Chapter 1
TCP/IP Configuration and Management

1

This chapter describes how to configure the TCP/IP protocol, and specifically, how Cisco Systems implements this protocol on its protocol translators. You will find the following information in this chapter:

- An overview of TCP/IP.
- Configuration procedures for Telnet lines and IP, and for setting up routing assistance methods whereby the protocol translator can learn routes to other networks.
- EXEC commands for troubleshooting TCP/IP connections.

Making Telnet connections is described in the chapter "Protocol Translator User Commands."

The commands to configure IP and the IP services and features are entered in configuration mode, which is privileged. To enter this mode, type the **configure** command at the EXEC prompt. For more information about the configuration mode, see the chapter "Startup and Basic Configuration."

A command summary is included at the end of this chapter.

Cisco's Implementation of TCP/IP

The Internet Protocol (IP) is a packet-based protocol used to exchange data over computer networks. IP handles addressing, fragmentation, re-assembly, and protocol demultiplexing. It is the foundation on which all other Internet protocols, collectively referred to as the Internet Protocol suite, are built.

Built upon the IP layer suite, the Transmission Control Protocol (TCP) is a connection-oriented protocol that specifies the format of data and acknowledgments that two computer systems exchange to transfer data. TCP also specifies the procedures the computers use to ensure that the data arrives correctly. TCP allows multiple applications on a system to communicate concurrently, as it handles all demultiplexing of the incoming traffic among the application programs.

The Cisco Systems implementation of TCP generally ensures good protocol translator performance on slow-speed links as well as high-speed LAN links. Cisco's TCP software includes Telnet, a simple remote terminal protocol that is part of the TCP/IP protocol suite. The software provides commands that allow you to turn on or off TCP services such as stream processing and debugging modes for monitoring the connection. Additionally, the software supports rlogin, the BSD UNIX remote login service.

In this chapter, the TCP and Telnet options are described first, followed by an overview of IP, including IP addressing, and then descriptions of the commands available for configuring IP.

Telnet Support

The Internet Protocol suite includes the simple remote terminal protocol called Telnet. Telnet allows a user at one site to establish a TCP connection to a login server at another site, then passes the keystrokes from one system to the other. Telnet can accept either an Internet address or a domain name as the remote system address. In short, Telnet offers three main services:

- Network Virtual Terminal Connection
- Option Negotiation
- Symmetric Connection

The Cisco Systems implementation of Telnet supports the following Telnet options:

- Remote Echo
- Binary Transmission
- Suppress Go Ahead
- Timing Mark
- Terminal Type
- Send Location
- Terminal Speed
- Remote Flow Control
- X display location

The following sections describe how to configure the lines to support Telnet connections. Making Telnet connections is described in the chapter "Protocol Translator User Commands."

This section describes the line subcommands for configuring Telnet protocol-specific lines. For information about the **configure** command and the **line** configuration command, see the chapter "System Configuration."

Refusing Full Duplex, Remote Echo Connections

The line configuration subcommand **telnet refuse-negotiations** causes Telnet to refuse to negotiate full duplex, Remote Echo options on incoming connections. The command has this simple syntax:

telnet refuse-negotiations

Use this command on reverse Telnet connections to allow the protocol translator to refuse these requests from the other end. This command suppresses negotiation of the Telnet Remote Echo and Suppress Go Ahead options.

Handling Different End-of-Line Interpretations

The line configuration subcommand **telnet transparent** causes the protocol translator to send a Return (CR) as a CR followed by a NULL instead of a CR followed by a Line Feed (LF). It has this simple syntax:

telnet transparent

This subcommand is useful for coping with different interpretations of end-of-line handling in the Telnet protocol specification.

Synchronizing the Break Signal

The line configuration subcommand **telnet sync-on-break** causes a reverse Telnet line to send a Telnet Synchronize signal when it receives a Telnet BREAK signal. The syntax of this command follows:

telnet sync-on-break

The TCP Synchronize signal clears the data path, but will still interpret incoming commands.

