Chapter 13 Routing IP

13

This chapter begins with an introduction to Cisco's implementation of the IP protocol for its line of routing products, and continues with an in-depth view of configuration options, IP addressing and its various protocols, and examples of well-designed networks. These tasks and topics are covered in this chapter:

- Configuring IP
- Assigning IP addresses, address resolution, and broadcast addresses
- Configuring access and security
- Configuring accounting

See the chapter "The IP Routing Protocols" for information on the various routing protocols, how they have evolved, and how they are best used in complex internetworks.

Cisco's Implementation of IP

Cisco's implementation of TCP/IP provides all major services contained in the various protocol specifications. Cisco routers also provide the TCP and UDP *little services* called Echo service and Discard service. These services are described in RFC 862 and RFC 863.

Cisco supports both TCP and UDP at the Transport Layer, for the maximum flexibility in services. Some Cisco global and interface commands require UDP packets to be sent (see the section "Configuring ICMP and Other IP Services"). Cisco supports all standards for IP broadcasts.

Configuring IP

The process of configuring your router for IP routing differs from the procedures for configuring other protocols in that you don't have to initially enable IP routing. All Cisco routers are shipped with IP already enabled. The **ip routing** global configuration command is described later in this chapter to allow you to re-enable IP routing if you have disabled it. You should follow the steps on the next page to configure individual interfaces and other options.

Step 1: Enter an address for the interface on which you will be routing IP using the **ip address** interface subcommand.

Step 2: Consider addressing options and broadcast packet handling, using commands described in the "Setting IP Interface Addresses" and "Broadcasting In the Internet" sections.

Step 3: Optionally, configure packet sizes and other performance parameters as well as ICMP and other IP services. Information for these tasks are in the section "Configuring ICMP and Other IP Services."

Step 4: Configure access lists and other security options, if desired.

Step 5: Configure routing. The IP routing protocols are discussed in the chapter "The IP Routing Protocols."

Each task is described in the following sections, and are followed by descriptions of the EXEC commands to maintain, monitor and debug an IP network. Summaries of the global configuration commands and interface subcommands described in this section appear at the end of this chapter.

Enabling IP Routing

The **ip routing** global configuration command enables IP routing for the router. Its full syntax follows.

ip routing
no ip routing

If the system is running bridging software, the **no ip routing** subcommand turns off IP routing when setting up a system to bridge (as opposed to route) IP datagrams. (See the explanations on bridging options in the chapters in Part Five.) The default setting is to perform IP routing.

Assigning IP Addresses

The official description of Internet addresses is found in RFC 1020, "Internet Numbers." The Network Information Center (NIC), which maintains and distributes the RFC documents, also assigns Internet addresses and network numbers. Upon application from an organization, NIC assigns a network number or range of addresses appropriate to the number of hosts on the network.

Address Classes and Formats

As described in RFC 1020, Internet addresses are 32-bit quantities, divided into five classes. The classes differ in the number of bits allocated to the *network* and *host* portions of the address. For this discussion, consider a network to be a collection of devices (hosts) that have the same network field value in their Internet addresses.

Note: When discussing IP, all network-attached devices are referred to as *hosts*.

The Class A Internet address format allocates the highest eight bits to the network field and sets the highest-order bit to 0 (zero). The remaining 24 bits form the host field. Only 128 Class A networks can exist, but each Class A network can have almost 17 million hosts. Figure 1-1 illustrates the Class A address format.

Figure 1-1 Class A Internet Address Format

The Class B Internet address format allocates the highest 16 bits to the network field and sets the two highest-order bits to 1,0. The remaining 16 bits form the host field. Over 16,000 Class B networks can exist, and each Class B network can have over 65,000 hosts. Figure 1-2 illustrates the Class B address format.

Figure 1-2 Class B Internet Address Format

The Class C Internet address format allocates the highest 24 bits to the network field and sets the three highest-order bits to 1,1,0. The remaining eight bits form the host field. Over two million Class C networks can exist, and each Class C network can have up to 254 hosts.Fig-ure 1-3 illustrates the Class C address format.

Figure 1-3 Class C Internet Address Format

The Class D Internet address format is reserved for multicast groups, as discussed in RFC 988. In Class D addresses, the four highest-order bits are set to 1,1,1,0.

The Class E Internet address format is reserved for future use. In Class E addresses, the four highest-order bits are set to 1,1,1,1. The router currently ignores Class D and Class E Internet addresses, except the global broadcast address *255.255.255.255*.

Internet Address Notation

The notation for Internet addresses consists of four numbers separated by dots (periods). Each number, written in decimal, represents an 8-bit octet. When strung together, the four octets form the 32-bit Internet address. This notation is called dotted decimal.

These samples show 32-bit values expressed as Internet addresses:

```
192.31.7.19
10.7.0.11
255.255.255.255
0.0.0.0
```

Note that 255, which represents an octet of all ones, is the largest possible value of a field in a dotted-decimal number.

Allowable Internet Addresses

Some Internet addresses are reserved for special uses and cannot be used for host, subnet, or network addresses. Table 1-1 lists ranges of Internet addresses and shows which addresses are reserved and which are available for use.

Table 1-1 Reserved and Available Internet Addresses

Class	Address or Range	Status
A	0.0.0.0	Reserved
	1.0.0.0 through 126.0.0.0	Available
	127.0.0.0	Reserved
B	128.0.0.0	Reserved
	128.1.0.0 through 191.254.0.0	Available
	191.255.0.0	Reserved
C	192.0.0.0	Reserved
	192.0.1.0 through 223.255.254	Available
	223.255.255.0	Reserved

D, E	224.0.0.0 through 255.255.255.254	Reserved
	255.255.255.255	Broadcast

Internet Address Conventions

To create an address that refers to a specific network, the bits in the host portion of the address must all be zero. For example, the Class C address *192.31.7.0* refers to a particular network (no local component).

Conversely, if you want a local address only, without a network portion, all the bits in the network portion of an address must be 0. For example, the Class C address *0.0.0.234* refers to a particular host (local address).

If you want to send a packet to all hosts on the network specified in the network portion of the address, the local address must be all ones. For example, the Class B address *128.1.255.255* refers to all hosts on network *128.1.0.0*. This is called a *broadcast*, which is described in the section "Broadcasting in the Internet." You can also find general information on broadcasts in the chapter "The IP Routing Protocols."

Note: Because of these conventions, do not use an Internet address with all zeros or all ones in the host portion for your router address.

You can manually configure the router's routing table, or you can specify that a high-level routing protocol dynamically build the routing table. In both cases, the routing table is based on the network portion of addresses. Consequently, the addresses of hosts on a single physical network must have the same network number to permit automatic routing. If a network does not meet this requirement, the routers will be unable to communicate with all of the hosts on that network. (The one exception to this general rule is the use of secondary addresses, which is described in the section "Setting IP Interface Addresses.")

Subnetting and Routing

Subnetting is a scheme for imposing a simple two-level hierarchy on host addresses, allowing multiple, logical networks to exist within a single Class A, B, or C network. The usual practice is to use a few of the leftmost bits in the host portion of the network addresses for a subnet field. For example, Figure 1-4 shows a Class B address with five bits of the host portion used as the subnet field. The official description of subnetting is contained in RFC 950, "Internet Standard Subnetting Procedure."

Figure 1-4 A Class B Address with a 5-Bit Subnet Field

As with the host portion of an address, do not use all zeros or all ones in the subnet field.

Routers and hosts can use the subnet field for routing. The rules for routing on subnets are identical to the rules for routing on networks. However, correct routing requires that all subnets of a network be physically contiguous. In other words, the network must be set up such that it does not require traffic between any two subnets to cross another network. The current Cisco implementation requires that all subnets of a network have the same number of subnet bits.

Creating a Single Network from Separated Subnets

You can create a single network from subnets that are physically separated by another network by using a *secondary address*. An example is shown in the section "Setting IP Interface Addresses."

Note: A subnet cannot appear on more than one active interface of the router at a time.

Subnet Masks

A *subnet mask* identifies the subnet field of network addresses. All subnets of a given class, A, B, or C, should use the same subnet mask. This mask is a 32-bit Internet address written in dotted-decimal notation with all ones in the network and subnet portions of the address. For the example shown in Figure 13-4, the subnet mask is *255.255.248.0*. Table 1-2 shows the subnet masks you can use to divide an octet into subnet and host fields. The subnet field can consist of any number of the host field bits; you do not need to use multiples of eight. However, you should use three or more bits for the subnet field—a subnet field of two bits yields only four subnets, two of which are reserved (the 1,1 and 0,0 values).

Table 1-2 Subnet Masks

Subnet Bits	Host Bits	Hex Mask	Decimal Mask
0	8	0	0
1	7	0x80	128
2	6	0xC0	192
3	5	0xE0	224
4	4	0xF0	240
5	3	0xF8	248
6	2	0xFC	252
7	1	0xFE	254
8	0	0xFF	255

Setting IP Interface Addresses

Use the **ip address** interface subcommand to set an IP address for an interface. The full command syntax follows:

ip address *address net-mask* [**secondary**]
no ip address *address net-mask* [**secondary**]

The two required arguments are an IP address and the network mask for the associated IP network. The subnet mask must be the same for all interfaces connected to subnets of the same network. Hosts can determine subnet masks using the Internet Control Message Protocol (ICMP) *Mask Request* message. Routers respond to this request with an ICMP *Mask Reply* message. (See the section "Configuring ICMP and Other IP Services" for more details.)

You can disable IP processing on a particular interface by removing its IP address with the **no ip address** subcommand. If the router detects another host using one of its IP addresses, it will print an error message on the console. The software supports multiple IP addresses per interface.

You may use this command to specify additional secondary IP addresses by including the keyword **secondary** after the IP address and subnet mask.

Example:

In the sample below, *131.108.1.27* is the primary address and *192.31.7.17* is a secondary address for Ethernet 0.

```
interface ethernet 0
ip address 131.108.1.27 255.255.255.0
ip address 192.31.7.17 255.255.255.0 secondary
```

Using Subnet Zero

Subnetting with a subnet address of zero is generally not allowed, because of the confusion inherent in having a network and a subnet with indistinguishable addresses. For example, if network *131.108.0.0* is subnetted as *255.255.255.0*, subnet zero would be written as *131.108.0.0*—which is identical to the network address.

To enable or disable the use of subnet zero for interface addresses and routing updates, use the global configuration command **ip subnet-zero**. Its full command syntax follows:

ip subnet-zero
no ip subnet-zero

Example:

In the example below, we enable subnet-zero for the router:

```
ip subnet-zero
```

Local and Network Addresses: Address Resolution

A device in the Internet may have both a local address, which uniquely identifies the device on its local segment or LAN, and a network address which identifies the network the device belongs to. The local address is more properly known as a data link address because it is contained in the data link layer (Layer 2 of the OSI Model) part of the packet header and is read by data link devices (bridges and all device transceivers, for example). The more technically inclined will refer to local addresses as MAC addresses because the Media Access Control (MAC) sublayer within the data link layer processes addresses for the layer.

To communicate with a device on Ethernet, the router must first determine the 48-bit MAC or local data link address of that device. The process of determining the local data link address from an Internet address is called *address resolution*. The process of determining the Internet address from a local data link address is called *reverse address resolution*. The router uses three forms of address resolution: Address Resolution Protocol (ARP), proxy ARP, and Probe (which is similar to ARP). The router also uses the Reverse Address Resolution Protocol (RARP). The ARP, proxy ARP, and RARP protocols, which are used on Ethernets, are defined in RFCs 826, 1027, and 903, respectively. Probe is a protocol developed by the Hewlett-Packard Company for use on IEEE-802.3 networks.

Address Resolution Using ARP

To send an Internet data packet to a local host with which it has not previously communicated, the router first broadcasts an ARP Request packet. The ARP Request packet requests the MAC local data link address corresponding to an Internet address. All hosts on the network receive this request, but only the host with the specified Internet address will respond.

If present and functioning, the host with the specified Internet address responds with an ARP Reply packet containing its local data link address. The router receives the ARP Reply packet, stores the local data link address in the ARP cache for future use, and begins exchanging packets with the host.

The EXEC command **show arp** may be used to examine the contents of the ARP cache. The **show ip arp** command will show IP entries.

Tailoring ARP: Static Entries and Timing

The function of ARP is to provide a dynamic mapping between 32-bit IP addresses and 48-bit local hardware (Ethernet, FDDI, Token Ring) addresses. ARP may also be used for protocols other than IP and media that have other than 48-bit addresses.

Because most hosts support *dynamic resolution*, you generally do not need to specify static ARP cache entries. If you do need to define static arp cache entries, you can do so globally.

When used as a global configuration command, the **arp** command installs a permanent entry in the ARP cache. The router uses this entry to translate 32-bit Internet Protocol addresses into 48-bit hardware addresses. The full syntax follows:

arp *internet-address hardware-address type* [**alias**]
no arp *internet-address*

The argument *internet-address* is the Internet address in dotted decimal format corresponding to the local data link address specified by the argument *hardware-address*.

The argument *type* is an encapsulation description. This is typically the **arpa** keyword for Ethernets and is always **snap** for FDDI and Token Ring interfaces, and **ultra** for the Ultranet interfaces. See the discussions of the individual interface types for more information on possible encapsulations.

