of SNMP packets transmitted with an error status of nosuchname. |
| outBadValues | Number of SNMP packets transmitted with an error status of badvalue. |
| outGenErrs | Number of SNMP packets transmitted with an error status of generror. |
| outGetRequests | Number of SNMP get request response packets transmitted by the router. |
| outGetNexts | Number of get Next response packets the router transmitted. |
| outSetRequests | Number of set request packets the router transmitted. |
| outGetResponses | Number of SNMP get response packets transmitted. |
| outTraps | Number of SNMP Traps the router transmitted. |

Figure 22-25: Example output from the **show ip counter=udp** command

```
UDP Counters

  inDatagrams ....... 307      outDatagrams ........ 0
  inErrors ............ 0      noPorts ............. 6
```

Table 22-25: Parameters in output of the **show ip counter=udp** command

| Parameter | Meaning |
|---|---|
| inDatagrams | Number of UDP packets the router received. |
| inErrors | Number of UDP packets dropped because they contained an error at the UDP layer. |
| outDatagrams | Number of UDP packets the router transmitted. |
| noPorts | Number of UDP packets dropped because their destination port was not known. |

**Related Commands**

reset ip counter
show ip interface
show ip route
show snmp community
show tcp

# show ip debug

**Syntax**

```
SHow IP DEBug[=1..40]
```

**Description**

This command displays selected entries from the IP debug queue. The debug queue is enabled with **enable ip debug** command on page 22-123. Incorrectly formatted IP packet headers are captured for later analysis. The queue can have up to 40 entries, where each entry consists of the first 64 bytes from the packet in question.

If no packet number is specified, the command returns the number of packets in the queue or that no packets have been found.

The following are possible responses to the **show ip debug** command:

```
No packets are currently stored in the debug queue.
<value> packets are currently stored in the debug queue.
```

Some limited analysis of the captured packets is done. The following are possible responses to the **show ip debug=*n*** command, where *n* is a number between 1 and 40, or the maximum number of packets captured so far.

```
Error = Bad destination or source address
Error = Packet length exceeds interface mtu
Error = Bad IP header checksum
Error = Unknown
Error = Packet IP header length too short
Error = Bad IP version
```

An explanation of the possible cause of these problems is beyond the scope of this document.

**Related Commands**

**disable ip debug**
**enable ip debug**
**show ip**
**disable debug active** in Chapter 4, Configuring and Monitoring the System
**show debug active** in Chapter 4, Configuring and Monitoring the System

# show ip dns

**Syntax** `SHow IP DNS`

**Description** This command displays information about the DNS name servers used by the router (Figure 22-26, Table 22-26).

Figure 22-26: Example output from the **show ip dns** command

```
DNS Server Configuration
--------------------------------------------------------------------------------
Domain                         Int/Status  Primary        Secondary     Requests
--------------------------------------------------------------------------------
ANY                            No          192.168.20.1   192.168.10.2       327
mycorp.local.pc                ppp0-1/Up   10.8.0.1       10.8.0.2            29
--------------------------------------------------------------------------------
Cache:
  Maximum entries ................... 250
  Current entries ................... 172 (50912 bytes)
  Timeout (minutes) ................. 4
  Cache hits ........................ 94
```

Table 22-26: Parameters in output of the **show ip dns** command

| Parameter | Meaning |
|---|---|
| Domain | Shows either the domain for which the following DNS server configuration applies or the string "ANY" if the configuration applies to all domains not covered specifically by another set of servers. |
| Int/Status | Interface over which DNS server information is learned; one of "No" if the DNS servers are statically configured, or an interface name if the server is determined via IPCP (over a PPP interface) or DHCP (over an Ethernet interface). When the interface is determined via IPCP or DHCP, this field also shows the interface status; one of "Up"(operational) or "Down"(not operational). |
| Status | Whether the PPP interface over which DNS server information is learned is active. This parameter is present when a PPP interface is displayed for the Interface parameter. |
| Primary | IP address of the primary DNS server used in resolving domain names that match the Domain parameter. When DNS server information is learned over a PPP interface, this parameter shows an IP address when the interface is active; otherwise it shows "Not set". When DNS server information is statically configured but no IP has been specified for the primary DNS server, "Not set" is displayed. |
| Secondary | IP address of the secondary DNS server used in resolving domain names that match the Domain parameter. When DNS server information is learned over a PPP interface, this parameter shows an IP address when the interface is active; otherwise it shows "Not set". When DNS server information is statically configured but no IP has been specified for the secondary DNS server, "Not set" is displayed. |
| Requests | Number of requests for which these name servers have been used. |
| Cache | Configuration of the DNS cache. |
| Maximum Entries | Maximum number of entries allowed in the DNS at any time. |

Table 22-26: Parameters in output of the **show ip dns** command (cont.)

| Parameter | Meaning |
|---|---|
| Current Entries | Number of entries currently in the DNS cache. Also shows the amount of RAM being used by the cache. |
| Timeout | Minutes that an entry can remain in the DNS cache. Entries with a TTL less than this parameter are stored with their own TTL; those with a TTL larger than this parameter are stored for the duration of **timeout**. |
| Cache Hits | Number of DNS requests that have been successfully matched to an entry in the cache. |

**Examples** To display the router's DNS server settings, use the command:

```
sh ip dns
```

**Related Commands**

**add ip dns**
**delete ip dns**
**set ip dns**
**set ip dns cache**
**show ip dns cache**

# show ip dns cache

**Syntax** `SHow IP DNS CAChe`

**Description** This command displays the contents of the router’s DNS cache (Figure 22-27, Table 22-27).

Figure 22-27: Example output from the **show ip dns cache** command

```
DNS Cache                Entries ... 4 (1184 bytes)
--------------------------------------------------------------------------------
Domain Name                                      IP Address        TTL   Matches
  (IPv6 Address)                                                  (Min)
--------------------------------------------------------------------------------
www.example.com                                  192.0.2.29         37     29817
ftp.example.com                                  192.0.2.30          1      1185
users.foobar.net                                 192.0.2.225        60        23
local.mycompany.org                              192.168.11.44      21       102
--------------------------------------------------------------------------------
```

Table 22-27: Parameters in output of the **show ip dns cache** command

| Parameter | Meaning |
|---|---|
| Entries | Number of domain names that currently have a record in the DNS cache, and the amount of RAM the cache is using. |
| Domain Name | Domain that this cache entry applies to. |
| IP Address | IP address to which traffic for the domain is be sent. |
| (IPv6 Address) | IPv6 address to which traffic for the domain is to be sent. |
| TTL | Amount of time, in minutes, that the entry will remain in the cache. |
| Matches | Number of times the cache entry has been used to respond to a DNS request. |

**Examples** To display information about the contents of the DNS cache, use the command:

```
sh ip dns cac
```

**Related Commands**

**add ip dns**
**delete ip dns**
**set ip dns**
**show ip dns**

# show ip filter

**Syntax** `SHow IP FILter[=0..999]`

**Description** This commands displays information about filters. If a filter is specified, the patterns in the filter are displayed. If a filter is not specified, the patterns in all filters are displayed (Figure 22-28, Table 22-28).

Figure 22-28: Example output from the **show ip filter** command

```
IP Filters
--------------------------------------------------------------------------------
No. Filter Type
  Ent. Source Port   Source Address    Source Mask       Session       Size
       Dest. Port    Dest. Address     Dest. Mask        Prot.(C/T)    Options
       Pattern Type  Act/Pol/Pri                         Logging       Matches
 --------------------------------------------------------------------------------
 1  Traffic
    1  Any           192.168.163.23    255.255.255.255   Any           No
       Any           192.168.163.39    255.255.255.255   Any           No
       General       Exclude                             Off           0
    2  Any           192.168.163.24    255.255.255.255   Any           No
       23            192.168.163.39    255.255.255.255   Any           No
       General       Exclude                             Off           0
    3  Any           192.168.163.22    255.255.255.255   Any           No
       23            192.168.163.39    255.255.255.255   Any           No
       General       Exclude                             Off           0
    4  Any           192.168.163.21    255.255.255.255   Any           No
       23            192.168.163.39    255.255.255.255   TCP           No
       General       Exclude                             Off           0
Requests: 636  Passes: 0  Fails: 636
--------------------------------------------------------------------------------
 2  Traffic
    1  Any           192.168.166.2     255.255.255.255   Any           Yes
       Any           192.168.163.39    255.255.255.255   Any           No
       General       Include                             Off           0
    2  Any           192.168.163.21    255.255.255.255   Any           Yes
       23            192.168.163.39    255.255.255.255   TCP           No
       General       Exclude                             Off           0
Requests:  0  Passes:  0  Fails: 0
--------------------------------------------------------------------------------
 3  Traffic
    1  2:34          192.168.163.0     255.255.255.0     Start         Yes
       Any           Any               Any               TCP           No
       General       Include                             Off           0
Requests:  0  Passes:  0  Fails: 0
--------------------------------------------------------------------------------
```