Generating a Hardware Break Signal

The **telnet break-on-ip** line configuration subcommand causes the system to generate a hardware Break signal on the RS-232 line that is associated with a reverse Telnet connection, when a Telnet Interrupt-Process (IP) command is received on that connection. This can be used to control the translation of Telnet IP commands into X.25 Break indications. The syntax for this command is:

telnet break-on-ip

Use this command to work around the following situations:

- Several user Telnet programs send an IP command, but cannot send a Telnet Break command.
- Some Telnet programs implement a Break command that sends an IP command.

A hardware BREAK signal is generated when a Telnet Break command is received.

Optimizing Response to User Interrupt Characters

When used with a correctly operating host, Cisco protocol translators implement the Telnet Synchronize and Abort Output signals, which can stop output within one packet's worth of data from the time the user types the interrupt character. For a faster response to user interrupt characters, use the **ip tcp chunk-size** *global configuration* command. The command has this syntax:

ip tcp chunk-size *number*

The argument *number* is the number of characters output before the interrupt executes. The suggested value of *number* is 80, which will typically abort output within a line or two of where the user types the interrupt character. Values of less than 50 are not recommended for reasons of efficiency.

Changing the chunk size affects neither the size of the packet used nor the TCP window size, either of which would cause serious efficiency problems for the remote host as well as for the protocol translator. Instead, the Telnet status is checked after the number of characters specified, causing only a relatively minor performance loss.

Example:

This command allows the protocol translator to react more quickly when an interrupt character or sequence is typed (Ctrl-C, for example).

```
ip tcp chunk-size 100
```

Telnet Line Configuration Example

The following example represents a typical listing for a modem configured for maximum transparency.

Example:

```
line 1-20
telnet transparent
telnet break-on-ip
```

Internet Addressing

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 computers (hosts) that have the same network field value in their Internet addresses.

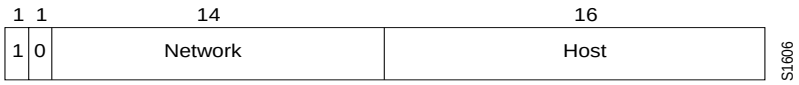
Internet Address Classes

The Class A Internet address format allocates the highest eight bits (octet) to the network field and sets the highest-order bit to 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 The Class A Internet Address Format

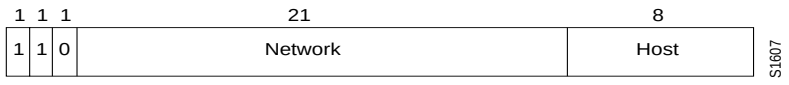
The Class B Internet address format allocates the highest 16 bits to the network field and sets the two highest-order bits to 1,0. The remaining 16 bits form the host field. Up to 16,384 Class B networks can exist, and each Class B network can have up to 65,536 hosts. Figure 1-2 illustrates the Class B address format.

Figure 1-2 The 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 2 million Class C networks can exist, and each Class C network can have up to 255 hosts. Figure 1-3 illustrates the Class C address format.

Figure 1-3 The 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 protocol translator 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 examples 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	Old broadcast address
	1.0.0.0 through 126.0.0.0	Available
	127.0.0.0	Reserved
	127.0.0.1	Software loopback
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 223.255.254.0	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

If the bits in the host portion of an address are all zero, that address refers to the network specified in the network portion of the address. For example, the Class C address 192.31.7.0 refers to a particular network.

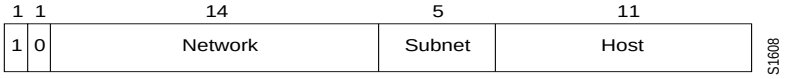
If the bits in the host portion of an address are all one, that address refers to all hosts on the network specified in the network portion of the address. For example, the Class B address 128.1.255.255 refers to all hosts on the 128.1.0.0 network.

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

Subnetting

Subnetting is a scheme for imposing a simple hierarchy on hosts that are connected using a single physical 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 RFC 950, "Internet Standard Subnetting Procedure."

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



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

Subnet Masks

A *subnet mask i*dentifies the subnet field of network addresses. 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 1-4, 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

Broadcast Addresses

A broadcast is a data packet destined for all hosts on a particular 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 protocol translator support for Internet broadcasts generally complies with RFC 919 and RFC 922; however, the protocol translator does not support multiple subnet 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. (Broadcast addresses that include the network or subnet fields are called *directed broadcasts*.) In the latter case, the sender need not be on the same network or subnet as the intended receiving hosts.