The optional keyword **alias** indicates that the router should respond to ARP requests as if it were the owner of the specified IP address.

Example:

The following is a sample of a static ARP entry for a typical Ethernet host.

```
arp 192.31.7.19 0800.0900.1834 arpa
```

The **no arp** subcommand removes the specified entry from the ARP cache. To remove all non-static entries from the ARP cache, use the privileged EXEC command **clear arp-cache.**

When used as an interface subcommand, the **arp** command controls the interface-specific handling of IP address resolution into 48-bit Ethernet hardware addresses. The full syntax of the **arp** interface subcommand follows:

arp {**arpa**|**probe**|**snap**}
no arp {**arpa**|**probe**|**snap**}

The keyword **arpa**, which is the default, specifies standard Ethernet style ARP (RFC 826), **probe** specifies the HP-proprietary Probe protocol for IEEE-802.3 networks, and **snap** specifies ARP packets conforming to RFC 1042. The **show interfaces** monitoring command displays the type of ARP being used on a particular interface. Probe is described more in a later section in this chapter.

Note: Unlike most commands that take multiple arguments, arguments to the **arp** command are not mutually exclusive. Each command enables or disables a specific type of ARP. For example, if you enter the **arp arpa** command followed by the **ip probe** command, the router would send two packets each time it needed to discover a MAC address.

To set the number of seconds an ARP cache entry will stay in the cache, use the **arp timeout** interface subcommand. The full syntax of this command follows:

arp timeout *seconds*
no arp timeout

The value of the argument *seconds* is used to age an ARP cache entry related to that interface. By default, the *seconds* argument is set to four hours (14,400 seconds). A value of zero seconds sets no timeout.

Use the **no arp timeout** command to return to the default value.

This command is ignored when issued on interfaces that do not use ARP. Use the EXEC command **show interfaces** to display the ARP timeout value. The value follows the `Entry Timeout:` heading, as seen in this sample display:

```
ARP type: ARPA, HP-PROBE, Entry Timeout: 14400 sec
```

Example:

The following example illustrates how to set the ARP timeout to 12000, to allow entries to time out more quickly than the default.

```
arp timeout 12000
```

Address Resolution Using Proxy ARP

The router uses proxy ARP, as defined in RFC 1027, to help hosts with no knowledge of routing determine the hardware addresses of hosts on other networks or subnets. Under proxy ARP, if the router receives an ARP Request for a host that is not on the same network as the ARP Request sender, and if the router has the best route to that host, then the router sends an ARP Reply packet giving its own local data link address. The host that sent the ARP Request then sends its packets to the router, which forwards them to the intended host.

The **no ip proxy-arp** interface subcommand disables proxy ARP on the interface. The full command syntax for this command follows.

ip proxy-arp
no ip proxy-arp

The default is to perform proxy ARP; the **no ip proxy-arp** command disables this default.

Address Resolution Using Probe

By default, the router uses the HP-proprietary Probe protocol (in addition to ARP) whenever it attempts to resolve an IEEE-802.3 or Ethernet local data link address. The subset of Probe that performs address resolution is called *Virtual Address Request and Reply.* Using Probe, the router can communicate transparently with Hewlett-Packard IEEE-802.3 hosts that use this type of data encapsulation.

The **arp probe** commands enable or disable Probe protocol for IEEE-802.3 networks, as follows.

arp probe
no arp probe

The other options of the **arp** command are discussed under ARP, earlier in this chapter.

Reverse Address Resolution Using RARP and BootP

Reverse ARP (RARP) was defined in RFC 903. If a router does not know the IP address of one of its Ethernet interfaces, it will try RARP during start up processing to attempt to determine the Internet address, based on its interface local data link address. Diskless hosts also use RARP at boot time to determine their protocol addresses. RARP works in the same way as ARP, except that the RARP Request packet requests an Internet address instead of a local data link address. Use of RARP requires a RARP server on the same network segment as the router interface.

A router without non-volatile memory uses both Reverse ARP (RARP) and Boot Protocol (BootP) messages when trying to obtain its interface address from network servers.

BootP, defined in RFC 951, specifies a method for determining the Internet address of a host from its Ethernet local data link address. The basic mechanism is similar to that used by Reverse ARP, but it is UDP-based rather than a distinct Ethernet protocol. The main advantage of BootP is that its messages can be routed through routers, whereas RARP messages cannot leave the local Ethernet-based network.

Broadcasting in the Internet

A broadcast is a data packet destined for all hosts on a particular physical network. Network hosts recognize broadcasts by special addresses. This section describes the meaning and use of Internet broadcast addresses. For detailed discussions of broadcast issues in general, see RFC 919, “Broadcasting Internet Datagrams,” and RFC 922, “Broadcasting Internet Datagrams in the Presence of Subnets.” The router support for Internet broadcasts generally complies with RFC 919 and RFC 922; however, the router does not support multisubnet broadcasts as defined in RFC 922.

The current standard for an Internet broadcast address requires that the host portion of the address consist of all ones. If the network portion of the broadcast address is also all ones, the broadcast applies to the local network only. If the network portion of the broadcast address is not all ones, the broadcast applies to the network or subnet specified.

Cisco routers support two kinds of broadcasting: *directed broadcasting* and *flooding*. A directed broadcast is a packet sent to a specific network or series of networks, while a flooded broadcast packet is sent to every network, as shown in . The packet that is incoming from interface E0 is flooded to interfaces E1, E2 and Serial 0. A directed-broadcast address includes the network or subnet fields.

Figure 1-5 IP Flooded Broadcast

For example, if the network address is *128.1.0.0*, then the address *128.1.255.255* indicates all hosts on network *128.1.0.0*. This would be a directed broadcast. If network *128.1.0.0* has a subnet mask of *255.255.255.0* (the third octet is the subnet field), then the address *128.1.5.255* specifies all hosts on subnet 5 of network *128.1.0.0*, another directed broadcast.

The **no ip directed-broadcast** interface subcommand disables forwarding of directed broadcasts on the interface. The full syntax of this command follows.

ip directed-broadcast
no ip directed-broadcast

The default is to forward directed broadcasts. You disable the default condition with the **no ip directed broadcast**, and re-enable forwarding of directed broadcasts with the **ip directed broadcast** command.

Internet Broadcast Addresses

The router supports Internet broadcasts on both local and wide area networks. There are at least four popular standard ways of indicating an Internet broadcast address. You can configure a router host to generate any form of Internet broadcast address. The router can also receive and understand any form of Internet broadcast address. By default, a router uses all ones for both the network and host portions of the Internet broadcast address (255.255.255.255). You can change the Internet broadcast address by using the **ip broadcast-address** interface subcommand. Following is the full command syntax:

ip broadcast-address [*address*]
no ip broadcast-address [*address*]

The argument *address* is the desired IP broadcast address for a network. If a broadcast address is not specified, the system defaults to a broadcast address of all ones or 255.255.255.255.

Use the **no ip broadcast-address** command to remove the broadcast address.

To change from another broadcast address to the default broadcast address of 255.255.255.255, you must enter the following command:

ip broadcast-address 255.255.255.255

You *cannot* use a command in the format **no ip broadcast-address** *x.x.x.x* (where *x.x.x.x* is *not* 255.255.255.255) to return to the default broadcast address.

If the router does not have nonvolatile memory, and you want the specify the broadcast address to use before it has its configuration, you can change the Internet broadcast address by setting jumpers in the processor configuration register. Setting bit 10 causes the router to use all zeros. Bit 10 interacts with bit 14, which controls the network and subnet portions of the broadcast address. Setting bit 14 causes the router to include the network and subnet portions of its address in the broadcast address. Table 1-3 shows the combined effect of setting bits 10 and 14.

Table 1-3 Configuration Register Settings for Broadcast Address Destination

Bit 14	**Bit 10**	**Address (<net><host>)**
out	out	<ones><ones>
out	in	<zeros><zeros>
in	in	<net><zeros>
in	out	<net><ones>

For more information about the configuration register, see the Cisco Systems hardware reference guide for your system.

Forwarding of Broadcast Packets and Protocols

There are circumstances in which you want to control which broadcast packets and which protocols are forwarded. You do this with helper addresses and the **forward-protocol** commands.

The **ip helper-address** interface subcommand tells the router to forward UDP broadcasts, including BootP, received on this interface. (UDP is the connectionless alternative to TCP at the Transport Layer.) Use the **ip helper-address** interface subcommand to specify the destination address for forwarding broadcast packets. Full command syntax follows.

ip helper-address *address*
no ip helper-address *address*

The *address* argument specifies a destination broadcast or host address to be used when forwarding such datagrams. You can have more than one helper address per interface. You remove the list with **no ip helper-address**.

If you do not specify a **helper address** command, the router will not forward UDP broadcasts.

Example:

This example defines an address that act as a helper address.

```
ip helper-address 121.24.43.2
```

The **ip forward-protocol** global configuration command allows you to specify which protocols and ports the router will forward. Its full syntax is listed next.

ip forward-protocol {**udp**|**nd**|**spanning-tree**} [*port*]
no ip forward-protocol {**udp**|**nd**|**spanning-tree**} [*port*]

The keyword **nd** is the ND protocol used by older diskless SUN workstations. The keyword **udp** is the UDP protocol. A UDP destination port can be specified to control which UDP services are forwarded. By default both UDP and ND forwarding are enabled if a helper address has been defined for an interface. If no ports are specified, these datagrams are forwarded, by default:

- Trivial File Transfer (TFTP)
- Domain Name System
- IEN-116 Name Server
- Time service
- NetBios Name Server
- NetBios Datagram Server
- Boot Protocol (BootP) client and server datagrams
- TACACS service

Use the **no ip forward-protocol** command with the appropriate keyword and argument to remove the protocol.

Example:

The example below first defines a helper address, then uses the **ip forward-protocol** command to specify forwarding of UDP only.

```
interface ethernet 1
ip helper-address 131.120.1.0
ip forward-protocol udp
```

Flooding IP Broadcasts

To permit IP broadcasts to be flooded throughout the internetwork in a controlled fashion, use the global configuration command **ip forward-protocol spanning-tree** (full command syntax follows):

ip forward-protocol spanning-tree

no ip forward-protocol spanning-tree

This command is an extension of the **ip helper-address** interface command, in that the same packets that may be subject to the helper address and forwarded to a single network may now be flooded. Only one copy of the packet will be put on each network segment in the network.

The **ip forward-protocol spanning-tree** command uses the database created by the bridging spanning tree protocol. The transparent bridging option must be in the routing software, and bridging must be configured on each interface which is to participate in the flooding. If an interface does not have bridging configured, it will still be able to receive broadcasts, but it will never forward broadcasts received on that interface, and it will never use that interface to send broadcasts received on a different interface. If no bridging is desired, then a type-code bridging filter may be configured which will deny all packet types from being bridged. The spanning-tree database is still available to the IP forwarding code to use for the flooding.

Packets must meet the following criteria to be considered for flooding (these are the same conditions for IP helper addresses):

- Packets must be MAC-level broadcasts.
- Packets must be IP-level broadcasts.
- Packets must be a TFTP, DNS, IEN-116, Time, NetBios, ND, or BootP packet, or a UDP protocol specified by the command **ip forward-protocol udp.**
- The packets' time to live (TTL) value must be at least two.

A flooded UDP datagram is given the destination address specified by the **ip broadcast** command on the output interface. This can be set to any desired address. Thus, the destination address may change as the datagram propagates through the network. The source address is never changed. The TTL value is decremented.

After a decision has been made to send the datagram out on an interface (and the destination address possibly changed), the datagram is handed to the normal IP output routines and is therefore subject to access lists, if they are present on the output interface.

Use the **no ip forward-protocol spanning-tree** command to prevent flooding of IP broadcasts.

Limiting Broadcast Storms

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

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

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

UDP Broadcasts

Network hosts occasionally employ UDP broadcasts to determine address, configuration, and name information. (UDP stands for User Datagram Protocol, an alternative to TCP for connectionless networks. UDP is defined in RFC 768.) If such a host is on a network segment that does not include a server host, UDP broadcasts fail.

To correct this situation, configure the interface of your Cisco router to forward certain classes of UDP broadcasts to a helper address. See the description of the **ip helper-address** and the **ip forward-protocol** subcommands in this chapter for more information.

Configuring ICMP and Other IP Services

The Internet Control Message Protocol (ICMP) is a special protocol within the IP protocol suite that focuses exclusively on control and management of IP connections. ICMP messages are generated by routers that discover a problem with the IP part of a packet's header; these messages could be alerting another router, or they could be sent to the source or destination device (host). Characteristics of the ICMP messages follow.

- The router listens to ICMP *Destination Unreachable* messages for packets that it originated.
- If the value in the time-to-live (TTL) field of a packet falls to zero, the router sends an ICMP *Time Exceeded* message to the source of the packet and discards the packet.
- If the router receives the ICMP *Information Request* or ICMP *Timestamp Request* message, it responds with an ICMP *Information Reply* or *Timestamp Reply* message.
- During the process of obtaining configuration information from network servers, the router sends broadcast ICMP *Mask Request* messages to determine subnet definitions for the local networks.