Table 22-28: Parameters in output of the **show ip filter** command

| Parameter | Meaning |
|---|---|
| No. | Number of the filter. |
| Filter Type | The filter type of the pattern; one of "Traffic", "Policy", "Priority", or "Routing". |
| Ent. | Entry number in this filter for the pattern. |
| Source Port | Source IP port for this pattern. |
| Source Address | Source IP address for this pattern. |

Table 22-28: Parameters in output of the **show ip filter** command (cont.)

| Parameter | Meaning | |
|---|---|---|
| Source Mask | Source IP address mask for this pattern. | |
| Session | Type of TCP packet to match when the Pro field contains TCP. | |
| | Start | Matches TCP packets with the SYN bit set and the ACK bit clear |
| | Established | Matches TCP packets with either the SYN bit clear or the ACK bit set |
| | Any | Matches any TCP packet |
| Size | Maximum reassembly size for IP fragments, or “Any” if no maximum size has been set. | |
| Dest. Port | Destination IP port for this pattern. | |
| Dest. Address | Destination IP address for this pattern. | |
| Dest. Mask | Destination IP address mask for this pattern. | |
| Prot. (C/T) | Protocol for this pattern; either ANY, ICMP, OSPF, TCP, or UDP. For the ICMP protocol, the ICMP code and type are also listed. | |
| Options | IP options field for this pattern; either Any, Yes, or No. | |
| Pattern Type | Whether the pattern type is general or specific. | |
| Act/Pol/Pri | Filter action for traffic filters (either Exclude or Include), the policy number for policy filters, or the priority of priority filters. | |
| Logging | Whether matches to this entry generate messages to the router’s Logging facility, and the content of log messages; either Off, Head, Dump, or a number from 4 to 1600. | |
| Matches | Number of IP packets that have matched this pattern. | |
| Requests | Number of IP packets checked against this filter. | |
| Passes | Number of IP packets included by this filter. | |
| Fails | Number of IP packets excluded by this filter. | |

**Related Commands**

**add ip filter**
**delete ip filter**
**set ip filter**

# show ip helper

**Syntax** `SHow IP HElper [COUnter]`

**Description** This command displays information about the state of broadcast forwarding on the router. If no optional parameters are specified, the current configuration is displayed (Figure 22-29, Table 22-29).

If the **counter** parameter is specified, counters for forwarded traffic are displayed (Figure 22-30, Table 22-30 on page 22-199).

Figure 22-29: Example output from the **show ip helper** command

```
IP HELPER Configuration

Status : Enabled
--------------------------------------------
Interface : eth0
  UDP port : 137
     Destination(s) ...... 192.168.2.2
  UDP port : 138
     Destination(s) ...... 192.168.2.2
--------------------------------------------
```

Table 22-29: Parameters in output of the **show ip helper** command

| Parameter | Meaning |
|---|---|
| Status | Whether broadcast forwarding is enabled. |
| Interface | Interface where broadcast UDP packets are received. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), interface names include a hyphen and the logical interface number. |
| UDP port | UDP port number to be matched against UDP broadcast packets that are received. If the port number of a UDP packet matches one on the list, then the packet is forwarded to each of the destination IP addresses. |
| Destination | Destination IP address where matching broadcast UDP packets are forwarded. |

Figure 22-30: Example output from the **show ip helper counter** command

```
IP HELPER Counters

----------------------------------------
Interface : eth0
  InPackets ............ 1
  InNoDestination ...... 0
  Port : 137
    OutPackets ......... 0
  Port : 138
    OutPackets ......... 1
----------------------------------------
```

Table 22-30: Parameters in output of the **show ip helper counter** command

| Parameter | Meaning |
|---|---|
| Interface | Interface where broadcast UDP packets are received. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number. |
| InPackets | Number of broadcast UDP packets received on the interface. Opening a UDP listen port means that all matching UDP packets received on any interface are processed. |
| InNoDestination | Number of broadcast UDP packets received on the interface that did not match a requested port. |
| Port | UDP port number to be matched against UDP broadcast packets that are received. |
| OutPackets | Number of packets forwarded to the destinations listed for the UDP port. |

**Examples** To display the current status of broadcast forwarding, use the command:

```
sh ip he
```

**Related Commands**

add ip helper
delete ip helper
disable ip helper
enable ip helper

# show ip host

**Syntax** `SHow IP HOst`

**Description** This command displays the IP host name table and the IP address of the nameserver, if defined. (Figure 22-31, Table 22-31). A host name can be any arbitrary string and need not be the full domain name. The host name table makes it easier to Telnet to commonly accessed hosts by enabling the user to enter a shorter, easier to remember name for the host rather than the host's full IP address or domain name.

When a host name is specified in the **telnet** command on page 61-34 of Chapter 61, Terminal Server, the entire name is used to match a name in the host name table. All characters are used in the comparison, including nonalphabetic characters if they are present. The comparison is not case-sensitive.

Figure 22-31: Example output from the **show ip host** command

```
  IP Address         Host Name
------------------------------------------------------------
  172.16.8.2         ip4
  172.16.8.3         Zaphod
  172.29.2.8         Admin
------------------------------------------------------------
```

Table 22-31: Parameters in output of the **show ip host** command

| Parameter | Meaning |
|---|---|
| IP Address | IP address of an IP host. |
| Host name | Nickname of the IP host that can be used in the **telnet** command on page 61-34 of Chapter 61, Terminal Server (for example, telnet zaphod). |

**Related Commands**
**add ip host**
**delete ip host**
**set ip host**
**set ip nameserver**
**set ip secondarynameserver**

# show ip icmpreply

**Syntax** `SHow IP ICMPreply`

**Description** This command lists ICMP-configurable reply messages and whether they are enabled (Figure 22-32).

Figure 22-32: Example output from the **show ip icmpreply** command

```
SHOW IP ICMP REPLY MESSAGES
---------------------------------------------------
ICMP REPLY MESSAGES:
  Network Unreachable ................ disabled
  Host Unreachable ................... disabled
  Redirect ........................... enabled
---------------------------------------------------
```

**Related Commands** enable ip icmpreply
disable ip icmpreply

# show ip interface

**Syntax** `SHow IP INTerface[=`*`interface`*`] [COUnter[=MULticast]]`

where *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. Logical Eth interfaces can be 0 to 999. If a logical interface is not specified, 0 is assumed.

**Description** This command displays interface configuration information for interfaces assigned to the IP module with the **add ip interface** command on page 22-80. If an interface is specified, details about it are displayed; otherwise, information for all IP interfaces is displayed (Figure 22-33, Table 22-32 on page 22-203).

A hash symbol (#) after the interface name indicates that the interface has an operational status of "down". Note that interface routes are propagated by RIP when their status at a physical level is "up".

The **counter** parameter displays counters for all interfaces or a specific one (Figure 22-33 on page 22-202, Figure 22-34 on page 22-205, Table 22-33 on page 22-205).

Figure 22-33: Example output from the **show ip interface** command

```
Interface     Type     IP Address       Bc Fr PArp  Filt RIP Met.   SAMode  IPSc
Pri. Filt     Pol.Filt Network Mask     MTU    VJC   GRE  OSPF Met.  DBcast  Mul.
GArp VLAN Tag VLAN Pri InvArp   NotifyOSPFDown
--------------------------------------------------------------------------------
Local         ---      Not set          -  -  -     ---  --         Pass    --
---           ---      Not set          1500  -     ---  --         ---     ---
On   none     none     -        -
Loopback      192.168.10.100            -  n  -     --   -          -       --
---           ---      -                -     -     --   -          -       ---
On   none     none     -        -
eth0-0        Static   192.168.2.1      1  n  On    ---  01         Pass    No
---           ---      255.255.255.0    1500  -     ---  0000000001 No      Rec
On   1        2        -        -
eth0-1        Static   192.101.2.1      1  n  On    ---  01         Pass    No
---           ---      255.255.255.0    1500  -     ---  0000000001 No      Rec
On   3        0        -        -
eth0-2        Static   192.168.23.3     1  n  On    ---  01         Pass    No
---           ---      255.255.255.0    1500  -     ---  0000000001 No      Rec
On   4        0        -        -
ppp0          Static   192.168.2.1      1  n  On    ---  01         Pass    No
---           ---      255.255.255.0    1500  -     ---  0000000001 No      Rec
Off  1        2        -        Yes
--------------------------------------------------------------------------------
```