For example, if the network is 128.1.0.0, the address 128.1.255.255 indicates all hosts on network 128.1.0.0. If network 128.1.0.0 has a subnet mask of 255.255.255.0 (the third octet is the subnet field), the address 128.1.5.255 specifies all hosts on subnet 5 of network 128.1.0.0.

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.

The Cisco Systems protocol translators support different broadcast addresses. To change the Internet broadcast address, see "Sending Internet Broadcast Messages" later in this chapter.

Configuring IP

To configure IP on the Cisco protocol translator you must specify the IP address to use on the interface with the **ip address** command. This command also requests the subnet mask to identify the network portion of the address.

Cisco's IP implementation also provides commands to:

- Configure network services such as ICMP and UDP.
- Set up router discovery mechanisms.
- Configure access lists that filter incoming or outgoing network transmissions.

The following sections provide procedures for the tasks listed above. The commands for monitoring and debugging IP follow these sections.

Specifying the IP Address

To set the interface address and the subnet mask, use the **ip address** interface subcommand. The command has this syntax:

ip address *address subnet-mask*

The argument *address* is a Class A, Class B, or Class C Internet address for the interface. The argument *subnet-mask* is a mask of address bits that specifies the network portion of the address.

Example:

These commands sets a Class A Internet address for Ethernet 0 with a subnet of 52:

```
interface ethernet 0
ip address 15.240.0.52 255.255.255.0
```

Configuring TCP/IP Network Services

The following sections describe the IP network services supported by the Cisco TCP/IP software implementation. These services include:

- Internet Control Message Protocol (ICMP)
- Address Resolution Protocol (ARP)
- Reverse ARP
- Proxy ARP (Cisco feature)
- Boot Protocol (BootP)
- User Datagram Protocol (UDP)
- Domain Name Mechanism

In many cases, these services require no configuration. Procedures requiring configuration generally also require entering IP addresses. Refer to "Internet Addressing" in this chapter for an overview of the Internet addressing scheme. Refer to the list of references at the end of this manual.

Internet Control Message Protocol

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 when a problem is discovered with the IP part of a packet's header. The messages serve to alert another device on the network of a problem, or they may be sent to the source or destination device (host). The protocol translator listens to ICMP *Destination Unreachable* messages as it tries to create TCP connections to other hosts.

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

The protocol translator caches ICMP *Redirect* messages and uses them to decide which local router to use when sending a packet. To display the current cache of these messages, use the EXEC command **show ip redirects**. For more information, see the section "Monitoring TCP/IP" at the end of this chapter.

When you use the privileged EXEC command **ping**, the protocol translator sends ICMP *Echo* messages to check host reachability and network connectivity. If the protocol translator receives an ICMP Echo message, it sends an ICMP *Echo Reply* message to the source of the ICMP Echo message.

If the value in the time-to-live field of a packet falls to zero, or if the protocol translator cannot re-assemble a fragmented packet in the time allowed, the protocol translator sends an ICMP Time *Exceeded* message to the source of the packet and discards the packet.

If the protocol translator receives an 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 protocol translator sends broadcast ICMP *Mask Request* messages to determine subnet definitions for the local networks.

Internet Header Options

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

The protocol translator examines the header options of every packet that it receives. If it finds a packet with an invalid option, the protocol translator 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 privileged EXEC command **ping** to specify several of the 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.

Dynamic Address Resolution Protocol (ARP)

The protocol translator uses address Resolution Protocol (ARP), defined in RFC 826, to locate other hosts on the network. ARP is a protocol for mapping 32-bit Internet protocol addresses to 48-bit Ethernet or IEEE 802.3 hardware addresses.

To send an Internet data packet to a local host with which it has not previously communicated, the protocol translator first broadcasts an ARP Request packet. The ARP Request packet requests the Ethernet hardware address corresponding to an Internet address. All hosts on the local Ethernet receive the ARP Request, but only the host with the specified Internet address may respond.

If present and functioning, the host with the specified address responds with an ARP Reply packet containing its hardware address. The protocol translator receives the ARP Reply packet, stores the hardware address in its ARP cache for future use, and begins exchanging packets with the receiving host.