The **ip mask-reply** interface subcommand tells the router to respond to mask requests. The full syntax of this command follows.

ip mask-reply
no ip mask-reply

The default is not to send *a Mask Reply*, and this default is restored with the **no ip mask-reply** command.

Each router interface has an output hold queue with a limited number of entries that it can store. Upon reaching this limit, the interface sends an ICMP *Source Quench* message to the source host of any additional packets and discards the packet. When the interface empties the hold queue by one or more packets, the interface can accept new packets again. The router limits the rate at which it sends *Source Quench* and *Unreachable* messages to one per second.

Generating Unreachable Messages

If the router receives a nonbroadcast packet destined for itself that uses a protocol the router does not recognize, it sends an ICMP *Protocol Unreachable* message to the source.

If the router receives a datagram which it is unable to deliver to its ultimate destination because it knows of no route to the destination address, it replies to the originator of that datagram with an ICMP *Host Unreachable* message. Use the **ip unreachables** interface subcommand to enable or disable the sending of these messages. The full syntax for this command follows.

ip unreachables
no ip unreachables

The **no ip unreachables** subcommand disables sending ICMP unreachable messages on an interface.

Generating Redirect Messages

The Cisco router sends an ICMP *Redirect* message to the originator of any datagram that it is forced to resend through the same interface on which it was received, since the originating host could presumably have sent that datagram to the ultimate destination without involving the router at all. The router ignores *Redirect* messages that have been sent to it by other routers. Use the **ip redirects** interface subcommand to enable or disable the sending of these messages, as follows:

ip redirects
no ip redirects

Setting and Adjusting Packet Sizes

All interfaces have a default maximum packet size or MTU. You can set the IP MTU to a smaller unit by using the **ip mtu** interface subcommand. If an IP packet exceeds the MTU set for the router's interface, the router will fragment it. The full command syntax follows.

ip mtu *bytes*
no ip mtu

The default and maximum MTU depends on the interface medium type. The minimum MTU is 128 bytes. The **no ip mtu** subcommand restores the default MTU for that interface.

Example:

In the following example, you are setting the maximum IP packet size for the first serial interface to 300 bytes.

```
interface serial 0
ip mtu 300
```

MTU Path Discovery

All Cisco routers running software release 8.3 or later have the IP MTU Path Discovery mechanism running by default. IP Path MTU Discovery allows a host to dynamically discover and cope with differences in the maximum allowable MTU size of the various links along the path. Sometimes a router is unable to forward a datagram because fragmentation of the datagram is required (the packet is larger than the MTU you set for the interface with the **ip mtu** command), but the “Don’t fragment” bit is set. If you have Path Discovery enabled, the router sends a message to the sending host, alerting it to the problem. The host will have to replicate packets destined for the receiving interface so that they fit the smallest packet size of all the links along the path. This technique is defined by RFC 1191 and shown in .

Figure 1-6 MTU Path Discovery

MTU Path Discovery is useful when a link in a network goes down, forcing use of another, different MTU-sized link (and different routers). As an example, suppose one were trying to send IP packets over a network where the MTU in the first router is set to 1500 bytes, but then reaches a router where the MTU is set to 512 bytes. If the datagram’s “Don’t fragment” bit is set, the datagram would be dropped because the 512-router is unable to forward it. The router returns an ICMP *Destination Unreachable* message to the source of the datagram with its Code field indicating “Fragmentation needed and DF set.” To support Path MTU Discovery, it would also include the MTU of the next-hop network link in the low-order bits of an unused header field.

MTU Path Discovery is also useful when a connection is first being established and the sender has no information at all about the intervening links. It is always advisable to use the largest MTU that the links will bear; the larger the MTU, the fewer packets the host needs to send.

You can test that MTU Path Discovery is functioning using the **ping** command; see the following section and the section “The IP Ping Command” later in this chapter for more information.

The Ping Function

When you use the privileged EXEC command **ping** (IP packet internet groper function), the router sends ICMP *Echo* messages to check host reachability and network connectivity. If the router receives an ICMP *Echo* message, it sends an ICMP *Echo Reply m*essage to the source of the ICMP *Echo* message. See the section “The IP Ping Command” later in this chapter for more information about the use of the **ping** command.

Configuring Internet Header Options

The router supports the Internet header options *Strict Source Route*, *Loose Source Route*, *Record Route*, and *Time Stamp.*

The router examines the header options to every packet that passes through it. If it finds a packet with an invalid option, the router sends an ICMP *Parameter Problem* message to the source of the packet and discards the packet.

You can use the extended command mode of the **ping** command to specify several Internet header options. To see the list of the options you can specify, type a question mark at the extended commands prompt of the **ping** command.

Configuring IP Host-Name-to-Address Conversion

The router maintains a cache of host-name-to-address mappings for use by the EXEC **connect** or **telnet** commands and related Telnet support operations. This cache speeds the process of converting names to addresses.

Defining Static Name-to-Address Mappings

To define a static host-name-to-address mapping in the host cache, use the **ip host** global configuration command, as shown below:

ip host *name address*

The argument *name* is the host name, and the argument *address* is the associated IP address. Additional addresses may be bound to a host name by repeated use of the **ip host** subcommand.

Example:

The following example uses the **ip host** command to define two static mappings.

```
ip host croff 192.31.7.18
ip host bisso-gw 10.2.0.2 192.31.7.33
```

Configuring Dynamic Name Lookup

You can specify that the Domain Name System (DNS) or IEN-116 Name Server automatically determines host-name-to-address mappings. Use these global configuration commands to establish different forms of dynamic name lookup:

- **ip name-server**
- **ip domain-name**
- **ip ipname-lookup**
- **ip domain-lookup**

Name Server

To specify one or more hosts that supply name information, use the **ip name-server** global configuration command, as follows:

ip name-server *server-address1* [*server-address2 . . . server-address6*]

The arguments *server-address* are the Internet addresses of up to six name servers.

Example:

This command specifies host *131.108.1.111* as the primary name server, and host *131.108.1.2* as the secondary server.

```
ip name-server 131.108.1.111   131.108.1.2
```

Domain Name

The global configuration command **ip domain-name** defines a default domain name the router uses to complete unqualified host names (names without a dotted domain name appended to them). The full syntax of this command follows:

ip domain-name *name*
no ip domain-name

The argument *name i*s the domain name; do not include the initial period that separates an unqualified name from the domain name. The **no ip domain-name** command disables use of the Domain Name System.

Example:

This command defines *cisco.com* to be used as the default name.

```
ip domain-name cisco.com
```

Any IP host name that does not contain a domain name, that is, any name without a dot (.), will have the dot and cisco.com appended to it before being added to the host table.

Domain Lookup

By default, the IP Domain Name System-based host-name-to-address translation is enabled. To enable or disable this feature, use the **ip domain-lookup** global configuration command as follows:

ip domain-lookup
no ip domain-lookup

IP Name Lookup

To specify the IP IEN-116 Name Server host-name-to-address translation, use the **ip ip name-lookup** global configuration command as follows:

ip ipname-lookup
no ip ipname-lookup

This command is disabled by default; the **no ip ipname-lookup** command restores the default.

HP Probe Proxy Support

HP Probe Proxy support allows a router to respond to HP Probe Proxy Name requests. This will typically be used at sites which have HP equipment and are already using HP Probe. Use the interface subcommand **ip probe proxy**, to enable or disable HP Proxy Probe, as follows:

ip probe proxy
no ip probe proxy

To use the proxy service, you must first enter the host name of the HP host into the host table through the configuration command **ip hp-host**. Full syntax follows:

ip hp-host *hostname ip-address*
no ip hp-host *hostname ip-address*

The *hostname* argument specifies the host's name and the argument *ip-address* specifies it's IP address. Use the **no ip hp-host** command with the appropriate arguments to remove the host name.

Example:

The following example specifies an HP host's name and address, and then enables Probe proxy.

```
ip hp-host BCWjo 131.108.1.27
interface ethernet 0
ip probe proxy
```

Commands that will help you to maintain and debug your HP-based network are listed in the sections "Monitoring the IP Network" and "Debugging the IP Network" at the end of this chapter.

Establishing Domain Lists

To define a list of default domain names to complete unqualified host names, use the **ip domain-list** global configuration command. The full syntax of this command follows.

ip domain-list *name*
no ip domain-list *name*

The **ip domain-list** command is similar to the **ip domain-name** command, except that with **ip domain-list** you can define a list of domains, each to be tried in turn.

The argument *name* is the domain name; do not enter an initial period. Specify only one *name* when you enter the **ip domain-list** command.

Use the **no ip domain-list** command with the appropriate argument to delete a name from the list.

Example 1:

In the example below, several domain names are added to a list:

```
ip domain-list martinez.com
ip domain-list stanford.edu
```

Example 2:

The example below adds a name to, and then deletes a name from the list:

```
ip domain-list sunya.edu
no ip domain-list stanford.edu
```

Note: If there is no domain list, the default domain name is used.

Configuring IP Access Lists

An *access list* is a sequential collection of permit and deny conditions that apply to Internet addresses. The router tests addresses against the conditions in an access list one by one. The first match determines whether the router accepts or rejects the address. Because the router stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the router rejects the address.

The two steps involved in using access lists are:

- Create a list
- Apply it to implement a policy

You apply access lists in several ways:

- Controlling the transmission of packets on an interface
- Controlling virtual terminal line access
- Restricting contents of routing updates

The Cisco software supports two styles of access lists for IP. The standard IP access lists have a single address for matching operations. Extended IP access lists have two addresses with optional protocol type information for matching operations.

Note: Keep in mind when making the access list that, by default, the end of the access list contains an implicit deny statement for *everything* that has not been permitted. Plan your access conditions carefully and be aware of this implicit deny.

Configuring Standard Access Lists

To create an access list, use the **access-list** global configuration command. Full command syntax follows:

> **access-list** *list* {**permit**|**deny**} *address wildcard-mask*
> **no access-list** *list*

The argument *list* is an integer from 1 through 99 that you assign to identify one or more permit/deny conditions as an access list. Access list 0 (zero) is predefined; it permits any address and is the default access list for all interfaces.

The router compares the address being tested to *address*, ignoring any bits specified in *wildcard-mask*. If you use the keyword **permit,** a match causes the address to be accepted. If you use the keyword **deny,** a match causes the address to be rejected.

The arguments *address* and *wildcard-mask* are 32-bit quantities written in dotted-decimal format. Address bits corresponding to wildcard mask bits set to 1 are ignored in comparisons; address bits corresponding to wildcard mask bits set to zero are used in comparisons. See the examples later in this section.

An access list can contain an indefinite number of actual and wildcard addresses. A wildcard address has a non-zero address mask and thus potentially matches more than one actual address. The router examines first the actual address, then the wildcard addresses. The order of the wildcard addresses is important because the router stops examining access-list entries after it finds a match.

The **no access-list** subcommand deletes the entire access list. To display the contents of all access lists, use the EXEC command **show access-lists.**

Example:

The following access list allows access for only those hosts on the three specified networks. It assumes that subnetting is not used; the masks apply to the host portions of the network addresses.

```
access-list 1 permit 192.5.34.0  0.0.0.255
access-list 1 permit 128.88.1.0  0.0.255.255
access-list 1 permit 36.0.0.0  0.255.255.255
```

To specify a large number of individual addresses more easily, you can omit the address mask that is all zeros from the **access-list** configuration command. Thus, the following two configuration commands are identical in effect:

```
access-list 2 permit 36.48.0.3
access-list 2 permit 36.48.0.3  0.0.0.0
```

Configuring Extended Access Lists

Extended access lists allow finer granularity of control. They allow you to specify both source and destination addresses and some protocol and port number specifications.

To define an extended access list, use the extended version of the **access-list** subcommand.

access-list *list* {**permit**|**deny**} *protocol source source-mask destination destination-mask* [*operator operand*] [**established**]

The argument *list* is an integer from 100 through 199 that you assign to identify one or more extended permit/deny conditions as an extended access list. Note that a list number in the range 100 to 199 distinguishes an extended access list from a standard access list. The condition keywords **permit** and **deny** determine whether the router allows or disallows a connection when a packet matches an access condition. The router stops checking the extended access list after a match occurs.

The argument *protocol* is one of the following keywords:

- **ip**
- **tcp**
- **udp**
- **icmp**

Use the keyword **ip** to match any Internet protocol, including TCP, UDP, and ICMP.

The argument *source* is an Internet source address in dotted-decimal format. The argument *source-mask* is a mask, also in dotted-decimal format, of source address bits to be ignored. The router uses the *source* and *source-mask* arguments to match the source address of a packet. For example, to match any address on a Class C network 192.31.7.0, the argument *source-mask* would be 0.0.0.255. The arguments *destination* and *destination-mask* are dotted-decimal values for matching the destination address of a packet.

To differentiate further among packets, you can specify the optional arguments *operator* and *operand* to compare destination ports, service access points, or contact names. Note that the **ip** and **icmp** protocol keywords do not allow port distinctions.

For the **tcp** and **udp** protocol keywords, the argument *operator* can be one of these keywords:

- **lt**—less than
- **gt**—greater than

- **eq**—equal
- **neq**—not equal

The argument *operand* is the decimal destination port for the specified protocol.