Table 22-32: Parameters in output of the **show ip interface** command

<table>
<tr><th>Parameter</th><th colspan="2">Meaning</th></tr>
<tr><td>Interface</td><td colspan="2">Name of the interface (such as PPP0), or “local” for the local IP interface. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number.</td></tr>
<tr><td>Type</td><td colspan="2">Type of interface; one of:</td></tr>
<tr><td></td><td>Static</td><td>Active and in use.</td></tr>
<tr><td></td><td>Dynamic</td><td>Non-permanent interface created, for example by Asynchronous Call Control (ACC), when a dial-in user initiates a SLIP or PPP connection. This interface disappears when the user logs off, when the router is restarted, or when the IP module is reset with the <b>reset ip</b> command on page 22-138.</td></tr>
<tr><td></td><td>Inactive</td><td>Permanent interface that could not attach to the lower-layer (FR, PPP, ETH, etc) interface for some reason. This interface is not in use but remains configured and becomes active when the lower-layer attachment succeeds on the next <b>reset ip</b> or <b>restart</b> command. The most common cause of inactive interfaces is the deletion of the lower-layer interface. Inactive interfaces can be deleted by the manager but cannot be modified.</td></tr>
<tr><td></td><td>Loopback</td><td>Virtual interface that is not attached to a physical interface.</td></tr>
<tr><td>IP Address</td><td colspan="2">IP address assigned to this interface. For an interface configured with DHCP, this field shows the value assigned by DHCP, or 0.0.0.0 if a DHCP reply has not yet been received.</td></tr>
<tr><td>Bc</td><td colspan="2">This parameter is set to 0 if an all ‘0’ broadcast is required and ‘1’ otherwise. It defaults to ‘1’.</td></tr>
<tr><td>PArp</td><td colspan="2">Whether this interface supports proxy ARP and if ARP responses will be generated if a default route exists; one of “On” (respond to ARP Requests only if a specific route exists), “Loc” (responds to ARP Requests if a specific route exists, including ARP requests for hosts within a subnet), “Off”, or “Def” (respond to ARP Requests if a specific route or a default route exists). This option is valid for Eth or VLAN interfaces.</td></tr>
<tr><td>Fr</td><td colspan="2">Whether packets larger than the interface MTU are fragmented, which overrides the “Do not fragment” bit.</td></tr>
<tr><td>Filt</td><td colspan="2">Number of the traffic filter applied to the interface, if any assigned.</td></tr>
<tr><td>RIP Met.</td><td colspan="2">RIP metric associated with transmitting packets over this interface.</td></tr>
<tr><td>SAMode</td><td colspan="2">Whether packets that do not belong to a security association assigned to the interface are forwarded or discarded.</td></tr>
<tr><td>IPSc</td><td colspan="2">Whether an IPSec policy is attached to the interface.</td></tr>
<tr><td>Pri. Filt</td><td colspan="2">Number of the priority filter applied to the interface, if any.</td></tr>
<tr><td>Pol.Filt</td><td colspan="2">Number of the policy filter applied to the interface, if any.</td></tr>
<tr><td>Network Mask</td><td colspan="2">Subnet mask assigned to the IP address of this interface. For an interface configured with DHCP, this field shows the value assigned by DHCP, or 0.0.0.0 if a DHCP reply has not been received.</td></tr>
<tr><td>MTU</td><td colspan="2">Maximum packet size that can be transmitted over this interface.</td></tr>
<tr><td>VJC</td><td colspan="2">Whether Van Jacobson’s header compression is active on the interface. This option is valid for PPP interfaces.</td></tr>
</table>

Table 22-32: Parameters in output of the **show ip interface** command (cont.)

| Parameter | Meaning | |
|---|---|---|
| GRE | Number of the GRE entity associated with the interface, if any. | |
| OSPF Met. | OSPF metric associated with transmitting packets over this interface. | |
| DBcast | Whether network and subnet broadcasts are forwarded to the network attached to the interface. | |
| Mul. | How multicast packets are handled on the interface: | |
| | On | Sent and received |
| | Rec | Received but not sent |
| | Snd | Sent but not received |
| | Off | Neither sent nor received |
| GArp | Whether the interface accepts or rejects gratuitous ARP messages. | |
| VLAN Tag | VID (VLAN Identifier) included in the header of each frame transmitted over the Eth interface. | |
| VLAN Pri | The 802.1p User Priority value that the router writes into the header of each VLAN-tagged frame transmitted over the Eth interface. | |
| InvArp | Whether inverse ARP is enabled on ATM or Frame Relay interfaces. Note that Frame Relay interfaces always operate with inverse ARP enabled. | |
| NotifyOSPFDown | Whether IP sends OSPF a notification message when an on-demand PPP link goes down. | |
| | Yes | Displays when the interface is a PPP link, and **notifyospfdown** is set to **yes**. This indicates that OSPF receives a notification from IP when the interface goes down. This also displays by default for non-PPP interface. |
| | No | Displays when the interface is a PPP link, and **notifyospfdown** is set to **no**. For on-demand PPP links, this indicates that OSPF does not receive a notification from IP when the interface goes down. For other PPP links, the router sends the notifications even if this is set to **no**. |

Figure 22-34: Example output from the **show ip interface counter** command

```
IP Interface Counters
--------------------------------------------------------------------------------
Interface       ifInPkts    ifInBcastPkts    ifInUcastPkts     ifInDiscards
Type           ifOutPkts   ifOutBcastPkts   ifOutUcastPkts    ifOutDiscards
--------------------------------------------------------------------------------
eth0               23531            23224              307                0
Static               230                0              230                0

eth1                   0                0                0                0
Static             63289            63289                0                0
```

Table 22-33: Parameters in output of the **show ip interface counter** command

| Parameter | Meaning | |
|---|---|---|
| Interface | The name of the interface (such as PPP0), or "local" for the local IP interface. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen, and the logical interface number. | |
| Type | Type of interface: | |
| | Static | Active and in use |
| | Dynamic | Non-permanent interface created, for example by Asynchronous Call Control (ACC), when a dial-in user initiates a SLIP or PPP connection. This interface disappears when the user logs off, when the router is restarted, or when the IP module is reset with the **reset ip** command on page 22-138. |
| | Inactive | Permanent interface that could not attach to the lower-layer interface for some reason. This interface is not in use but remains configured and becomes active when the lower-layer attachment succeeds on the next **reset ip** or **restart** command. The most common cause of inactive interfaces is the deletion of the lower-layer interface. Inactive interfaces can be deleted by the manager but cannot be modified. |
| ifInPkts | Number of packets received over the interface. | |
| ifOutPkts | Number of packets transmitted over the interface. | |
| ifInBcastPkts | Number of multicast packets received over the interface. | |
| ifOutBcastPkts | Number of multicast packets transmitted over the interface. | |
| ifInUcastPkts | Number of unicast packets received over the interface. | |
| ifOutUcastPkts | Number of unicast packets transmitted over the interface. | |
| ifInDiscards | Number of packets received via the interface that were discarded. | |
| ifOutDiscards | Number of packets to be transmitted over the interface that were discarded. | |

**Related Commands**

**add ip interface**
**delete ip interface**
**disable ip interface**
**enable ip interface**
**reset ip interface**
**set ip interface**
**show ip counter**

# show ip nat

**Syntax** `SHow IP NAT [COUnter] [SUMmary]`

**Description** This command displays information about the configuration of NAT on the router, or counters for traffic handled by NAT. If no optional parameters are specified, information about the configuration of NAT is displayed (Figure 22-35, Table 22-34 on page 22-207).

If **counter** is specified, traffic counters for packets processed by NAT are displayed (Figure 22-36 on page 22-208, Table 22-35 on page 22-209). If **summary** is specified, only summary information is displayed.

Figure 22-35: Example output from the **show ip nat** command

```
IP NAT Configuration

Status  : Enabled
Logging : Fails
Enhanced Fragment Handling: udp
Maximum Packet Fragments : 20
-------------------------------------------------------------------------------
Private IP : 10.20.20.0 - 10.20.20.255
Global IP  : 192.168.34.96
  Method .................. Dynamic ENAT
  Number of entries ....... 5
  Current port ............ 5062
  ENAT static configurations:
    Protocol  PrivateIP:Port        GlobalIP:Port
    TCP       10.20.20.5:23         192.168.34.97:23
    TCP       10.20.20.4:23         192.168.34.96:23
  Current entries:
    Protocol  PrivateIP:Port        GlobalIP:Port         DestinationIP:Port
    TCP       10.20.20.4:53330      192.168.34.96:5027    202.1.1.20:23
      Start time ............. 23:01:11 04-Mar-1997
      TCP state .............. established
      Minutes to deletion ..... 1438
    TCP       10.20.20.4:23         192.168.34.96:23      202.1.1.56:1025
      Start time ............. 22:58:29 04-Mar-1997
      TCP state .............. established
      Minutes to deletion ..... 1435
    TCP       10.20.20.10:1024      192.168.34.96:5013    202.1.1.20:21
      Start time ............. 23:00:18 04-Mar-1997
      TCP state .............. established
      Minutes to deletion ..... 1437
    ICMP      10.20.20.4:19         192.168.34.96:5039    202.1.1.42
      Start time ............. 23:02:27 04-Mar-1997
      ICMP type .............. Echo request
      Minutes to deletion ..... 2
    UDP       10.20.20.4:2051       192.168.34.96:5020    202.1.1.20:53
      Start time ............. 23:01:11 04-Mar-1997
      Minutes to deletion ..... 3
-------------------------------------------------------------------------------
```