Specifying Permanent ARP Cache Entries

The global configuration command **arp** installs a permanent entry in the system ARP cache, which the protocol translator uses to translate 32-bit Internet Protocol addresses into 48-bit Ethernet or IEEE 802.3 hardware addresses that use the specified type of encapsulation. The full syntax of this command follows:

arp *internet-address hardware-address type-keyword*
no arp *internet-address*

The argument *internet-address* is the Internet address in dotted-decimal format corresponding to the hardware address specified by the argument *hardware-address*, specified as a dotted triple of hexadecimal digits. The argument *type-keyword* is the encapsulation keyword, and can be one of the following:

- **arpa**—Specifies standard Ethernet style ARP (RFC 826) encapsulation. This is the default encapsulation method for Ethernet interfaces.
- **snap**—Specifies ARP packets conforming to RFC 1042.
- **probe**—Specifies the HP-proprietary Probe protocol for IEEE-802.3 networks and resolves IEEE 802.3 or Ethernet local data link virtual addresses using the *Virtual Address Request and Reply* message.

The **no arp** command removes the specified entry from the ARP cache.

To remove all noninterface entries from the ARP cache, use the privileged EXEC command **clear arp-cache**.

Note: Because most hosts support the use of address resolution protocols to determine and cache address information (called *dynamic address resolution)*, you generally do not need to specify permanent ARP cache entries.

Example:

This example establishes ARPA Ethernet encapsulation for Internet address 192.31.7.19 and hardware address 0800.0900.1B34:

```
arp 192.31.7.19 0800.0900.1B34 arpa
```

Finding an Internet Address

The protocol translator uses both Reverse ARP (RARP) and Boot Protocol (BootP) messages when trying to obtain interface addresses from network servers. Reverse ARP, defined in RFC 903, works the same way as ARP, except that the RARP Request packet requests an Internet address instead of a hardware address. The Cisco Systems protocol translators use RARP during startup processing to attempt to determine their Internet addresses from other network servers.

BootP, defined in RFC 951, specifies a method for determining the Internet address of a host from its Ethernet hardware address. The basic mechanism is similar to that used by RARP, but it is UDP-based rather than a distinct Ethernet protocol. (For more information, see "Sending UDP Broadcasts" later in this chapter.) The main advantage of BootP is that its messages can be routed through routers, whereas RARP messages cannot leave the local Ethernet.

Sending Internet Broadcast Messages

The protocol translator supports Internet broadcasts on both local and wide area networks. At least four ways exist to indicate the Internet broadcast address. You can configure a protocol translator host to generate any form of Internet broadcast address. The protocol translator can also receive and understand any form of Internet broadcast address.

By default, a protocol translator host 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 setting jumpers in the processor configuration register. Setting bit 10 causes the protocol translator 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 protocol translator to include the network and subnet portions of its address in the broadcast address.

Table 1-3 shows the combined effect of bits 10 and 14. For more information about the configuration register, see the hardware installation and reference publication for your particular product.

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>

Changing the Internet Broadcast Address

If the protocol translator has the nonvolatile memory option, you can also change the Internet broadcast address by using the interface subcommand **ip broadcast-address**. The command has this syntax:

ip broadcast-address *address*

The argument *address* specifies an Internet address for the protocol translator to use in all broadcasts. The default broadcast address is 255.255.255.255.

Note: The protocol translator recognizes most forms of the broadcast address, but generates only one type of address when broadcasting on its own behalf.

Sending UDP Broadcasts

The protocol translator occasionally employs UDP broadcasts to determine address, configuration, and name information. (UDP stands for User Datagram Protocol, defined in RFC 768.) If the protocol translator is on a network segment that does not include a server host, UDP broadcasts fail.

To correct this situation, you can configure an interface of a Cisco Systems router to forward certain classes of UDP broadcasts to a specified address. See the Cisco *Router Products Configuration and Reference* publication for more information.

Assigning an Internet Address to a Service

The **ip alias** global configuration command assigns an Internet address to the service provided on a TCP port.

ip alias *internet-address TCP-port*
no ip alias *internet-address*

The argument *internet-address* is the Internet address for the service, and the argument *TCP-port* is the number of the TCP port. Note that *internet-address* must be on the same network or subnet as the protocol translator's main address, and must not be used by another host on that network or subnet. Connecting to *internet-address* has the same effect as connecting to the protocol translator's main address, using *TCP-port* as the TCP port.