For the TCP protocol there is an additional keyword, **established**, that does not take an argument. A match occurs if the TCP datagram has the ACK or RST bits set, indicating an established connection. The non-matching case is that of the initial TCP datagram to form a connection; the software goes on to other rules in the access list to determine if a connection is allowed in the first place.

Ethernet to Internet Example

For an example of using an extended access list, suppose you have an Ethernet-to-Internet routing network, and you want any host on the Ethernet to be able to form TCP connections to any host on the Internet. However, you do not want Internet hosts to be able to form TCP connections into the Ethernet except to the mail (SMTP) port of a dedicated mail host.

To do this, you must ensure that the initial request for an SMTP connection is made on TCP destination port 25 from port *X* where *X* is a number greater than 1023. The two port numbers continue to be used throughout the life of the connection, with the originator always using port 25 as the destination, and the acceptor always using port *X* as the destination. The fact that the secure system behind the router will always be accepting mail connections on port 25, with a foreign port number greater than 1023, is what makes it possible to separately allow/disallow incoming and outgoing services. Also remember that the access list used is that of the interface on which the packet would ordinarily be transmitted.

Example:

In the following example, the Ethernet network is a Class B network with the address 128.88.0.0 and the mail host’s address is 128.88.1.2.

```
access-list 101 permit tcp  128.88.0.0 0.0.255.255 0.0.0.0 255.255.255.255
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.0.0 0.0.255.255 estab-
lished
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.1.2 eq 25
interface serial 0
access-group 101
interface ethernet 0
access-group 102
```

This is a complex example, designed to show the power of all the options we’ve just discussed. The **access group** interface subcommand will be described in detail shortly.

Controlling Line Access

To restrict incoming and outgoing connections between a particular virtual terminal line and the addresses in an access list, use the **access-class** line configuration subcommand. Full command syntax for this command follows:

access-class *list* {**in** | **out**}
no access-class *list* {**in** | **out**}

This command restricts connections on a line or group of lines to certain Internet addresses.

The argument *list* is an integer from 1 through 99 that identifies a specific access list of Internet addresses.

The keyword **in** applies to incoming connections, such as virtual terminals. The keyword **out** applies to outgoing Telnet connections.

The **no access-class** line configuration subcommand removes access restrictions on the line for the specified connections.

Example 1:

The following example defines an access list that permits only hosts on network *192.89.55.0* to connect to the virtual terminals on the router.

```
access-list 12 permit 192.89.55.0  0.0.0.255
line 1 5
access-class 12 in
```

Use the **access-class** keyword **out** to define the access checks made on outgoing connections. (A user who types a host name at the router prompt to initiate a Telnet connection is making an outgoing connection.)

Note: Set identical restrictions on all the virtual terminal lines, because a user may connect to any of them.

Example 2:

The following example defines an access list that denies connections to networks other than network *36.0.0.0* on terminal lines 1 through 5.

```
access-list 10 permit 36.0.0.0   0.255.255.255
line 1 5
access-class 10 out
```

To display the access lists for a particular terminal line, use the EXEC command **show line** and specify the line number.

Controlling Interface Access

To control access to an interface, use the **access-group** interface subcommand, as shown below:

ip access-group *list*

The argument *list* is an integer from 1 through 199 that specifies an access list.

After receiving and routing a packet to a controlled interface, the router checks the destination address of the packet against the access list. If the access list permits the address, the router transmits the packet. If the access list rejects the address, the router discards the packet and returns an ICMP *Destination Unreachable* message. Access lists are applied on *outbound* interfaces, to *outbound* traffic.

Note: If this statement is made without the access list number, the implicit deny default takes precedence and will deny all access.

Example:

The following example applies list 101:

```
interface ethernet 0
access-group 101
```

Configuring the IP Security Option (IPSO)

All aspects of the IP Security Option (IPSO) are set up using configuration commands. The Cisco IPSO support addresses both the Basic and Extended security options described in a draft of the IPSO circulated by the Defense Communications Agency. This draft document superceded RFC1038, but has not yet been published as an RFC.

The following list describes some of the abilities of the IP security option (IPSO).

- Defines security level on a per-interface basis.
- Defines single level or multilevel interfaces.
- Provides a label for incoming datagrams.
- Strips labels on a per-interface basis.
- Re-orders options to put any Basic Security Option first.
- Accepts or rejects messages with extended security options.

IPSO Definitions

The following definitions apply to the descriptions of IPSO in this section.

- **level**—The degree of sensitivity of information. For example, data marked TOPSECRET is more sensitive than data marked SECRET. Table 1-4 lists the level keywords used by the Cisco software and their corresponding bit patterns.
- **authority**—An organization that defines the set of security levels that will be used in a network. For example, the Genser authority consists of level names defined by the Defense Communications Agency (DCA). Table 1-5 lists the authority keywords used

by the Cisco software and their corresponding bit patterns.

- **label**—A combination of a security level and an authority or authorities.

Table 1-4 IPSO Level Keywords and Bit Patterns

Level Keyword	Bit Pattern
Reserved4	0000 0001
TopSecret	0011 1101
Secret	0101 1010
Confidential	1001 0110
Reserved3	0110 0110
Reserved2	1100 1100
Unclassified	1010 1011
Reserved1	1111 0001

Table 1-5 IPSO Authority Keywords and Bit Patterns

Authority Keyword	Bit Pattern
Genser	1000 0000
Siop-Esi	0100 0000
SCI	0010 0000
NSA	0001 0000

Disabling IPSO

The **no ip security** interface subcommand resets an interface to its default state, dedicated, unclassified Genser; no extended state is allowed.

no ip security

Use one of the **ip security** commands to enable other kinds of security.

Setting Security Classifications

The **ip security dedicated** interface subcommand sets the interface to the requested classification and authorities.

ip security dedicated *level authority* [*authority* . . .]

All traffic entering the system on this interface must have a security option that exactly matches this label. Any traffic leaving via this interface will have this label attached to it. The levels and authorities were listed previously in tables.

Example:

In the following example, we set a confidential level with Genser authority:

```
ip security dedicated confidential Genser
```

Setting a Range of Classifications

The **ip security multilevel** interface subcommand sets the interface to the requested range of classifications and authorities. All traffic entering or leaving the system must have a security option that falls within this range. The levels are set with this command:

ip security multilevel *level1* [*authority* **...**] **to** *level2 authority2* [*authority2...*]

Being within range requires that the following two conditions be met:

- The classification level must be greater than or equal to *level1*, and less than or equal to *level2*.
- The authority bits must be a superset of *authority1*, and a subset of *authority2*. *T*hat is, *authority1* specifies those authority bits that are required on a datagram, while *authority2* specifies the required bits plus any optional authorities that can also be included. If the *authority1* field is the empty set, then a datagram is required to specify any one or more of the authority bits in *authority2*.

Example:

In the example below, we specify levels Unclassified to Secret and NSA authority.

```
ip security multilevel unclassified to secret nsa
```

Modifying Security Levels

IPSO allows you to choose from several interface subcommands if you decide you need to modify your security levels.

Ignore Authority Field

The **ip security ignore-authorities** interface subcommand ignores the authorities field of all incoming datagrams. The value used in place of this field will be the authority value declared for the given interface. Full syntax for this command follows.

ip security ignore-authorities
no ip security ignore-authorities

This action is only allowed for single-level interfaces. Enter the **no ip security ignore-authorities** command to turn this function off.

Accept Unlabeled Datagrams

The **ip security implicit-labelling** interface subcommand accepts datagrams on the interface, even if they do not include a security option. If your interface has multilevel security set, you must use the second form of the command (because it specifies the precise level and authority to use when labeling the datagram, just like your original **ip security multilevel** subcommand.) The full syntax of the **ip security implicit-labelling** command follows.

ip security implicit-labelling
no ip security implicit-labelling

ip security implicit-labelling *level authority* [*authority* ...]
no ip security implicit-labelling *level authority* [*authority* ...]

Enter the **ip security implicit-labelling** command (optionally, with the appropriate arguments) to turn these functions off.

Example:

In the example below, an interface is set for security and will accept unlabeled datagrams.

```
ip security dedicated confidential genser
ip security implicit-labelling
```

Accept Datagrams with Extended Security Option

The **ip security extended-allowed** interface subcommand accepts datagrams on the interface that have an extended security option present. Full syntax is shown below:

ip security extended-allowed
no ip security extended-allowed

The default condition rejects the datagram immediately; the **no ip security extended-allowed** command restores this default.

Adding or Removing Security Option by Default

The **ip security add** interface subcommand ensures that all datagrams leaving the router on this interface contain a basic security option. Its full syntax is as follows.

ip security add
no ip security add

If an outgoing datagram does not have a security option present, this subcommand will add one as the first IP option. The security label added to the option field is the label that was computed for this datagram when it first entered the router. Because this action is performed after all the security tests have been passed, this label will either be the same as or will fall within the range of the interface. This action is always enforced on multilevel interfaces.

The **ip security strip** interface subcommand removes any basic security option that may be present on a datagram leaving the router through this interface. The full syntax of this command follows.

ip security strip
no ip security strip

This is performed after all security tests in the router have been passed and is not allowed for multilevel interfaces.

Prioritizing the Presence of a Security Option

The **ip security first** interface subcommand prioritizes the presence of security options on a datagram. The full syntax of this command is as shown:

ip security first
no ip security first

If a basic security option is present on an outgoing datagram, but it is not the first IP option, then it is moved to the front of the options field when this subcommand is used.

Default Values for Minor Keywords

In order to fully comply with IPSO, the default values for the minor keywords have become complex:

- The default for all of the minor keywords is *off*, with the exception of **implicit-labelling** and **add.**
- The default value of **implicit-labelling** is *on*, if the interface is unclassified genser, and otherwise *off*.
- The default value for **add** is *on* if the interface is not unclassified genser, and otherwise *off*.

Table 1-6 provides a list of all default values.

Table 1-6 Default Security Keyword Values

Type	**Level**	**Authority**	**Implicit**	**Add**
none	(none)	(none)	on	off
dedicated	Unclassified	Genser	on	off
dedicated	any	any	off	on
multilevel	any	any	off	on

The default value for an interface is “dedicated, unclassified genser.” Note that this implies implicit labeling. This may seem unusual, but it makes the system entirely transparent to datagrams without options. This is the setting generated when the **no ip security** subcommand is given.

IPSO Configuration Examples

In this first example, three Ethernet interfaces are presented. These interfaces are running at security levels of Confidential Genser, Secret Genser, and Confidential to Secret Genser as shown in .

Figure 1-7 IPSO Security Levels

Example 1:

The following commands set up interfaces for the configuration in .

```
interface ethernet 0
ip security dedicated confidential genser
interface ethernet 1
ip security dedicated secret genser
interface ethernet 2
ip security multilevel confidential genser to secret genser
end
```

It is possible for the set up to be much more complex.

Example 2:

In this next example, there are devices on Ethernet 0 that cannot generate a security option, and so must accept datagrams without a security option. These hosts also crash when they receive a security option, therefore, never place one on such interfaces. Furthermore, there are hosts on the other two networks that are using the extended security option to communicate information, so you must allow these to pass through the system. Finally, there is also a host on Ethernet 2 that requires the security option to be the first option present, and this condition must also be specified. The new configuration follows.

```
interface ethernet 0
ip security dedicated confidential genser
ip security implicit-labelling
ip security strip
interface ethernet 1
```

```
ip security dedicated secret genser
ip security extended-allowed
!
interface ethernet 2
ip security multilevel confidential genser to secret genser
ip security extended-allowed
ip security first
```

Debugging IPSO

Debugging of security-related problems can be performed by using the EXEC command **debug ip-packet**. Each time a datagram fails any security test in the system, a message is logged describing the exact cause of failure.

Security failure is also reported to the sending host when allowed by the specification. This calculation on whether to send an error message can be somewhat confusing. It depends upon both the security label in the datagram and the label of the incoming interface. First, the label contained in the datagram is examined for anything obviously wrong. If nothing is wrong, it should be assumed to be correct. If there is something wrong, then the datagram should be treated as *unclassified genser*. Then this label is compared to the interface range, and the appropriate action is taken.

Table 1-7 Security Actions

Classification	Authorities	Action Taken
Too low	Too low	No Response
	Good	No Response
	Too high	No Response
In range	Too Low	No Response
	Good	Accept
	Too high	Send Error
Too high	Too Low	No Response
	In range	Send Error
	Too high	Send Error

The range of ICMP error messages that can be generated by the security code is very small. The only possible error messages are:

- `ICMP Parameter problem, code 0` — Error at pointer
- `ICMP Parameter problem, code 1` — Missing option
- `ICMP Parameter problem, code 2` — See Note, below
- `ICMP Unreachable, code 10` — Administratively prohibited

Note: The message `ICMP Parameter problem, code 2` identifies a very specific error that occurs in the processing of a datagram. This message indicates that a datagram containing a maximum length IP header, but no security option, was received by the router. After being processed and routed to another interface, it is discovered that the outgoing interface is marked with "add a security label." Since the IP header is already full, the system cannot add a label and must drop the datagram and return an error message.

Configuring IP Accounting

IP accounting is enabled on a per-interface basis. The IP accounting support records the number of bytes and packets switched through the system on a source and destination IP address basis. Only transit IP traffic is measured and only on an outbound basis; traffic generated by the router, or terminating in the router, is not included in the accounting statistics.