Table 22-34: Parameters in output of the **show ip nat** command

| **Parameter** | **Meaning** |
|---|---|
| Status | Whether NAT is enabled. |
| Logging | NAT events being logged. |
| Enhanced Fragment Handling | A list of the protocol types for which large fragmented packets can be handled when IP NAT is enabled. If "other" is listed, protocols that are not ICMP or UDP (or TCP) are permitted to send large fragmented packets. If "none" is listed, no protocols are allowed to send large fragmented packets. If a protocol is not listed, or "none" is listed, then the default fragment constraints apply, which means that the IP packet must consist of no more than 8 fragments with a total of 1780 bytes of data. |
| Maximum Packet Fragments | Maximum number of fragments that a packet may consist of when enhanced fragment handling is enabled. |
| Private IP | IP address of a host or a range of them on the private network. |
| Global IP | Officially assigned global IP address or a range of them. |
| Global interface | Interface connected to the Internet with an officially assigned global IP address. This is displayed for interface ENAT entries. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number. |
| Method | Method of translation; either Static NAT, Dynamic NAT, Static ENAT, Dynamic ENAT, or Interface ENAT. |
| Number of entries | Number of current TCP sessions, UDP flows, or ICMP requests currently mapped by NAT for an interface. |
| Current port | Last assigned unique port or sequence number NAT uses for the interface. This is displayed for ENAT entries. |
| Protocol | Protocol or IP protocol number. of the port. |
| PrivateIP:Port | IP address of a host on the private network and the local port on that host of the session or flow. |
| GlobalIP:Port | IP address and port where the private IP address and port are translated before being forwarded. |
| DestinationIP:Port | Destination IP address and port where the session or flow is forwarded. |
| Start time | Time the session or flow started; not displayed for summary information. |
| TCP state | For TCP sessions, the state of the TCP session; not displayed for summary information. |
| ICMP type | For ICMP flows, this is the type of the ICMP frame that started the flow; not displayed for summary information. |
| Minutes to deletion | Time before the session or flow when information is terminated in the absence of any further traffic; not displayed for summary information. |

Figure 22-36: Example output from the **show ip counter** command

```
IP NAT Counters

--------------------------------------------------------------------------------
Private IP : 10.20.20.0 - 10.20.20.255
Global IP  : 192.168.34.96
  Total packets received from private address(es) ....... 256
  Total packets received by global address(es) .......... 290
  Number of cache hits from private address(es) ......... 247
  Number of cache hits from global address(es) .......... 284
  Number of entries created for configuration ........... 10
  Number of dropped packets due to no match ............. 0
  Number of unknown IP protocols packets dropped ........ 0
  Number of unknown ICMP type packets dropped ........... 0
  Number of dropped ICMP packets ........................ 0
  Number of spoofing packets for private address(es) .... 0
  Number of dropped packets as global address zero ...... 0
  Number of dropped packets due to no spare entries ..... 0
  Number of FTP port commands processed ................. 1
  Number of FTP port commands dropped ................... 0
  ENAT static configurations:
    Protocol  PrivateIP:Port        GlobalIP:Port         Number hits
    TCP       10.20.20.5:23         192.168.34.97:23      0
    TCP       10.20.20.4:23         192.168.34.96:23      1
  Current entries:
    Protocol  PrivateIP:Port        GlobalIP:Port         DestinationIP:Port
    TCP       10.20.20.4:53330      192.168.34.96:5027    202.1.1.20:23
      Packets from private IP ............. 92
      Octets from private IP .............. 3776
      Packets to private IP ............... 96
      Octets to private IP ................ 4234
    TCP       10.20.20.4:23         192.168.34.96:23      202.1.1.56:1025
      Packets from private IP ............. 79
      Octets from private IP .............. 3455
      Packets to private IP ............... 102
      Octets to private IP ................ 4165
    TCP       10.20.20.10:1024      192.168.34.96:5013    202.1.1.20:21
      Packets from private IP ............. 16
      Octets from private IP .............. 712
      Packets to private IP ............... 16
      Octets to private IP ................ 999
    ICMP      10.20.20.4:19         192.168.34.96:5039    202.1.1.42
      Packets from private IP ............. 1
      Octets from private IP .............. 56
      Packets to private IP ............... 0
      Octets to private IP ................ 0
    UDP       10.20.20.4:2051       192.168.34.96:5020    202.1.1.20:53
      Packets from private IP ............. 1
      Octets from private IP .............. 67
      Packets to private IP ............... 1
      Octets to private IP ................ 134
--------------------------------------------------------------------------------
```

Table 22-35: Parameters in the output of the **show ip nat counter** command

| Parameter | Meaning |
|---|---|
| Private IP | IP address of a host or a range of them on the private network. |
| Global IP | Officially assigned global IP address or range of them. |
| Global interface | Interface connected to the Internet with an officially assigned global IP address. This is displayed for interface ENAT entries. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number. |
| Interface | Interface associated with the private IP network |
| Total packets received from private address(es) | Number of packets received from hosts with the private IP address or range of addresses. |
| Total packets received by global address(es) | Number of packets received from external Internet hosts addressed to the global IP address or range of addresses. |
| Number of cache hits from private address(es) | Number of IP packets received from the private network that matched a known session or flow. |
| Number of cache hits from global address(es) | Number of IP packets received from the global Internet that matched a known session or flow. |
| Number of entries created for configuration | Number of sessions or flow entries created for this private network address or range. |
| Number of dropped packets due to no match | Number of IP packets dropped because there was not a current flow or session started to map the packet to its destination. |
| Number of unknown IP protocols packets dropped | Number of IP packets dropped because they contained an IP protocol other than the protocols listed on the *Protocols* field. |
| Number of unknown ICMP type packets dropped | Number of ICMP packets dropped because the ICMP type field was set to an unknown value. |
| Number of dropped ICMP packets | Number of ICMP packets dropped because there was no known destination for the packet. |
| Number of spoofing packets for private address(es) | Number of packets dropped because they were directly addressed to the private network. |
| Number of dropped packets as global address zero | Number of packets dropped because the global IP address for the interface was set to zero. This may happen on a interface ENAT where the IP address has still to be determined. The router buffers a limited number of packets while trying to bring up the link but drops packets and increment this counter. |
| Number of dropped packets due to no spare entries | Number of IP packets that failed to create a new session or flow entry and were dropped because the router reached the maximum number of session or flow entries. |
| Number of FTP port commands processed | Number of FTP port commands processed. Each FTP port command requires special processing by NAT because the IP address of the source is embedded as ASCII text. |
| Number of FTP port commands dropped | Number of FTP port commands that could not be translated successfully. |
| Protocol | Protocol or IP protocol number of the port. |

Table 22-35: Parameters in the output of the **show ip nat counter** command

| Parameter | Meaning |
|---|---|
| PrivateIP:Port | IP address of a host on the private network and the local port on that host of the session or flow. |
| GlobalIP:Port | IP address and port to which the private IP address and port are translated before forwarding. |
| Number Hits | IP protocol number of sessions or flows that have started to the specified entry. |
| DestinationIP:Port | Destination IP address and port where the session or flow is forwarded. |
| Packets from private IP | Number of packets NAT received for the entry from the host on the private network. |
| Octets from private IP | Number of octets NAT received for the entry from the host on the private network. |
| Packets to private IP | Number of packets NAT sent to the host on the private network for this session or flow. |
| Octets to private IP | Number of octets NAT sent to the host on the private network for this session or flow. |

**Examples** To show the NAT configuration, use the command:

```
sh ip nat
```

**Related Commands**

**add ip nat**
**delete ip nat**
**disable ip nat**
**enable ip nat**

# show ip pool

**Syntax** `SHow IP POOL[=pool-name] [IPaddress=ipadd[-ipadd]] [SUMmary]`

where:

- *pool-name* is a string 1 to 15 characters long. Valid characters are any printable characters. If *pool-name* contains spaces, it must be in double quotes.
- *ipadd* is an IP address in dotted decimal notation.

**Description** This command displays information about a single IP address pool or all IP address pools.

The **pool** parameter specifies the name of an existing pool to display. If a value is not specified, information for all defined IP pools is displayed (Figure 22-37, Table 22-36).

The **ip** parameter limits the display to a specific IP address or range of IP addresses from the pool.

If **summary** is specified, summary information is displayed (Figure 22-38 on page 22-212).