Enabling IP Accounting

The interface subcommand **ip accounting** enables or disables IP accounting for transit traffic outbound on an interface. Full syntax of this command follows.

ip accounting
no ip accounting

It does not matter whether or not IP fast-switching or IP access lists are being used on that interface; the numbers will be accurate; however, IP accounting does not keep statistics if autonomous switching is set.

Defining Maximum Entries

The global configuration command **ip accounting-threshold** enables or disables IP accounting for transit traffic outbound on an interface, as follows.

ip accounting-threshold *threshold*
no ip accounting-threshold *threshold*

The accounting threshold defines the maximum number of entries (source and destination address pairs) that the router accumulates, preventing IP accounting from possibly consuming all available free memory. This level of memory consumption could occur in a router that is switching traffic for many hosts. The default threshold value is 512 entries. Overflows will be recorded; see the monitoring commands for display formats.

Example:

The following example sets the IP accounting threshold to only 500 entries.

```
ip accounting-threshold 500
```

Specifying Accounting Filters

Use the **ip accounting-list** global configuration command to filter accounting information for hosts. The full syntax for this command follows.

ip accounting-list *ip-address mask*
no ip accounting-list *ip-address mask*

The source and destination address of each IP datagram is logically ANDed with the *mask* and compared with *ip-address*. If there is a match, the information about the IP datagram will be entered into the accounting database. If there is no match, then the IP datagram is considered a *transit* datagram and will be counted according to the setting of the **ip accounting-transits** command described next.

Use the **no ip accounting-list** command with the appropriate argument to remove this function.

Controlling the Number of Transit Records

The **ip accounting-transits** global configuration command controls the number of transit records that will be stored in the IP accounting database. The full syntax of this command is as follows.

ip accounting-transits *count*
no ip accounting-transits *count*

Transit entries are those that do not match any of the filters specified by **ip accounting-list** commands. If you do not define filters, the router will not maintain transit entries. To maintain accurate accounting totals, the router software maintains two accounting databases: an active and a checkpointed database.

Use the **no ip accounting-transits** command to remove this function.

Example:

The following example specifies that no more than 100 transit records are stored.

```
ip accounting-transit 100
```

Use the EXEC command **show ip accounting** to display the active accounting database. The EXEC command show ip accounting checkpoint displays the checkpointed database. The EXEC command **clear ip accounting** clears the active database and creates the checkpointed database. See the sections “Maintaining the IP Network” and “Monitoring the IP Network” later in this chapter for more options on monitoring your network’s accounting.

Special IP Configurations

This section discusses how to configure static routes, source routing, how to control IP processing on serial interfaces, and how to manage fast-switching.

Configuring Source Routing

The command **no ip source-route** causes the system to discard any IP datagram containing a source-route option. The **ip source-route** global configuration subcommand allows the router to handle IP datagrams with source routing header options.

ip source-route
no ip source-route

The default behavior is to perform the source routing.

IP Processing on a Serial Interface

The **ip unnumbered** interface subcommand enables IP processing on a serial interface, but does not assign an explicit IP address to the interface. The full command syntax is shown below:

ip unnumbered *interface-name*
no ip unnumbered *interface-name*

The argument *interface-name* is the name of another interface on which the router has an assigned IP address. The interface may not be another unnumbered interface, or the interface itself.

Whenever the unnumbered interface generates a packet (for example, for a routing update), it uses the address of the specified interface as the source address of the IP packet. It also uses the address of the specified interface in determining which routing processes are sending updates over the unnumbered interface. Restrictions include:

- Only those serial interfaces using HDLC encapsulation may be unnumbered. It is not possible to use this subcommand with X.25 interfaces.
- You cannot use the **ping** command to determine if the interface is up, since the interface has no address. SNMP (Simple Network Management Protocol) may be used to remotely monitor interface status.
- You cannot netboot a runnable image over an unnumbered serial interface.
- You cannot support IP security options on an unnumbered interface.
- The argument *interface-name* is the name of another interface in the network server which has an IP address, not another unnumbered interface.
- The interface defined by the *interface-name* argument must be enabled (listed as “up” in the **show interfaces** command display) for dynamic routing to work properly.

Example:

In the example below, the first serial interface is given Ethernet 0's address.

```
        interface ethernet 0
        ip address 131.108.6.6 255.255.255.0
        interface serial 0
        ip unnumbered ethernet 0
```

Configuring Simplex Ethernet Interfaces

The **transmit-interface** interface subcommand assigns a transmit interface to a receive-only interface.

transmit-interface *interface-name*

When a route is learned on this receive-only interface, the interface designated as the source of the route is converted to *interface-name*. This is useful in setting up dynamic IP routing over a simplex circuit, that is, a circuit that receives only or transmits only. When packets are routed out *interface-name*, they are sent to the IP address of the source of the routing update. To reach this IP address on a transmit-only Ethernet link, a static ARP entry mapping this IP address to the hardware address of the other end of the link is required.

Example:

This example illustrates how to configure IP on two routers sharing transmit only and receive only Ethernet connections.

Figure 1-8 Simplex Ethernet Connections

Example for Router 1:

```
   interface ethernet 0
   ip address 128.9.1.1
   !
   interface ethernet 1
   ip address 128.9.1.2
   transmit-interface ethernet 0
   !
   !use show interfaces command to find router2-MAC-address-E0
   arp router2-MAC-address-E0 128.9.1.4 arpa
```

Example for Router 2:

```
interface ethernet 0
ip address 128.9.1.3
transmit-interface ethernet 1
!
interface ethernet 1
ip address 128.9.1.4
!
!use show interfaces command to find router1-MAC-address-E1
arp router1-MAC-address-E1 128.9.1.1 arpa
```

Enabling Fast Switching

The **ip route-cache** interface subcommand controls the use of outgoing packets on a high-speed switching cache for IP routing. The route cache is enabled by default and allows load-balancing on a *per-destination* basis.

ip route-cache
no ip route-cache

To enable load-balancing on a *per-packet* basis, use the **no ip route-cache** command to disable fast-switching.

Cisco routers generally offer better packet transfer performance when fast switching is enabled with one exception. On networks using slow serial links (56K and below) disabling fast switching to enable the per-packet load-sharing is usually the better choice.

Enabling IP Autonomous Switching

Autonomous switching gives a router faster packet processing by allowing the cBus to switch packets independently, without interrupting the system processor. It works only in AGS+ systems with high-speed network controller cards, such as the CSC-MEC and CSC-FCI, and with a cBus controller card running microcode version 1.4 or later. (See the “Microcode Revisions” section in the release notes accompanying this publication for other microcode revision requirements.)

Autonomous switching is enabled by adding the **cbus** keyword to the existing **ip route-cache** interface subcommand. The syntax to enable and disable this function follows.

ip route-cache [cbus]
no ip route-cache [cbus]

By default, IP autonomous switching is not enabled. The **ip route-cache** command sets up fast switching, and by default, fast switching is enabled on all MCI/cBus interfaces.

To turn *on* both fast switching and autonomous switching use this syntax:

ip route-cache cbus

To turn *off* both fast and autonomous switching on an interface, add the **no** keyword as shown below:

```
no ip route-cache
```

To turn off autonomous switching only on an interface, use this syntax:

```
no ip route-cache cbus
```

To return to the default, use the standard ip route-cache command:

```
ip route-cache
```

This turns fast switching on and autonomous switching off.

TCP Header Compression

You can compress the TCP headers of your Internet packets in order to reduce the size of your packets. TCP header compression is only supported on serial lines using HDLC encapsulation. (RFC1144 specifies the compression process.) Compressing the TCP header can speed up Telnet connections dramatically. In general, TCP header compression is advantageous when your traffic consists of many small packets, not for traffic that consists of large packets. Transaction-processing (using terminals, usually) tends to use small packets while file transfers use large packets. This feature only compresses the TCP header, of course, so it has no effect on UDP packets or other protocol headers.

The **ip tcp header-compression** interface subcommand enables header compression. Full command syntax for this command follows:

```
ip tcp header-compression [passive]
no ip tcp header-compression [passive]
```

If you use the optional **passive** keyword, outgoing packets are only compressed if TCP incoming packets on the same interface are compressed. Without the **passive** keyword, the router will compress all traffic. The **no ip tcp header-compression** command (the default) disables compression. You must enable compression on both ends of a serial connection.

When compression is enabled, fast switching is disabled which means that fast interfaces like T-1 can overload the router. Think about your network's traffic characteristics before using this command. See the section "Monitoring the IP Network" for more explanation of commands for monitoring your compressed traffic.

The **ip tcp compression-connection**s interface subcommand specifies the total number of header compression connections that can exist on an interface. Each connection sets up a compression cache entry, so you are, in effect, specifying the maximum number of cache entries and the size of the cache.

```
ip tcp compression-connections number
```

The argument *number* specifies the number of connections the cache will support. The default is 16; *number* can vary between 3 and 256, inclusive. Too few cache entries for the specific interface can lead to degraded performance while too many cache entries leads to wasted memory.

Note: Both ends of the serial connection must use the same number of cache entries.

Example:

In the following example, the first serial interface is set for header compression with a maximum often cache entries.

```
interface serial 0
ip tcp header-compression
ip tcp compression-connections 10
```

IP Configuration Examples

This section shows complete configuration examples for the most common configuration situations.

Configuring Serial Interfaces

In the example below, the second serial interface is given ethernet 0's address. The serial interface is unnumbered.

Example:

```
interface ethernet 0
ip address 145.22.4.67 255.255.255.0
interface serial 1
ip unnumbered ethernet 0
```

Flooding of IP Broadcasts

In this example, flooding of IP broadcasts is enabled on all interfaces (two Ethernet and two serial). No bridging is permitted. (The access list denies all protocols.) No specific UDP protocols are listed by a separate **ip forward-protocol udp** command, so the default protocols (TFTP, DNS, IEN-116, Time, NetBios, and BootP) will be flooded.

Example:

```
ip forward-protocol spanning-tree
bridge 1 protocol dec
access-list 201 deny 0x0000 0xFFFF
interface ethernet 0
bridge-group 1
bridge-group 1 input-type-list 201
interface ethernet 1
bridge-group 1
bridge-group 1 input-type-list 201
```

```
interface serial 0
bridge-group 1
bridge-group 1 input-type-list 201
interface serial 1
bridge-group 1
bridge-group 1 input-type-list 201
```

Creating a Network from Separated Subnets

In the example below, networks 132 and 196 are separated by a backbone as shown in . The two networks are brought into the same logical network through the use of secondary addresses.

Figure 1-9 Creating a Network from Separated Subnets

Example—Router B:

```
interface ethernet 2
ip address 192.5.10.1 255.255.255.0
ip address 131.108.3.1 255.255.255.0 secondary
```

Example—Router C:

```
interface ethernet 1
ip address 192.5.10.2 255.255.255.0
ip address 131.108.3.2 255.255.255.0 secondary
```

Customizing ICMP Services

The example below changes some of the ICMP defaults for the first Ethernet interface. Disabling the sending of redirects could mean you don't think your routers on this segment will ever have to send a redirect. Lowering the error processing load on your router would be an efficiency move in this case. Disabling the unreachables messages will have a secondary effect—it will also disable MTU path discovery because path discovery works by having routers send unreachables messages. If you have a network segment with a small number of devices and an absolutely reliable traffic pattern—which could easily happen on a segment with a small number of little-used user devices—this would disable options your router would be unlikely to need to use anyway.

Example:

```
interface ethernet 0
no ip unreachables
no ip redirects
```

Helper Addresses

In this example, one server is on network *191.24.1.0*. and the other is on network *110.44.0.0*, and you want to permit IP broadcasts from all hosts to reach these servers. The example below illustrates how to configure the router that connects network 110 to network 191.

Figure 1-10 IP Helper Addresses

Example:

```
ip forward-protocol udp
!
interface ethernet 1
ip helper-address 110.44.23.7
interface ethernet 2
ip helper-address 191.24.1.19
```

HP Hosts on a Network Segment

The following example has a network segment with Hewlett-Packard devices on it. The commands listed customize the router's first Ethernet port to accommodate the HP devices.

Example:

```
ip hp-host bl4zip 131.24.6.27
interface ethernet 0
arp probe
ip probe proxy
```

Establishing IP Domains

The example below establishes a domain list with several alternate domain names.

Example:

```
ip domain-list cisco.com
ip domain-list telecomprog.edu
ip domain-list merit.edu
```

Configuring Access Lists

In the next example, network *36.0.0.0* is a Class A network whose second octet specifies a subnet; that is, its subnet mask is *255.255.0.0*. The third and fourth octets of a network *36.0.0.0* address specify a particular host. Using access list 2, the router would accept one address on subnet 48 and reject all others on that subnet. The router would accept addresses on all other network *36.0.0.0* subnets; that is the purpose of the last line of the list.

Example:

```
access-list 2 permit 36.48.0.3  0.0.0.0
access-list 2 deny 36.48.0.0  0.0.255.255
access-list 2 permit 36.0.0.0  0.255.255.255
interface ethernet 0
access-group 2
```

Configuring Extended Access Lists

In the example below, the first line permits any incoming TCP connections with destination port greater than 1023. The second line permits incoming TCP connections to the SMTP port of host *128.88.1.2*. The last line permits incoming ICMP messages for error feedback.