Figure 22-37: Example output from the **show ip pool** command.

```
IP Pool
--------------------------------------------------------------------------------
Pool Name: dialin ( 192.168.1.1 - 192.168.1.8 )
Number of requests ........................ 102
Request successes ......................... 101
Request failures .......................... 1
Number in use ............................. 5
IP Address Interface Status Start Time End time
192.168.1.1 PPP0 inuse 24-Jun-1999 15:21:58
192.168.1.2 PPP1 free  24-Jun-1999 10:02:04 24-Jun-1999 16:23:50
192.168.1.3 PPP2 inuse 24-Jun-1999 15:32:17
192.168.1.4 PPP3 inuse 24-Jun-1999 15:36:01
192.168.1.5 PPP4 inuse 24-Jun-1999 15:37:46
192.168.1.6 PPP5 inuse 24-Jun-1999 15:51:06
192.168.1.7 PPP6 free  24-Jun-1999 15:59:51 24-Jun-1999 16:03:11
192.168.1.8      free never used
--------------------------------------------------------------------------------
```

Table 22-36: Parameters in output of the **show ip pool** command

| Parameter | Meaning |
|---|---|
| Pool Name | Name of the IP address pool and the IP addresses assigned to the pool. |
| Number of requests | Total number of requests to allocate an IP address from the specified pool. |
| Request successes | Number of successful requests to allocate an IP address from the specified pool. |
| Request failures | Number of failed requests to allocate an IP address from the specified pool. |

Table 22-36: Parameters in output of the **show ip pool** command (cont.)

| **Parameter** | **Meaning** |
|---|---|
| Number in use | Number of IP addresses currently in use for the specified pool. |
| IP Address | IP address in the specified pool. |
| Interface | Interface that last requested the IP address. |
| Status | Whether the IP address is in use or free. |
| Start Time | Date and time the IP address was allocated from the pool. |
| End Time | Data and time the IP address was released back to the pool. |

Figure 22-38: Example output from the **show ip pool summary** command

```
IP Pool
--------------------------------------------------
Pool Name: dialin ( 192.168.1.1 - 192.168.1.16 )
Number of requests ........................ 102
Request successes ......................... 101
Request failures .......................... 1
Number in use ............................. 5
--------------------------------------------------
```

**Examples** To display detailed information about the IP address pool named “dialin”, use the command:

```
sh ip pool=dialin
```

**Related Commands** create ip pool
destroy ip pool

# show ip route

**Syntax** `SHow IP ROUte[=ipadd] [{GENeral|CAChe|COUnt|FULl}]`

where *ipadd* is an IP address in dotted decimal notation

**Description** This command displays information about the IP route table. If no optional parameters are specified, the contents of the route table is displayed (Figure 22-39 on page 22-214, Table 22-37 on page 22-214). If **route** is specified with an IP address that does not contain the wildcard character ("*"), the display lists all routes that can be used to reach the specified destination address, including the default route 0.0.0.0. If **route** is specified with an IP address that ends with the wildcard character ("*"), the display lists all routes beginning with the specified address. The wildcard character can be used to replace a complete number in the address, but not part of a number. For example, 192.168.*.* is valid and displays all routes in the route table that start with 192.168, but 192.168.12*.* is not valid.

This command shows the "best" routes—routes whose outgoing Layer 2 interface is up. The exceptions are interface and static routes, which are always displayed. If the outgoing Layer 2 interface for the route is down, a "#" character is displayed after the Layer 2 interface name. Note that interface routes are only propagated by RIP when their status at a physical level is up.

If **general** is specified, summary information is displayed (Figure 22-40 on page 22-215, Table 22-38 on page 22-215).

If **cache** is specified, the contents of the route cache is displayed (Figure 22-41 on page 22-215, Table 22-39 on page 22-215). If **route** is also specified with an IP address, routes in the route cache are displayed that were used to forward packets to the destination specified by the IP address.

If **count** is specified, summary information about the numbers of octets received and transmitted via each route is displayed (Figure 22-42 on page 22-216, Table 22-40 on page 22-216).

If **full** is specified, all routes in the route table regardless of whether the outgoing Layer 2 interface is up or down are displayed (Figure 22-43 on page 22-216 and Table 22-37 on page 22-214). Routes whose outgoing Layer 2 interface is down are marked with the "#" character after the Layer 2 interface name.

Figure 22-39: Example output from the **show ip route** command

```
IP Routes
--------------------------------------------------------------------------------
Destination       Mask              NextHop              Interface           Age
DLCI/Circ.        Type     Policy   Protocol    Tag      Metrics      Preference
--------------------------------------------------------------------------------
0.0.0.0           0.0.0.0           202.36.163.21        vlan1                 1
-                 remote   0        rip         -        5                   100
10.0.0.0          255.0.0.0         0.0.0.0              vlan1                 4
-                 direct   0        interface   -        1                     0
11.0.1.0          255.255.255.0     10.42.0.22           vlan1                 4
-                 direct   0        static      -        1                    60
11.0.2.0          255.255.255.0     10.42.0.22           vlan1                 4
-                 direct   0        static      45535    1                    60
192.168.69.0      255.255.255.0     202.36.163.35        vlan1                 0
-                 remote   0        rip         -        2                   100
192.168.201.0     255.255.255.0     202.36.163.21        vlan1                 1
-                 remote   0        rip         -        5                   100
192.168.202.0     255.255.255.0     202.36.163.21        vlan1                 1
-                 remote   0        rip         -        6                   100
--------------------------------------------------------------------------------
```

Table 22-37: Parameters in output of the **show ip route** and the **show ip route full** command

| **Parameter** | **Meaning** |
|---|---|
| Destination | IP address of the destination network. |
| Mask | Subnet mask for the route. |
| NextHop | IP address of the next router on the route to the destination, or the *ifIndex* of an addressless PPP interfaces in dotted decimal notation. |
| Interface | Interface over which the destination network can be reached, followed by a “#” if the interface is down. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number. |
| Age | Time in seconds that the route has been known. |
| Circuit/DLCI | Circuit name or DLCI number if this is an X.25 or Frame Relay interface. |
| Type | Whether the route is remote, direct, or another kind. |
| Policy | Policy number of this route. |
| Protocol | Whether the protocol that determines the route is interface (automatically created when the interface is created), static (manually created), RIP, or OSPF. |
| Tag | A number to identify the route. You can match against this number in a route map and only import the appropriately-tagged routes into BGP. |
| Metrics | Routing metric (cost) to reach the destination network. |
| Preference | Routing preference value. Routes with a high preference (low value) are used before routes with a low preference (high value). |

Figure 22-40: Example output from the **show ip route general** command

```
IP Route General Information
----------------------------
Number of routes .............. 12
Cache size ..................... 1024
Source route byte counting ..... no
Route debugging ................ no
Multipath routing .............. yes
```

Table 22-38: Parameters in output of the **show ip route general** command

| Parameter | Meaning |
|---|---|
| Number of routes | Number of routes in the route table. |
| Cache size | Size of the route cache (the number of entries). |
| Source route byte counting | Whether source route byte counting is enabled. |
| Route debugging | Whether route debugging is enabled. |
| Multipath routing | Whether multipath route is enabled. |

Figure 22-41: Example output from the **show ip route cache** command

```
IP Route Cache
--------------------------------------------------------------------------------
Destination      Route            Route mask       Nexthop          Interface
--------------------------------------------------------------------------------
202.36.163.4     202.36.163.0     255.255.255.192  0.0.0.0          eth0
202.36.163.5     202.36.163.0     255.255.255.192  0.0.0.0          eth0
202.36.163.6     202.36.163.0     255.255.255.192  0.0.0.0          eth0
202.36.163.11    202.36.163.0     255.255.255.192  0.0.0.0          eth0
202.36.163.21    202.36.163.0     255.255.255.192  0.0.0.0          eth0
202.36.163.31    202.36.163.0     255.255.255.192  0.0.0.0          eth0
202.36.163.36    202.36.163.0     255.255.255.192  0.0.0.0          eth0
202.36.163.51    202.36.163.0     255.255.255.192  0.0.0.0          eth0
202.36.163.61    202.36.163.0     255.255.255.192  0.0.0.0          eth0
202.36.163.5     202.36.163.0     255.255.255.192  0.0.0.0          eth0
            hits:        875          misses:         11
--------------------------------------------------------------------------------
```

Table 22-39: Parameters in output of the **show ip route cache** command

| Parameter | Meaning |
|---|---|
| Destination | Destination IP address. |
| Route | Route used to forward packets to the destination IP address. |
| Route mask | Network mask for the route. |
| NextHop | Next hop on the route. |
| Interface | Interface over which the destination network can be reached. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number. |

Figure 22-42: Example output from the **show ip route count** command

```
Route Counters

 IP address       NextHop         Interface  Metric   Octets rcvd   Octets sent
--------------------------------------------------------------------------------
 192.168.1.0      202.36.163.21   eth1          1               0             0
 192.168.1.0      202.36.163.21   eth1          1               0             0
 192.168.1.64     202.36.163.21   eth1          1               0             0
 192.168.1.128    202.36.163.21   eth1          1               0             0
 192.168.1.192    202.36.163.21   eth1          1               0             0
 192.168.1.208    202.36.163.21   eth1          1               0             0
--------------------------------------------------------------------------------
```

Table 22-40: Parameters in output of the **show ip route count** command

| **Parameter** | **Meaning** |
|---|---|
| IP address | IP address of the destination to which packets were transmitted using this route. |
| NextHop | IP address of the next router on the route to the destination. |
| Interface | Interface over which the destination network can be reached. When multihoming is enabled (two or more logical interfaces have been assigned to a single Layer 2 interface), all interface names include a hyphen and the logical interface number. |
| Metric | Routing metric (cost) to reach the destination network. |
| Octets rcvd | Number of octets received through this route. |
| Octets sent | Number of octets transmitted through this route. |

Figure 22-43: Example output from the **show ip route full** command

```
IP Routes
--------------------------------------------------------------------------------
Destination       Mask              NextHop              Interface           Age
DLCI/Circ.        Type     Policy   Protocol             Metrics      Preference
--------------------------------------------------------------------------------
192.168.1.0       255.255.255.0     0.0.0.0              eth0#               166
-                 direct   0        interface            1                     0
192.168.2.0       255.255.255.0     0.0.0.0              eth0                166
-                 direct   0        interface            1                     0
192.175.176.0     255.255.255.0     192.168.1.1          eth1#               137
-                 remote   0        rip                  16                  100
--------------------------------------------------------------------------------
```