Example:

```
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.0.0 0.0.255.255 gt 1023
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.1.2 0.0.0.0 eq 25
access-list 102 permit icmp 0.0.0.0 255.255.255.255 128.88.0.0 255.255.255.255
interface ethernet 0
access-group 102
```

Maintaining the IP Network

Use the EXEC commands described in this section to maintain IP routing caches, tables, and databases.

Removing Dynamic Entries from the ARP Cache

The **clear arp-cache** EXEC command removes all dynamic entries from the Address Resolution Protocol (ARP) cache, and clears the fast-switching cache. Enter this command at the EXEC prompt:

clear arp-cache

Removing Entries from the Host-Name-and-Address Cache

Use the EXEC command **clear host** to removes one or all entries from the host-name-and-address cache, depending upon the argument you specify.

clear host {*name*|*****}

To remove a particular entry, use the argument *name* to specify the host. To clear the entire cache, use the asterisk (*) argument.

Clearing the Checkpointed Database

Use the **clear ip accounting** command to clear the active database when IP accounting is enabled. Use the **clear ip accounting checkpoint** command to clear the checkpointed database when IP accounting is enabled. You may also clear the checkpointed database by issuing the **clear ip accounting** command twice in succession. Enter one of these commands at the EXEC prompt.

clear ip accounting
clear ip accounting [**checkpoint**]

Removing Routes

Use the **clear ip route** command to remove a route from the IP routing table. Enter this command at the EXEC prompt:

clear ip route {*network*|*****}

The optional argument *network* is the network or subnet address of the route that you want to remove. Use the asterisk (*) argument to clear the entire routing table.

Monitoring the IP Network

Use the EXEC commands described in this section to obtain displays of activity on the IP network.

Displaying the IP Show Commands

Use the **show ip ?** command to display a list of all the available EXEC commands for monitoring the IP network. Following is sample output:

```
accounting <checkpoint>    Accounting statistics
arp                        IP ARP table
bgp <address>              Border Gateway Protocol
cache                      Fast switching cache
egp                        EGP peers
interface <name>           Interface settings
protocols                  Routing processes
route <network>            Routing table
tcp <keyword>              TCP information, type "show ip tcp ?" for list
traffic                    Traffic statistics
```

Displaying the ARP Cache

To display the ARP cache, use the following EXEC command:

show ip arp

This command displays the contents of the ARP cache. ARP establishes correspondences between network addresses (an IP address, for example) and LAN hardware addresses (Ethernet addresses). A record of each correspondence is kept in a cache for a predetermined amount of time and then discarded. Following is sample output.Table 1-8 describes the fields seen.

```
Protocol  Address          Age (min)     Hardware Addr   Type      Interface
AppleTalk 4.57                     0     aa00.0400.6408  ARPA      Ethernet0
Internet  131.108.1.140          137     aa00.0400.6408  ARPA      Ethernet0
Internet  131.108.1.111          156     0800.2007.8866  ARPA      Ethernet0
AppleTalk 4.128                    0     aa00.0400.6508  ARPA      Ethernet0
AppleTalk 4.129                    -     aa00.0400.0134  ARPA      Ethernet0
Internet  131.108.1.115           33     0000.0c01.0509  ARPA      Ethernet0
Internet  192.31.7.24              5     0800.0900.46fa  ARPA      Ethernet2
Internet  192.31.7.26             41     aa00.0400.6508  ARPA      Ethernet2
Internet  192.31.7.27              -     aa00.0400.0134  ARPA      Ethernet2
Internet  192.31.7.28             67     0000.0c00.2c83  ARPA      Ethernet2
Internet  192.31.7.17             67     2424.c01f.0711  ARPA      Ethernet2
Internet  192.31.7.18             64     0000.0c00.6fbf  ARPA      Ethernet2
Internet  192.31.7.21            114     2424.c01f.0715  ARPA      Ethernet2
Internet  131.108.1.33            15     0800.2008.c52e  ARPA      Ethernet0
Internet  131.108.1.55            44     0800.200a.bbfe  ARPA      Ethernet0
Internet  131.108.1.6             89     aa00.0400.6508  ARPA      Ethernet0
Internet  131.108.7.1              -     0000.0c00.750f  ARPA      Ethernet3
Internet  131.108.1.1              -     aa00.0400.0134  ARPA      Ethernet0
Internet  131.108.1.27            75     0800.200a.8674  ARPA    Ethernet0
```

Table 1-8 Show IP Arp Field Displays

Field	Description
Protocol	Protocol for network address in the Address field
Address	The network address that corresponds to Hardware Addr
Age (min)	Age, in minutes, of the cache entry
Hardware Addr	LAN hardware address that corresponds to network address
Type	Type of ARP (Address Resolution Protocol):
	ARPA = Ethernet-type ARP
	SNAP = RFC 1042 ARP
	Probe = HP Probe Protocol

Displaying IP Accounting

The **show ip accounting** command displays the active accounting database. The **show ip accounting checkpoint** command displays the checkpointed database.

show ip accounting

show ip accounting checkpoint

Following is sample output for the **show ip accounting** and **show ip accounting checkpoint** commands:

```
   Source          Destination             Packets            Bytes
 131.108.19.40    192.67.67.20                   7              306
 131.108.13.55    192.67.67.20                  67             2749
 131.108.2.50     192.12.33.51                  17             1111
 131.108.2.50     130.93.2.1                     5              319
 131.108.2.50     130.93.1.2                   463            30991
 131.108.19.40    130.93.2.1                     4              262
 131.108.19.40    130.93.1.2                    28             2552
 131.108.20.2     128.18.6.100                  39             2184
 131.108.13.55    130.93.1.2                    35             3020
 131.108.19.40    192.12.33.51                1986            95091
 131.108.2.50     192.67.67.20                 233            14908
 131.108.13.28    192.67.67.53                 390            24817
 131.108.13.55    192.12.33.51              214669          9806659
 131.108.13.111   128.18.6.23                27739          1126607
 131.108.13.44    192.12.33.51               35412          1523980
 192.31.7.21      130.93.1.2                    11              824
 131.108.13.28    192.12.33.2                   21             1762
 131.108.2.166    192.31.7.130                 797           141054
 131.108.3.11     192.67.67.53                   4              246
 192.31.7.21      192.12.33.51               15696           695635
 192.31.7.24      192.67.67.20                  21              916
 131.108.13.111   128.18.10.1                   16             1137
```

The output lists the source and destination addresses, as well as total number of packets and bytes for each address pair.

Displaying Host Statistics

The **show hosts** command displays the default domain name, the style of name lookup service, a list of name server hosts, and the cached list of host names and addresses.

show hosts

Enter **show host**s at the user-level prompt.

Following is sample output:

```
show hosts
Default domain is CISCO.COM
Name/address lookup uses domain service
Name servers are 255.255.255.255
Host                     Flags      Age Type   Address(es)
SLAG.CISCO.COM           (temp, OK)  1   IP    131.108.4.10
CHAR.CISCO.COM           (temp, OK)  8   IP    192.31.7.50
CHAOS.CISCO.COM          (temp, OK)  8   IP    131.108.1.115
DIRT.CISCO.COM           (temp, EX)  8   IP    131.108.1.111
DUSTBIN.CISCO.COM        (temp, EX)  0   IP    131.108.1.27
DREGS.CISCO.COM          (temp, EX) 24   IP    131.108.1.30
```

In the display:

- A temp entry in the Flags field is entered by a name server; the router removes the entry after 72 hours of inactivity.
- A perm entry is entered by a configuration command and is not timed out. Entries marked OK are believed to be valid. Entries marked ?? are considered suspect and subject to revalidation. Entries marked EX are expired.
- The Age field indicates the number of hours since the router last referred to the cache entry. The Type field identifies the type of address, for example, IP, CLNS, or X.121.
- The Address(es) field shows the address of the host. One host may have up to eight addresses.

If you have used the **ip hp-host** configuration command (see the section “HP Probe Proxy Support”), the **show hosts** command will display these host names as type `HP-IP`.

Displaying the Route Cache

The **show ip cache** command displays the routing table cache that is used to rapidly switch Internet traffic. Enter this command at the EXEC prompt:

show ip cache

Following is sample output:

```
IP routing cache version 435, entries 19/20, memory 880

 Hash   Destination     Interface  MAC Header
*6D/0   128.18.1.254    Serial0    0F000800
*81/0   131.108.1.111   Ethernet0  00000C002C83AA00040002340800
*8D/0   131.108.13.111  Ethernet0  AA0004000134AA00040002340800
 99/0   128.18.10.1     Serial0    0F000800
*9B/0   128.18.10.3     Serial0    0F000800
*B0/0   128.18.5.39     Serial0    0F000800
*B6/0   128.18.3.39     Serial0    0F000800
*C0/0   131.108.12.35   Ethernet0  AA0004000134AA00040002340800
*C4/0   131.108.2.41    Ethernet0  00000C002C83AA00040002340800
*C9/0   192.31.7.17     Ethernet0  2424C01F0711AA00040002340800
*CD/0   192.31.7.21     Ethernet0  2424C01F0715AA00040002340800
*D5/0   131.108.13.55   Ethernet0  AA0004006508AA00040002340800
*DC/0   130.93.1.2      Serial0    0F000800
*DE/0   192.12.33.51    Serial0    0F000800
*DF/0   131.108.2.50    Ethernet0  AA0004000134AA00040002340800
*E7/0   131.108.3.11    Ethernet0  00000C002C83AA00040002340800
*EF/0   192.12.33.2     Serial0    0F000800
*F5/0   192.67.67.53    Serial0    0F000800
*F5/1   131.108.1.27    Ethernet0  AA0004006508AA00040002340800
*FE/0   131.108.13.28   Ethernet0  AA0004006508AA00040002340800
```

In the display:

- The * designates valid routes.
- The Destination field shows the destination IP address.
- The Interface field specifies the interface type and number (serial 1, Ethernet 2, etc.).
- The MAC Header field displays the MAC header.

Displaying Interface Statistics

To display the usability status of interfaces, use the EXEC command **show interfaces.** If the interface hardware is usable, the interface is marked “up.” If the interface can provide two-way communication, the line protocol is marked “up.” For an interface to be usable, both the interface hardware and line protocol must be up.

show ip interface [*interface unit*]

If you specify an optional interface type, you will see only information on that specific interface.

If you specify no optional parameters you will see information on all the interfaces.

The following sample output was obtained by specifying the serial 0 interface:

```
Serial 0 is up, line protocol is up
  Internet address is 192.31.7.129, subnet mask is 255.255.255.240
  Broadcast address is 255.255.255.255
  Address determined by non-volatile memory
  MTU is 1500 bytes
  Helper address is 131.108.1.255
  Outgoing access list is not set
  Proxy ARP is enabled
  Security level is default
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  IP fast switching is enabled
  Gateway Discovery is disabled
  IP accounting is enabled, system threshold is 512
  TCP/IP header compression is disabled
  Probe proxy name replies are disabled
```

In the display:

- The Broadcast Address field shows the broadcast address.
- The Helper Address field specifies a helper address, if one has been set.
- The Outgoing Access List field indicates whether or not the interface has an outgoing access list set.
- The Proxy ARP field indicates whether Proxy ARP is enabled for the interface.
- The Security Level field specifies the IPSO security level set for this interface.
- The ICMP redirects field specifies whether redirects will be sent on this interface.
- The ICMP unreachables field specifies whether unreachable messages will be sent on this interface.
- The ICMP mask replies field specifies whether mask replies will be sent on this interface.
- The IP fast switching field specifies whether fast switching has been enabled for this interface. It is generally enabled on serial interfaces, such as this one.
- The Gateway Discovery field specifies whether the discovery process has been enabled for this interface. It is generally disabled on serial interfaces, such as this one.
- The IP accounting field specifies whether IP accounting is enabled for this interface and what the threshold (maximum number of entries) is.
- The TCP/IP header compression field indicates whether compression is enabled or disabled.
- The Probe proxy name field indicates whether the function is enabled or disabled.

Displaying the Routing Table

The **show ip route** command displays the IP routing table. Enter this command at the EXEC prompt:

show ip route [*network*]

A specific network in the routing table is displayed when the optional *network* argument is entered.