**Related Commands**

**add ip route**
**delete ip route**
**set ip route**

# show ip route multicast

**Syntax** `SHow IP ROUte MULticast`

**Description** This command displays information about the IP multicast forwarding table (Figure 22-44, Table 22-41).

Figure 22-44: Example output from the **show ip route multicast** command

```
Source Subnet   Group       Prot     Uptime   InPort     Expire
   Outports
-------------------------------------------------------------
192.168.1.10    224.1.1.1   DVMRP     11      vlan       140
   vlan2
```

Table 22-41: Parameters in output of the **show ip route multicast** command

| Parameter | Meaning |
|---|---|
| Source Subnet | Host that sources multicast datagrams addressed to the specified groups. |
| Group | Class D IP address to which multicast datagrams are addressed. Note that a given Source may send packets to many different Multicast Groups. |
| Prot | Multicast routing protocol that contributes this forwarding entry. |
| Uptime | The total time in seconds that this route has been up for. If the route is continually used, the Expire timer will be continuously reset (refreshed), and the route will never be (aged out of the route table. |
| InPort | Parent port for the (source, group) pair. |
| OutPorts | Child ports over which multicast datagrams for the (source, group) pair are forwarded. |
| Expire | The time, in seconds, till the route expires and is remote from the route table. This timer is restarted every time the route is used. |

**Examples** To display the forwarding information for multicast groups, use the command:

```
sh ip rou mul
```

**Related Commands** show dvmrp
show pim

# show ip route preference

**Syntax**

`SHow IP ROUte PREFerence`

**Description**

This command displays information about the current IP route table preferences for each of the routing protocols (Figure 22-45, Table 22-42).

Figure 22-45: Example output from the **show ip route preference** command

```
IP Route Preference
------------------------------------------------------------
 Protocol                            Preference
 -----------------------------------------------------------
 RIP ............................... 100 (default)
 OSPF-INTRA ........................ 10 (default)
 OSPF-INTER ........................ 11 (default)
 OSPF-EXT1 ......................... 97
 OSPF-EXT2 ......................... 98
 OSPF-OTHER ........................ 99
 BGP-INT ........................... 170 (default)
 BGP-EXT ........................... 170 (default)
------------------------------------------------------------
```

Table 22-42: Parameters in output of the **show ip route preference** command

| Parameter | Meaning |
|---|---|
| Protocol | Available routing protocols. |
| Preference | Preference value for the routing protocol - a larger preference value indicates a less desirable routing protocol. |

**Related Commands**

**set ip route preference**

# show ip route template

**Syntax** `SHow IP ROUte TEMPlate[=name]`

where *name* is a string 1 to 31 characters long, and is not case-sensitive. Valid characters are any printable character. If *name* contains spaces, it must be in double quotes.

**Description** This command displays information about the specified or all IP route templates. If a template is not specified, summary information about all IP route templates is displayed (Figure 22-46, Table 22-43).

If a name is specified, details are displayed for it (Figure 22-47, Table 22-44 on page 22-220).

Figure 22-46: Example output from the **show ip route template** command

```
Template                                Interface
-----------------------------------------------------------------
branch_office                           vlan1
home                                    vlan1
-----------------------------------------------------------------
```

Table 22-43: Parameters in output of the **show ip route template** command

| Parameter | Meaning |
|---|---|
| Template | Name of the IP route template. |
| Interface | IP interface specified by the IP route template. |

Figure 22-47: Example output from the **show ip route template** command for a specific template

```
IP route template .................. branch_office
Interface .......................... 0
Next hop ........................... 192.168.23.3
Rip metric ......................... DEFAULT (1)
Ospf metric ........................ DEFAULT (FFFFFFFF)
Policy ............................. DEFAULT (0)
Preference ......................... 90
Dlci ............................... 67
```

Table 22-44: Parameters in output of the **show ip route template** command for a specific template

| Parameter | Meaning |
|---|---|
| IP route template | Name of the IP route template. |
| Interface | IP interface specified by the IP route template. |
| Next hop | Next hop specified by the IP route template. |
| Rip metric | RIP metric specified by the IP route template. |
| Ospf metric | OSPF metric specified by the IP route template. |
| Policy | Policy specified by the IP route template. |
| Preference | Preference specified by the IP route template. |
| Dlci | DLCI specified by the IP route template. |

**Examples**

To display detailed information about the IP route template named "branch_office", use the command:

```
sh ip rou temp=branch_office
```

**Related Commands**

**add ip route template**
**create ipsec policy**
**delete ip route template**
**set ip route template**

# show ip sa

**Syntax**

```
SHow IP SA INTerface=interface
```

where *interface* is an interface name formed by concatenating a Layer 2 interface type, an interface instance, and optionally a hyphen followed by a logical interface number from 0 to 15. If a logical interface is not specified, 0 is assumed.

**Description**

This command displays the list of security associations assigned to an IP interface.

The IP SA commands provide support for RFCs 1825, 1827, and 1829, which have been superseded by IP Security. See Chapter 48, IP Security (IPsec) and RFCs 2401–2412 for more information about IPsec.

The **interface** parameter specifies the name of the interface. The interface must already be assigned to the IP routing module. Valid interfaces are:

- eth (such as eth0, eth0-1)
- PPP (such as ppp0, ppp1-1)
- VLAN (such as vlan1, vlan1-1)
- FR (such as fr0, fr0-1)
- X.25 DTE (such as x25t0, x25t0-1)

To see a list of interfaces currently available, use the **show interface** command on page 9-72 of Chapter 9, Interfaces, or the **show ip interface** command on page 22-202.

**Examples**

To display the list of security associations assigned to interface ppp0, use the command:

```
sh ip sa int=ppp0
```

**Related Commands**

**add ip sa**
**create sa**
**delete ip sa**
**set ip interface**
**show ip interface**

# show ip udp

**Syntax** `SHow IP UDP`

**Description** This command displays the state of current UDP sessions (Figure 22-48, Table 22-45).

Figure 22-48: Example output from the **show ip udp** command

```
  Local port     Local address     Remote port     Process
------------------------------------------------------------
    1698         1.1.3.1              4660          RSVP
    5023         0.0.0.0              5023          SRLP LOG
    5024         0.0.0.0              5024          NETM LOG
    1701         3.3.3.2                 0          L2TP
     520         1.1.2.2                 0          RIP
     514         0.0.0.0               514          SYSLOG
------------------------------------------------------------
```

Table 22-45: Parameters in output of the **show ip udp** command

| Parameter | Meaning | |
|---|---|---|
| Local port | Port number for the UDP connection on this router. See Table 22-7 on page 22-73 for a list of commonly assigned UDP port numbers. | |
| Local address | The IP address of the last interface that was used to transport UDP packets from the router, for a given process. An address of 0.0.0.0 indicates that the UDP session is active, but either no packets have been transmitted yet, or packets have been transmitted without specifying the source IP address. | |
| Remote port | Port number for the UDP connection on the remote host. See Table 22-7 on page 22-73 for a list of commonly assigned UDP port numbers. | |
| Process | The process that is using the UDP session. The following process types may use UDP on the router: | |
| | NTP | Time synchronisation using the Network Time Protocol |
| | LB | Load Balancing |
| | RSVP | Quality of Service determination using the Resource Reservation Protocol |
| | UPNP | Universal Plug and Play |
| | VOIP | Voice over IP |
| | L2TP | Tunnelling of PPP Link Layer data using the Layer 2 Tunnelling Protocol |
| | X25 | The X25 protocol |
| | SYSLOG | Generation/reception of syslog type logs |
| | SRLP LOG | Generation/reception of logs using the Secure Router Log Protocol |
| | NETM LOG | Generation/reception of logs using the Net Manage protocol |
| | TFTP | Download/upload of files using the Trivial File Transfer Protocol |

Table 22-45: Parameters in output of the **show ip udp** command

| Parameter | Meaning | |
|---|---|---|
| Process (cont.) | SNMP | Transfer of device management data using the Simple Network Management Protocol |
| | DHCP SVR | External network node configuration by the router acting as a Dynamic Host Configuration Protocol Server |
| | DHCP CLT | Communications by the router when acting as a client, using the Dynamic Host Configuration Protocol |
| | BOOTP | Communications by the router when acting as a BOOTP Relay Agent |
| | UDP FWD | Forwarding of UDP packets to an external device using IP Helper. |
| | DNS | Hostname resolution using the Domain Name System Protocol |
| | DNS RELAY | The relaying of DNS messages from the router to an external host |
| | RIP | Routing of IP packets using the Routing Information Protocol |
| | IKMP | Secure communications using the Internet Security Association and Key Management Protocol |
| | IKMP NAT | Secure communications using the Internet Security Association and Key Management Protocol via devices configured using Network Address Translation |
| | IPSEC | Secure communications using the IP Security Protocol |
| | TACACS | User authentication using the Terminal Access Controller Access Control System protocol |
| | RADIUS | User authentication using the Remote Authentication Dial In User Service Protocol |
| | RAD ACC | Accounting using the RADIUS protocol |

**Related Commands**

**show ip counter**
**show tcp**

# show ping

**Syntax** `SHow PING`

**Description** This command displays information about the ping configuration and the results of the current or previous **ping** command (Figure 22-49, Table 22-46 on page 22-225).

Figure 22-49: Example output from the **show ping** command

```
Ping Information
----------------------------------------------------------
Defaults:
  Type .......................... IP
  Source ........................ 0.0.0.0
  Destination ................... 192.168.2.1
  Number of packets ............. 10
  Size of packets (bytes) ....... 24
  Timeout (seconds) ............. 1
  Delay (seconds) ............... 1
  Data pattern .................. Not set
  Type of service ............... 0
  Direct output to screen ....... Yes

Current:
  Type .......................... IP
  Source ........................ 0.0.0.0
  Destination ................... 192.168.2.1
  Number of packets ............. 10
  Size of packets (bytes) ....... 24
  Timeout (seconds) ............. 1
  Delay (seconds) ............... 1
  Data pattern .................. 0x00000000
  Type of service ............... 0
  Direct output to screen ....... Yes

Results:
  Ping in progress .............. No
  Packets sent .................. 10
  Packets received .............. 10
  Round trip time minimum (ms) .. 20
  Round trip time average (ms) .. 22
  Round trip time maximum (ms) .. 40
  Last message .................. Finished succesfully
----------------------------------------------------------
```

Table 22-46: Parameters in output of the **show ping** command

| **Parameter** | **Meaning** |
|---|---|
| Type | Network protocol; one of "-", "AppleTalk", "IP", "IPX", "OSI-CLNS", or "IPV6". |
| Source | Source address used in the ping packet, or "Undefined" if an address has not been specified. |
| Destination | Destination address, host name or domain name to ping, or "Undefined" if an address has not been specified. |
| Number of packets | Number of ping packets to send. |
| Size of packets (bytes) | Number of data pattern bytes to include in the packet. |
| Timeout (seconds) | Seconds to wait for a reply before sending the next packet. |
| Delay (seconds) | Seconds to wait before sending the next packet. |
| Data pattern | Data bytes to be used in the data portion of packets transmitted, or "Not Set" if a pattern has not been specified. |
| Type of service | Value of the TOS (Type Of Service) field in the IP header of IP ping packets transmitted. |
| Direct output to screen | Whether the output is sent to the terminal. |
| Ping in progress | Whether a ping is in progress. |
| Packets sent | Number of packets sent. |
| Packets received | Number of packets received. |
| Round trip time minimum (ms) | Quickest round trip time in milliseconds. |
| Round trip time average (ms) | Average round trip time in milliseconds. |
| Round trip time maximum (ms) | Slowest round trip time in milliseconds. |
| Last message | Last message from the **ping** command on page 22-135. |

**Examples** To display the current ping configuration, use the command:

```
sh ping
```

**Related Commands**

**ping**
**set ping**
**stop ping**

# show tcp

**Syntax** SHow TCP[=*tcb*]

where *tcb* is the index of a TCP connection in the TCP connection table

**Description** This command displays the state of current TCP connections. If a TCP connection is specified, details are displayed for it (Figure 22-50, Table 22-47).

If a TCP connection is not specified, the TCP portion of the MIB-II MIB is displayed along with summary information about all current TCP connections (Figure 22-51 on page 22-228, Table 22-48 on page 22-229). Index numbers 3 and 4 indicate locally sourced Telnet sessions. These sessions are from the asynchronous ports attached to the router. Index numbers 5 and 6 indicate *remotely* sourced Telnet sessions.

This command is useful to show if Telnet or other TCP sessions are active, and whether they are running over IPv4 or IPv6. Port 23 is typically reserved for Telnet. Other typical listen ports are reserved for X.25 over TCP and for permanent assignments. When a Telnet session is active, the IP address of the source and destination allows the particular session to be identified.

See Table 22-7 on page 22-73 for a list of commonly assigned TCP port numbers.

Figure 22-50: Example output from the **show tcp** command for a specific TCP connection

```
TCB: 05  Local: 192.168.35.45,00023  Remote: 192.168.35.61,01032
State: ESTAB  O/P State: IDLE
SND.UNA: 0047376265  SND.NXT: 0047376265  SND.WND: 04096
Last Seq: 0641204304  Last Ack: 0047376265
SendCon: 06022  DataCount: 0000000000
RCV.NXT: 0641204305  RCV.WND: 00000
Round Trip Time
SendSrt: 00218  Deviation: 00013  SendReXmit: 00033
Timers:
Event       Time (cs)
No events in timer queue
Fragment list:
Sequence     Length    End sequence
No fragments in fragment list
```

Table 22-47: Parameters in output of the **show tcp** command for a specific TCP connection

| **Parameter** | **Meaning** |
|---|---|
| TCB | Index into the TCP connection table for this connection. |
| Local | Local IP address and port for the connection. See Table 22-7 on page 22-73 for a list of commonly assigned TCP port numbers. |
| Remote | Remote IP address and port for the connection. See Table 22-7 on page 22-73 for a list of commonly assigned TCP port numbers. |
| State | State of the connection (Table 22-49 on page 22-230). |
| O/P State | Output queue state:<br>IDLE<br>PERST — Remote host has closed its receive window and router is transmitting data one character at a time to aid the process of re-opening the window<br>TRANS — There is data to transmit<br>RETRN — The router is retransmitting data |
| SND.UNA | Sequence number of the last unacknowledged octet transmitted over the connection. |
| SND.NXT | Sequence number of the next octet to be transmitted over the connection. |
| SND.WND | Transmit window for the connection. |
| Last Seq | Packet received from the connection. |
| Last Ack | Last acknowledgement received from the connection. |
| SendCon | Internal congestion parameter. |
| DataCount | Number of data octets transmitted over this connection. |
| RCV.NXT | Next octet expected from the connection. |
| RCV.WND | Receive window for the connection. |
| SendSrt, Deviation, SendReXmit | Round trip time parameters used to implement Van Jacobson’s retransmit time algorithm. |
| Event | An event on the timer queue:<br>None — No data<br>Send — Transmit data<br>Persist — Transmit data one character at a time if in persist state<br>Transmit — Retransmit data<br>Delete — Clear TCP connection |
| Time (cs) | Time to this event in centiseconds. |
| Sequence | First sequence number of a fragment waiting for defragmentation. |
| Length | Length of the fragment. |
| End sequence | Last sequence number of the fragment. |

Figure 22-51: Example output from the **show tcp** command

```
 TCP MIB parameters, counters and connections
------------------------------------------------------------
 RTO Algorithm:           vanj
 RTO Min (ms):      0000000500   RTO Max (ms):    0000020000

 Maximum connections:    00040
 Maximum connections
         since restart:  00003

 Active Opens:           00004   Passive Opens:        00005
 Attempt Fails:          00000   Established Resets:   00000
 Current Established:    00004

 In Segs:           0000000070   In Segs Error:   0000000000
 Out Segs:          0000000104   Out Segs Retran: 0000000000
 Out Segs With RST: 0000000000



 TCP Debug counters (only shown if tcp debuging is enabled)
 ======================================================

 Tcp Timer magic ivalid: 00000   TCP timer id ivalid:  00000
 Output Evnt Magic Ivld: 00000   Output Event id ivld: 00000
 Connection Table:
 Index   Proto  State
         Local port and address
         Remote port and address
 ----------------------------------------------------------
     0   IPv4   listen
         00023  0.0.0.0
         00000  0.0.0.0
----------------------------------------------------------
     1   IPv6   listen
         00023  ::
         00000  ::
------------------------------------------------------------
     2   IPv4   listen
         00080  0.0.0.0
         00000  0.0.0.0
----------------------------------------------------------
     3   IPv4   established
         00127  172.16.253.2
         00023  172.16.8.5
----------------------------------------------------------
     4   IPv4   established
         00133  172.16.253.2
         00023  172.16.8.5
----------------------------------------------------------
     5   IPv4   established
         00023  172.16.40.254
         00002  172.16.248.51
----------------------------------------------------------
     6   IPv4   established
         00023  172.16.40.254
         02123  172.16.9.190
----------------------------------------------------------
     7   IPv6   established
         00023  3001:0001::0022
         01046  3001:0001::0001
```