Following is sample output with the optional network argument:

```
Routing entry for 131.108.1.0
  Known via "igrp 109", distance 100, metric 1200
  Redistributing via igrp 109
  Last update from 131.108.6.7 on Ethernet0, 35 seconds ago
  Routing Descriptor Blocks:
  * 131.108.6.7, from 131.108.6.7, 35 seconds ago, via Ethernet0
      Route metric is 1200, traffic share count is 1
      Total delay is 2000 microseconds, minimum bandwidth is 10000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 0
```

This display is the result of the **show ip route** command without the network number:

```
Codes: I - IGRP derived, R - RIP derived, H - HELLO derived
       C - connected, S - static, E - EGP derived, B - BGP derived
       * - candidate default route

Gateway of last resort is 131.108.6.7 to network 131.119.0.0

I*Net 128.145.0.0 [100/1020300] via 131.108.6.6, 30 sec, Ethernet0
I Net 192.68.151.0 [100/160550] via 131.108.6.6, 30 sec, Ethernet0
I Net 128.18.0.0 [100/8776] via 131.108.6.7, 58 sec, Ethernet0
                            via 131.108.6.6, 31 sec, Ethernet0
E Net 128.128.0.0 [140/4] via 131.108.6.64, 130 sec, Ethernet0
C Net 131.108.0.0 is subnetted (mask is 255.255.255.0), 54 subnets
I     131.108.144.0 [100/1310] via 131.108.6.7, 78 sec, Ethernet0
C     131.108.91.0 is directly connected, Ethernet1
```

The output begins by showing the address of the gateway of last resort for this network. In the rest of the display:

- The first field specifies how the route was derived. The options are listed above the routing table.
- The second field specifies a remote network/subnet to which a route exists. The first number in brackets is the administrative distance of the information source; the second number is the metric for the route.
- The third field specifies the IP address of a router that is the next hop to the remote network.
- The fourth field specifies the number of seconds since this network was last heard.
- The final field specifies the interface through which you can reach the remote network via the specified router.

Displaying Protocol Traffic Statistics

The **show ip traffic** command displays IP protocol statistics. Enter this command at the EXEC prompt:

show ip traffic

Following is sample output:

```
IP statistics:
  Rcvd:  98 total, 98 local destination
         0 format errors, 0 checksum errors, 0 bad hop count
         0 unknown protocol, 0 not a gateway
         0 security failures, 0 bad options
  Frags: 0 reassembled, 0 timeouts, 0 too big
         0 fragmented, 0 couldn't fragment
  Bcast: 38 received, 52 sent
  Sent:  44 generated, 0 forwarded
         0 encapsulation failed, 0 no route

ICMP statistics:
  Rcvd: 0 checksum errors, 0 redirects, 0 unreachable, 0 echo
        0 echo reply, 0 mask requests, 0 mask replies, 0 quench
        0 parameter, 0 timestamp, 0 info request, 0 other
  Sent: 0 redirects, 3 unreachable, 0 echo, 0 echo reply
        0 mask requests, 0 mask replies, 0 quench, 0 timestamp
        0 info reply, 0 time exceeded, 0 parameter problem

UDP statistics:
  Rcvd: 56 total, 0 checksum errors, 55 no port
  Sent: 18 total, 0 forwarded broadcasts

TCP statistics:
  Rcvd: 0 total, 0 checksum errors, 0 no port
  Sent: 0 total

EGP statistics:
  Rcvd: 0 total, 0 format errors, 0 checksum errors, 0 no listener
  Sent: 0 total

IGRP statistics:
  Rcvd: 73 total, 0 checksum errors
  Sent: 26 total

HELLO statistics:
  Rcvd: 0 total, 0 checksum errors
  Sent: 0 total

ARP statistics:
  Rcvd: 20 requests, 17 replies, 0 reverse, 0 other
  Sent: 0 requests, 9 replies (0 proxy), 0 reverse

Probe statistics:
  Rcvd: 6 address requests, 0 address replies
        0 proxy name requests, 0 other
  Sent: 0 address requests, 4 address replies (0 proxy)
        0 proxy name replies
```

In the display:

- A format error is a gross error in the packet format, such as an impossible Internet header length.
- A bad hop count occurs when a packet is discarded because its time-to-live (TTL) field was decremented to zero.
- An encapsulation failure usually indicates that the router received no reply to an ARP request and therefore did not send a datagram.
- A no route occurrence is counted when the router discards a datagram it did not know how to route.
- A proxy reply is counted when the router sends an ARP or Probe Reply on behalf of another host. The display shows the number of probe proxy requests that have been received and the number of responses that have been sent.

Monitoring TCP Header Compression

The **show ip tcp header-compression** command shows statistics on compression. Enter this command at the EXEC prompt:

show ip tcp header-compression

Following is sample output:

```
TCP/IP header compression statistics:
  Interface Serial1: (passive, compressing)
    Rcvd:    4060 total, 2891 compressed, 0 unknown type, 0 errors
             0 dropped, 1 buffer copies, 0 buffer failures
    Sent:    4284 total, 3224 compressed,
             105295 bytes saved, 661973 bytes sent
             1.15 efficiency improvement factor
    Connect: 16 slots, 1543 long searches, 2 misses, 99% hit ratio
             Five minute miss rate 0 misses/sec, 0 max misses/sec
```

In the display:

- The buffer copies are the number of packets that had to be copied into bigger buffers for decompression.
- The report buffer failures is the number of packets dropped due to a lack of buffers.
- The efficiency improvement factor is the improvement in line efficiency because of TCP header compression.
- The number of slots is the size of the cache.
- The long searches field indicates the number of times the software had to look “hard” to find a match.
- The misses field indicates the number of times a match could not be made. If your output shows a large miss rate, then the number of allowable simultaneous compression connections may be too small.

- The hit ratio is the percentage of times the software found a match and was able to compress the header.
- The Five minute miss rate calculates the miss rate over the previous five minutes for a longer-term (and more accurate) look at miss rate trends.

The IP Ping Command

The EXEC command **ping** allows the administrator to diagnose network connectivity by sending ICMP *Echo Request* messages and waiting for ICMP *Echo Reply* messages. The following sample session shows **ping** command output for IP:

Sample Session 1:

```
Protocol [ip]:
Target IP address: 131.108.1.27
Repeat count [5]:
Datagram size [100]: 1000
Timeout in seconds [2]:
Extended commands [n]: yes
Source address:
Type of service [0]:
Set DF bit in IP header? [no]: yes
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Sending 5, 1000-byte ICMP Echos to 131.108.2.27, timeout is 2 seconds:
M.M.M
Success rate is 60 percent, round-trip min/avg/max = 4/6/12 ms
```

The **ping** command uses the following notation to indicate the responses it sees:

Table 1-9 Ping Test Characters

Char	**Meaning**
!	Each exclamation point indicates receipt of a reply.
.	Each period indicates the network server timed out while waiting for a reply.
U	Destination unreachable error PDU received.
N	Network unreachable.
P	Protocol unreachable.
Q	Source quench.
M	Could not fragment.
?	Unknown packet type.

To abort a ping session, type the escape sequence (by default, Ctrl-^, X).

The IP **ping** command, in verbose mode, accepts a data pattern. The pattern is specified as a 16-bit hexadecimal number. The default pattern is 0xABCD. Patterns such as all ones or all zeros can be used to debug data sensitivity problems on CSU/DSUs.

Note: If the IP version of the **ping** command is used on a directly connected interface, the packet is sent out the interface and should be forwarded back to the router from the far end. The time travelled reflects this round trip route. This feature can be useful for diagnosing serial line problems. By placing the local or remote CSU/DSU into loopback mode and pinging your own interface, you can isolate the problem to the router or leased line.

Sample Session 2:

You can also specify the router address to use as the source address for ping packets, which here is *131.108.105.62*.

```
Sandbox#ping
Protocol [ip]:
Target IP address: 131.108.1.111
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: yes
Source address: 131.108.105.62
Type of service [0]:
Set DF bit in IP header? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 131.108.1.111, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 4/4/4 ms
```

The IP Trace Command

The EXEC command **trace** allows you to discover the routing path your router's packets are taking through your network. It sends probe packets to destination hosts and routers and takes advantage of the ICMP message packets that are generated when a packet exceeds its time-to-live (TTL) value.

The **trace** command offers default and settable parameters for specifying a simple or extended trace mode.

How Trace Works

The **trace** command works by taking advantage of the error messages generated by routers when a datagram exceeds its time-to-live (TTL) value.

The **trace** command starts by sending probe datagrams with a TTL value of one. This causes the first router to discard the probe datagram and send back an error message. The **trace** command sends several probes at each TTL level and displays the round trip time for each.

The **trace** command sends out one probe at a time. Each outgoing packet may result in one or two error messages. A *time exceeded* error message indicates that an intermediate router has seen and discarded the probe. A *destination unreachable* error message indicates that the destination node has received the probe and discarded it because it could not deliver the packet. If the timer goes off before a response comes in, **trace** prints an asterisk (*).

The **trace** command terminates when the destination responds, when the maximum TTL was exceeded, or when the user interrupts the trace with the escape sequence, by default Ctrl-^,X).

Common Trace Problems

Due to bugs in the IP implementations of various hosts and routers, the **trace** command may behave in odd ways.

Not all destinations will correctly respond to a *probe* message by sending back an *ICMP port unreachable* message. A long sequence of TTL levels with only asterisks, terminating only when the maximum TTL has been reached, may indicate this problem.

There is a known problem with the way some hosts handle an *ICMP TTL exceeded* message. Some hosts generate an *ICMP* message but they re-use the TTL of the incoming packet. Since this is zero, the ICMP packets do not make it back. When you trace the path to such a host, you may see a set of TTL values with asterisks (*). Eventually the TTL gets high enough that the *ICMP* message can get back. For example, if the host is six hops away, **trace** will timeout on responses 6 through 11.

Tracing IP Routes

When tracing IP routes, the following **trace** command parameters may be set:

- Target IP address. You must enter a host name or an IP address. There is no default.
- Source Address. One of the interface addresses of the router to use as a source address for the probes. The router will normally pick what it feels is the best source address to use.
- Numeric Display. The default is to have both a symbolic and numeric display; however, you may suppress the symbolic display.
- Timeout in seconds. The number of seconds to wait for a response to a probe packet. The default is three seconds.
- Probe count. This is the number of probes to be sent at each TTL level. The default count is 3.
- Minimum Time to Live [1]: The TTL value for the first probes. The default is 1, but may be set to a higher value to suppress the display of known hops.
- Maximum Time to Live [30]: This is the largest TTL value which may be used. The

default is 30. The **trace** command terminates when the destination is reached or when this value is reached.

- Port Number. This is the destination port used by the UDP probe messages. The default is 33,434.
- Loose, Strict, Record, Timestamp, Verbose. These are IP header options. You may specify any combination. The trace command issues prompts for the required fields. Note that trace will place the requested options in each probe; however, there is no guarantee that all routers (or end-nodes) will process the options.
- Loose Source Routing. You may specify a list of nodes which must be traversed when going to the destination.
- Strict Source Routing. You may specify a list of nodes which must be the only nodes traversed when going to the destination.
- Record. You may specify the number of hops to leave room for.
- Timestamp. You may specify the number of time stamps to leave room for.
- Verbose. If you select any option, the verbose mode is automatically selected and **trace** prints the contents of the option field in any incoming packets. You can prevent verbose mode by selecting it again, toggling its current setting.

The following table describes the output from this test.

Table 1-10 Trace Test Characters

Char	**Meaning**
nn msec	The probe was successfully returned in *nn* milliseconds.
*	The probe timed out.
?	Unknown packet type.
Q	Source quench.
P	Protocol unreachable.
N	Network unreachable.
U	Host unreachable.

Sample Session 1:

The following is an example of the simple use of **trace**.

```
chaos#trace ABA.NYC.mil
Type escape sequence to abort.
Tracing the route to ABA.NYC.mil (26.0.0.73)
  1 DEBRIS.CISCO.COM (131.108.1.6) 1000 msec 8 msec 4 msec
  2 BARRNET-GW.CISCO.COM (131.108.16.2) 8 msec 8 msec 8 msec
  3 EXTERNAL-A-GATEWAY.STANFORD.EDU (192.42.110.225) 8 msec 4 msec 4 msec
  4 BB2.SU.BARRNET.NET (131.119.254.6) 8 msec 8 msec 8 msec
  5 SU.ARC.BARRNET.NET (131.119.3.8) 12 msec 12 msec 8 msec
  6 MOFFETT-FLD-MB.in.MIL (192.52.195.1) 216 msec 120 msec 132 msec
  7 ABA.NYC.mil (26.0.0.73) 412 msec 628 msec 664 msec
```

Sample Session 2:

Following is an example of going through the extended dialog of the **trace** command.

```
chaos#trace
Protocol [ip]:
Target IP address: mit.edu
Source address:
Numeric display [n]:
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to MIT.EDU (18.72.2.1)
  1 DEBRIS.CISCO.COM (131.108.1.6) 1000 msec 4 msec 4 msec
  2 BARRNET-GW.CISCO.COM (131.108.16.2) 16 msec 4 msec 4 msec
  3 EXTERNAL-A-GATEWAY.STANFORD.EDU (192.42.110.225) 16 msec 4 msec 4 msec
  4 NSS13.BARRNET.NET (131.119.254.240) 112 msec 8 msec 8 msec
  5 SALT_LAKE_CITY.UT.NSS.NSF.NET (129.140.79.13) 72 msec 64 msec 72 msec
  6 ANN_ARBOR.MI.NSS.NSF.NET (129.140.81.15) 124 msec 124 msec 140 msec
  7 PRINCETON.NJ.NSS.NSF.NET (129.140.72.17) 164 msec 164 msec 172 msec
  8 ZAPHOD-GATEWAY.JVNC.NET (128.121.54.72) 172 msec 172 msec 180 msec
  9 HOTBLACK-GATEWAY.JVNC.NET (130.94.0.78) 180 msec 192 msec 176 msec
 10 CAPITAL1-GATEWAY.JVNC.NET (130.94.1.9) 280 msec 192 msec 176 msec
 11 CHEESESTEAK2-GATEWAY.JVNC.NET (130.94.33.250) 284 msec 216 msec 200 msec
 12 CHEESESTEAK1-GATEWAY.JVNC.NET (130.94.32.1) 268 msec 180 msec 176 msec
 13 BEANTOWN2-GATEWAY.JVNC.NET (130.94.27.250) 300 msec 188 msec 188 msec
 14 NEAR-GATEWAY.JVNC.NET (130.94.27.10) 288 msec 188 msec 200 msec
 15 IHTFP.MIT.EDU (192.54.222.1) 200 msec 208 msec 196 msec
 16 E40-03GW.MIT.EDU (18.68.0.11) 196 msec 200 msec 204 msec
 17 MIT.EDU (18.72.2.1) 268 msec 500 msec 200 msec
```

Debugging the IP Network

Use the EXEC commands described in this section to troubleshoot and monitor the IP network transactions. For each **debug** command there is an corresponding **undebug** command that turns the display off. In general, you need use these commands only during troubleshooting sessions with Cisco personnel, as display of debugging messages can impact the operation of the router.

debug arp

The **debug arp** command enables logging of ARP and Probe protocol transactions.

debug ip-icmp

The **debug ip-icmp** command enables logging of ICMP transactions. Refer to the ICMP section for an in-depth look at the various ICMP messages.

debug ip-packet [*list*]

The **debug ip-packet** command enables logging of general IP debugging information as well as IPSO security transactions. IP debugging information includes packets received, generated, and forwarded. This command can also be used to debug IPSO security-related problems. Each time a datagram fails a security test in the system, a message is logged describing the cause of failure. An optional IP access *list* may be specified. If the datagram is not permitted by that access list, then the related debugging output is suppressed.

debug ip-routing

The **debug ip-routing** command enables logging of routing table events such as network appearances and disappearances.

debug ip tcp

The **debug ip tcp** command enables logging of significant TCP transactions such as state changes, retransmissions, and duplicate packets.

debug ip-tcp-packet *list*

The **debug ip-tcp-packet** command enables logging of each TCP packet that meets the permit criteria specified in the access list.

debug ip-udp

The **debug ip-udp** command enables logging of UDP-based transactions.

debug probe

Debugging information, including information about HP Probe Proxy Requests, is available through **debug probe**.

debug ip-tcp-header-compression

The **debug ip-header-compression** command enables logging of TCP header compression statistics.

IP Global Configuration Command Summary

This section lists and summarizes all the commands you can use to configure your IP router. Commands are listed in alphabetical order.

[**no**] **access-list** *list* {**permit**|**deny**} *address wildcard-mask*

Creates or removes an access list. The argument *list* is an IP list number from 1 to 99. The keywords **permit** and **deny** specify the security action to take. The argument *address* is a 32-bit, dotted decimal notation IP address to which the router compares the address being tested. The argument *wildcard-mask* are wildcard mask bits for the address in 32-bit, dotted decimal notation.

[**no**] **access-list** *list* {**permit**|**deny**} *protocol source source-mask destination destination-mask* [*operator operand*] [**established**]

Creates or removes an extended access list. The argument *list* is an IP list number from 100 to 199. The keywords **permit** and **deny** specify the security action to take. The argument **protocol** is one of the supported protocol keywords—**ip**, **tcp**, **udp**, **icmp**. The argument *source* is a 32-bit, dotted decimal notation IP address. The argument *source-mask* are mask bits for the source address in 32-bit, dotted decimal notation. The arguments *destination* and *destination-mask* the destination address and mask bits for the destination address in 32-bit, dotted decimal notation. Using TCP and UDP, the optional arguments *operator* and *operand* can be used to compare destination ports, service access points, or contact names. The optional **established** keyword is for use in matching certain TCP datagrams (see “Configuring Extended Access Lists”).

[**no**] **arp** *internet-address hardware-address type* [**alias**]

Installs a permanent entry in the ARP cache. The router uses this entry to translate 32-bit Internet Protocol addresses into 48-bit hardware addresses. The argument *internet-address* is the Internet address in dotted decimal format corresponding to the local data link address specified by the argument *hardware-address*. The argument *type* is an encapsulation description—**arpa** for Ethernets; **snap** for FDDI and Token Ring interfaces; **ultra** for the Ultranet interfaces. The optional keyword **alias** indicates that the router should respond to ARP requests as if it were the owner of the specified IP address.

[**no**] **ip domain-list** *name*

Defines a list of default domain names to complete unqualified host name. The argument *name* is the domain name.

[**no**] **ip domain-name** *name*

Defines the default domain name, which is specified by the argument name. The router uses the default domain name to complete unqualified domain names—names without a dotted domain name.

[**no**] **ip accounting-list** *ip-address mask*

Specifies a set of filter to control the hosts for which IP accounting information is kept. The source and destination address of each IP datagram is logically ANDed with the *mask* and compared with *ip-address*. If there is a match, the information about the IP datagram will be entered into the accounting database. If there is no match, then the IP datagram is considered a transit datagram and will be counted according to the setting of the **ip accounting-transits** command.

[**no**] **ip accounting-threshold** *threshold*

Sets the maximum number of accounting entries to be created.

[**no**] **ip accounting-transits** *count*

Controls the number of transit records that will be stored in the IP accounting database. Transit entries are those that do not match any of the filters specified by **ip accounting-list** commands. If no filters are defined, no transit entries are possible.

[**no**] **ip default-network** *network*

Flags networks as candidates for default routes. The argument network specifies the network number.

[**no**] **ip domain-lookup**

Enables or disables IP Domain Name System-based host-name-to-address translation. Enabled by default. The **no** variation of the command disables the feature.

[**no**] **ip forward-protocol spanning-tree**

Permits IP broadcasts to be flooded throughout the internetwork in a controlled fashion. This command is an extension of the **ip helper-address** command, in that the same packets that may be subject to the helper address and forwarded to a single network may now be flooded.

[**no**] **ip host** *name address*

Defines a static host-name-to-address mapping in the host cache. The argument *name* is the host name and the argument *address* is the associated IP address.

[**no**] **ip hp-host** *hostname ip-address*

Enables the use of the proxy service. You enter the *hostname* of the HP host into the host table, along with its IP address.

[**no**] **ip ipname-lookup**

Specifies or removes the IP IEN-116 Name Server host-name-to-address translation.

This command is disabled by default; the **no** variation of the command restores the default.

[**no**] **ip name-server** *address*

Specifies the address of name server to use for name and address resolution. By default, the router uses the all-ones broadcast address (*255.255.255.255*).

[**no**] **ip routing**

Controls the system's ability to do IP routing. If the system is running optional bridging-enabled software, the **no ip routing** subcommand will turn off IP routing when setting up a system to bridge (as opposed to route) IP datagrams. The default setting is to perform IP routing.

[**no**] **ip source-route**

Controls the handling of IP datagrams with source routing header options. The default behavior is to perform the source routing. The **no** keyword causes the system to discard any IP datagram containing a source-route option.

[**no**] **ip subnet-zero**

Enables or disables the ability to configure and route to "subnet zero" subnets. The default condition is disabled.

IP Interface Subcommand Summary

This section lists and summarizes all the commands in the interface subcommand list for your IP router. Preceding any of these commands with a **no** keyword undoes their effect or restores the default condition. Commands are listed in alphabetical order.

[**no**] **arp** {**arpa**|**probe**|**snap**}

Controls the interface-specific handling of IP address resolution into 48-bit Ethernet, FDDI, and Token Ring hardware addresses. The keyword **arpa**, which is the default, specifies standard Ethernet style ARP (RFC 826), **probe** specifies the HP-proprietary Probe protocol for IEEE-802.3 networks, and **snap** specifies ARP packets conforming to RFC 1042.

[**no**] **arp timeout** *seconds*

Sets the number of seconds an ARP cache entry will stay in the cache. The value of the argument *seconds* is used to age an ARP cache entry related to that interface, and by default is set to14,400 seconds. A value of zero seconds sets no timeout. The **no** form of the command returns the default.

[**no**] **access-group** *list*

Defines an access group. This subcommand takes a standard or extended IP access list number as an argument.

[**no**] **ip accounting**

Enables or disables IP accounting on an interface.

[**no**] **ip address** *address subnet-mask* [**secondary**]

Sets an IP address for an interface. The two required arguments are an IP address and the subnet mask for the associated IP network. The subnet mask must be the same for all interfaces connected to subnets of the same network.

[**no**] **ip broadcast-address** *address*

Defines a broadcast address. The *address* argument is the desired IP broadcast address for a network. If a broadcast address is not specified, the system will default to a broadcast address of all ones or *255.255.255.255*

[**no**] **ip directed-broadcast**

Enables or disables forwarding of directed broadcasts on the interface. The default is to forward directed broadcasts.

[**no**] **ip forward-protocol** {**udp**|**nd**} [*port*]

Controls forwarding of physical and directed IP broadcasts. This command controls which protocols and ports are forwarded for an interface on which an **ip helper-address** command has been specified. The keyword **nd** is the ND protocol used by older diskless SUN workstations. The keyword **udp** is the UDP protocol. By default both UDP and ND forwarding are enabled if a helper address has been given for an interface.

[**no**] **ip helper-address** *address*

Defines a helper-address for a specified address. The helper-address defines the selective forwarding of UDP broadcasts, including BootP, received on the interface. The *address* argument specifies a destination broadcast or host address to be used when forwarding such datagrams.

[**no**] **ip mask-reply**

Sets the interface to send ICMP *Mask Reply* messages. The default is not to send *Mask Reply* messages.

[**no**] **ip mtu** *bytes*

Sets the maximum transmission unit (MTU) or size of IP packets sent on an interface. The argument *bytes* is the number of bytes with a minimum of 128 bytes. The **no** form of the command restores the default.

[**no**] **ip probe proxy**

Enables or disables HP Probe Proxy support, which allows a router to respond to HP Probe Proxy Name requests.

[**no**] **ip proxy-arp**

Enables or disables proxy ARP on the interface. The default is to perform proxy ARP.

[**no**] **ip redirects**

Disables sending ICMP redirects on the interface. ICMP redirects are normally sent.

[**no**] **ip route-cache**

Controls the use of outgoing packets on a high-speed switching cache for IP routing. The cache is enabled by default and allows load-balancing on a per-destination basis. To enable load-balancing on a per-packet basis, use the **no ip route-cache** to disable fast-switching.

[**no**] **ip security** *arguments*

Controls the use of the Internet IP Security Option.

[**no**] **ip security add**

Adds a basic security option to all datagrams leaving the router on the specified interface. The **no** form of the command disables the function.

ip security dedicated *level authority* [*authority...*]

Sets or unsets the requested level of classification and authority on the interface. See Tables 13-4 and 13-5 for the *level* and *authority* arguments.

[**no**] **ip security extended-allowed**

Allows or rejects datagrams with an extended security option on the specified interface.

[**no**] **ip security ignore-authorities**

Sets or unsets an interface to ignore the authority fields of all incoming datagrams.

[**no**] **ip security implicit-labelling** [*level authority* [*authority...*]]

In the simplest form, sets or unsets the interface to accept datagrams, even if they do not include a security option. With the arguments level and authority, a more precise condition is set. See Tables 13-4 and 13-5 for the *level* and *authority* arguments.

ip security multilevel *level1* [*authority...*] **to** *level2 authority2* [*authority2...*]

Sets or unsets the requested range of classification and authority on the interface. Traffic entering or leaving the system must have a security option that falls within the specified range. See Tables 13-4 and 13-5 for the *level* and *authority* arguments.

[**no**] **ip security strip**

Removes any basic security option on all datagrams leaving the router on the specified interface. The **no** form of the command disables the function.

[**no**] **ip tcp compression-connections** *number*

Sets the maximum number of connections per interface that the compression cache can support. Default is 16; *number* can vary from 3 to 256.

[**no**] **ip tcp header-compression** [**passive**]

Enables TCP header compression. The **no** keyword disables (the default) compression. The optional keyword **passive** sets the interface to only compress outgoing traffic on the interface for a specific destination if incoming traffic is compressed.

[**no**] **ip unnumbered** *interface-name*

Enables IP processing on a serial interface, but does not assign an explicit IP address to the interface. The argument *interface-name* is the name of another interface on which the router has assigned an IP address. The interface may not be another unnumbered interface, or the interface itself.

[**no**] **ip unreachables**

Enables or disables (with the no argument) sending ICMP unreachable messages on an interface. ICMP unreachables are normally sent.

transmit-interface *interface-name*

Assigns a transmit interface to a receive-only interface. When a route is learned on this receive-only interface, the interface designated as the source of the route is converted to *interface-name.*

IP Line Subcommand Summary

This section contains a list of the line subcommands used to configure IP routing.

[**no**] **access-class** *list* {**in**|**out**}

Restricts incoming and outgoing connections between a particular virtual terminal line and the addresses in an access list. Serves to restrict connections on a line or group of lines to certain Internet addresses. The argument *list* is an integer from 1 through 99 that identifies a specific access list of Internet addresses. The keyword **in** applies to incoming connections; the keyword **out** applies to outgoing Telnet connections.