Table 22-48: Parameters in output of the **show tcp** command

| Parameter | Meaning |
|---|---|
| RTO Algorithm | Retransmit time algorithm. |
| RTO Min (ms), RTO Max (ms) | Retransmit time algorithm parameters (milliseconds) |
| Maximum connections | Maximum number of TCP connections allowed. |
| Maximum connections since restart | Maximum number of TCP connections that the router has had at any time since its last restart. |
| Active Opens | Number of active TCP opens. Active opens initiate connections. |
| Passive Opens | Number of TCP passive opens. Passive opens are issued to wait for a connection from another host. |
| Attempt Fails | Number of failed connection attempts. |
| Established Resets | Number of connections established but have been reset. |
| Current Established | Number of current connections. |
| In Segs | Number of segments received. |
| In Segs Error | Number of segments received with an error. |
| Out Segs | Number of segments transmitted. |
| Out Segs Retran | Number of segments retransmitted. |
| Out Segs With RST | Number of segments transmitted with the RST bit set. |
| **TCP Debug counters** | Internal debug counters for TCP. These counters display only when a TCP debugging mode is enabled. |
| Tcp Timer magic ivalid | Internal counter for TCP. Contact your authorised distributor or reseller if this number increments. |
| TCP timer id ivalid | Internal counter for TCP. Contact your authorised distributor or reseller if this number increments. |
| Output Event Magic Ivld | Internal counter for TCP. Contact your authorised distributor or reseller if this number increments. |
| Output Event id ivld | Internal counter for TCP. Contact your authorised distributor or reseller if this number increments. |
| **Connection Table** | |
| Index | Entry number in the table. |
| Proto | Protocol type of the session; IPv4 or IPv6. |
| State | State of the session (Table 22-49 on page 22-230). These are the names of the various states in the TCP state diagram. For more detailed information, refer to the RFC or a text on TCP/IP. |
| Local port and address | The router's TCP port number and IP address. See Table 22-7 on page 22-73 for a list of commonly assigned UDP port numbers. |
| Remote Port and address | TCP port number and IP address of the remote host. See Table 22-7 on page 22-73 for a list of commonly assigned UDP port numbers. |

Table 22-49: TCP states

| State | Meaning |
|---|---|
| Closed | The starting state and should not be present at any time since the server module should immediately go into the listen state. |
| Listen | This is a passive open and is entered when the server module is waiting for external connections to be made. |
| Synsent | The server enters this state when a connection is being initiated from a local session and also when a remotely initiated session is being set up just prior to entering the established state. |
| Synreceived | This state is entered when a SYN packet is received indicating that a remote system is attempting to establish a session. |
| Established | This state indicates that a connection has been made and is currently active. Data packets can now flow in both directions. |
| Finwait1 | This state indicates the first step of a locally initiated termination of a session. The closewait state indicates a remote station is initiating the termination. |
| Finwait2 | This state is also part of the local termination process and is required to ensure that no data in transit in lost. |
| Closewait | This state is entered when the remote entity has sent a FIN packet to terminate this link. The server entity sends an ACK packet. |
| Lastack | The ACK packet from above causes the remote system to send a close packet and the server enters this state and sends a FIN packet thereby terminating this link. |
| Closing | This state is entered when the established local session has initiated a termination (gone to FINWAIT1) and received a FIN packet from the remote entity indicating that it can now terminate also. This is an alternate path to FINWAIT2. |
| Timewait | This state may be entered as part of the termination process while waiting for a remote entity to respond to the final ACK packet. The session is then closed. |

**Related Commands**

**disable tcp debug**
**enable tcp debug**
**show ip counter**
**show ip udp**

# show trace

**Syntax** `SHow TRAce`

**Description** This command displays information about the current trace route configuration and the result of the current or previous trace route operation (Figure 22-52, Table 22-50 on page 22-232).

Figure 22-52: Example output from the **show trace** command

```
Trace information
------------------------------------------------------------
Defaults:
  Destination ................... 121.23.5.4
  Source ........................ 202.36.163.31
  Number of packets per hop ..... 3
  Timeout (seconds) ............. 1
  Type of service ............... 8
  Port .......................... 33434
  Minimum time to live .......... 1
  Maximum time to live .......... 20
  Addresses only output ......... Yes
  Direct output to screen ....... Yes

Current:
  Destination ................... 206.123.21.3
  Source ........................ 202.36.163.31
  Number of packets per hop ..... 3
  Timeout (seconds) ............. 1
  Type of service ............... 8
  Port .......................... 33434
  Minimum time to live .......... 1
  Maximum time to live .......... 12
  Addresses only output ......... Yes
  Direct output to screen ....... Yes

Results:
  Trace route in progress ....... No

 1. 202.36.163.21       20     20     20 (ms)
 2. 202.49.72.62         0      0      0 (ms)
 3. 203.97.191.65        0      0      0 (ms)
 4. 203.97.191.22       80     93    100 (ms)
 5. 140.200.128.2       40     46     60 (ms)
 6. 131.119.17.205     460    473    480 (ms)
 7. 131.119.0.129      540    553    560 (ms)
 8. 4.0.1.90           800    800    800 (ms)
 9. 4.0.1.14           440    440    440 (ms)
10. 198.32.136.39      480    480    480 (ms)
11. 140.223.9.21       520    520    520 (ms)
12. 140.223.9.18       560    560    560 (ms)

  Last message .................. Target unreached
------------------------------------------------------------
```

Table 22-50: Parameters in output of the **show trace** command

| Parameter | Meaning |
|---|---|
| Destination | Destination IP address, host name or domain name. If the destination address has not been specified, “Not Set” is displayed. |
| Source | Source IP address to use in the packets transmitted. If the source address has not been specified, “Not Set” is displayed. |
| Number of packets per hop | Number of packets to transmit to each hop on the route. |
| Timeout | Seconds to wait for a reply before sending the next packet. |
| Type of service | Value of the TOS field in the IP header of packets transmitted. |
| Port | Destination UDP port number. |
| Minimum time to live | Minimum TTL (Time To Live) used to skip some hops at the start of the route. |
| Maximum time to live | Maximum hops to which packets are transmitted. |
| Addresses only output | Whether address-to-name translation is performed for the output. |
| Direct output to screen | Whether output is sent to the terminal. |
| Trace route in progress | Whether a trace route is in progress. |
| 1-*n* | Hop number, IP address, and the maximum, minimum and average round trip time in milliseconds to each hop on the route. |
| Last message | Last message from the **ping** command on page 22-135. |

**Examples** To show the current trace route configuration, use the command:

```
sh tra
```

**Related Commands**

**set trace**
**stop trace**
**trace**

# stop ping

**Syntax**

```
STop PING
```

**Description** This command stops a ping in progress.

**Examples** To stop a ping in progress, use the command:

```
st ping
```

**Related Commands**

ping
set ping
show ping

# stop trace

**Syntax**

```
STop TRAce
```

**Description** This command stops a trace route in progress.

**Examples** To stop a trace route that is in progress, use the command:

```
st tra
```

**Related Commands**

show trace
stop trace
trace

# trace

**Syntax**

```
TRAce [[IPaddress=]ipadd] [ADDROnly={No|OFf|ON|Yes}]
    [MAXTtl=number] [MINTtl=number] [NUMber=number]
    [POrt=1..65535] [SCReenoutput={No|OFf|ON|Yes}]
    [SOurce=ipadd] [TIMEOut=number] [TOS=0..255]
```

where:

- *ipadd* is an IPv4 address in dotted decimal notation, a valid IPv6 address, or a host name from the host name table.
- *domain-name* is a valid network domain name.
- *number* is a decimal number.

**Description**

This command performs a trace route. The parameters in this command override defaults set with the **set trace** command on page 22-174.

This command can be used to view the path to a node running IPv6.

The **ipaddress** parameter specifies the destination IP address; this command traces the route to this IP address. If you do not specify an IP address here or in the **set trace** command on page 22-174 then a trace is not performed and an error message is displayed. To configure a DNS for the router to use to resolve domain names, use the **add ip dns** command on page 22-68.

The **addronly** parameter specifies whether trace output is presented as IP addresses only, as opposed to IP addresses and their DNS name equivalent. If **on**, output is presented as IP addresses. If **addronly** is not specified, the default is used.

The **maxttl** parameter specifies the maximum value for the TTL (Time To Live) field in the IP packet, and is used to limit the trace route to a maximum number of hops. If **maxttl** is not specified, the default is used.

The **minttl** parameter specifies the initial value of the TTL (Time To Live) field in the IP packet, and can be used to skip hops at the start of the route. If **minttl** is not specified, the default is used.

The **number** parameter specifies the number of packets to send to each hop. A maximum of 100 packets may be transmitted. If **number** is not specified, the default is used.

The **port** parameter specifies the UDP destination port number that **trace** uses to transmit the packet to the first hop destination. The destination UDP port number is incremented by one for each subsequent hop so that **trace** can match reply packets to transmitted packets. You should set **port** to a value such that all devices along the trace route will be listening on the appropriate UDP port. If **port** is not specified, the default is used.

The **screenoutput** parameter specifies whether the output is sent to the terminal. If **screenoutput** is not specified, the default is used.

The **source** parameter specifies the IP address to use as a source address in the packets. If this parameter is not set, the default IPv4 address of the interface is set as the source address. Because this IPv4 address causes a conflict when an IPv6 address is specified in the **ipaddress** parameter, this parameter is required when tracing a route to an IPv6 address.

The **timeout** parameter specifies how many seconds to wait for a response before sending packets to the next hop. If ICMP “unreachable” messages are received within the timeout period, packets are transmitted to the next hop immediately. If **timeout** is not specified, the default is used.

The **tos** parameter specifies the value of the TOS (Type Of Service) field in the IP header of the packets being transmitted. If **tos** is not specified, the default is used.

**Related Commands**

**set trace**
**show trace**
**stop trace**