You can use the **ip alias** command to assign multiple Internet addresses to the protocol translator. For example, in addition to the primary alias address, you can specify addresses that correspond to lines or rotary groups. Using the **ip alias** command in this way makes the process of connecting to a specific rotary group transparent to the user.

The **no ip alias** command removes the specified address for the protocol translator.

Example:

This command configures connections to IP address 131.108.42.42 to act identically to connections made to the server's primary IP address on TCP port 3001. In other words, the user trying to connect will get connected to the first free line in rotary group 1 using the Telnet protocol.

```
ip alias 131.108.42.42  3001
```

Defining Static Host Name to Address Mappings

The global configuration command **ip host** enters a static host-name-to-address mapping in the host cache.

ip host *name* **[***TCP-port***]** *internet-address1* **[***internet-address2 . . . internet-address8***]**
no ip host *name*

The argument *name* is the host name, and the optional argument *TCP-port* is the TCP port number (in decimal). If you omit the argument *TCP-port*, the Telnet port is assumed. You may specify up to eight Internet address arguments (the first address argument is required), separated by spaces.

To remove a host cache entry, use the privileged EXEC command **clear host**. See the section "Maintaining TCP/IP" at the end of this chapter for more information.

Example:

You can use the **ip host** command to set up aliases for hosts and for host-and-port pairs.

```
ip host chaff 192.31.7.17
ip host printer 2034 192.31.7.65
ip host cisco-gw 192.31.7.18  128.5.0.67
```

In the above example, the host *chaff* corresponds to host address 192.31.7.17, printer is TCP port 2034 at Internet address 192.31.7.65, and *cisco-gw* is a host with two addresses.

Configuring Dynamic Name Lookup

You can specify that the Domain Name System (DNS) 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**

To have one or more hosts supply name information, use the **ip name-server** global configuration command. The full syntax of the command follows:

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

The *server-address* arguments are the Internet addresses of up to six name servers. By default, the protocol translator uses the all-ones broadcast address (255.255.255.255).

Example:

This command specifies host 131.108.1.111 as the primary name server, and host 131.108.1.2 as the secondary server. Domain name requests, which by default are controlled by the **service domain** command, are sent to each address in the list until a response is received.

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

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

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

The argument *name* is 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:

The following defines *cisco.com* to be used as the default name. 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 domain name system.

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

By default, 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

The **no** variation of the command disables the feature; the **ip domain-lookup** command restores the default.

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

ip ipname-lookup
no ip ipname-lookup

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

Establishing Domain Lists

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

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

This 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.

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:

In the example below a name is added and another one is deleted from the list.

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

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

Setting Up Routing Assistance

The protocol translator software provides three methods by which the protocol translator can learn about routes to other networks:

- Using Proxy ARP
- Using a default router
- Using the Router Discovery mechanism and Gateway Discovery Protocol (GDP)

The first and most common method is by using proxy ARP. The software treats all networks as if they are local, and performs ARPs for every IP address. The appropriate router to that network answers the ARP request, allowing the software to reach any network. This method works so long as the routers support proxy ARP. While this is true of Cisco routers, many other routers, and especially host-based routing software, do not support this feature.

The second method for locating routes is to define a default router (described in the subsequent section "Using the Default Gateway"). The software sends all nonlocal packets to this router, which would route them appropriately, and perhaps send an ICMP *redirect* message back to the protocol translator, telling it of a better router. The software would cache these redirect messages, and route each packet thereafter as efficiently as possible. The problem with this method is that there is no means to detect when the default router had crashed or was unavailable, and no method of picking another router is possible in that event.

The Cisco protocol translator software provides a third method, called *Router Discovery*, by which the protocol translator may learn about routes to other networks using the Cisco-defined Gateway Discovery Protocol (GDP) for detecting routers. It is also capable of wiretapping RIP and IGRP routing updates, and inferring from those updates the location of routers. The protocol translator client implementation of router discovery does not actually examine or store the full routing tables sent by routers, it merely keeps track of which systems are sending such data. Each method is described in the following sections.

Using Proxy ARP

Proxy ARP, defined in RFC 1027, enables an Ethernet host with no knowledge of routing to communicate with hosts on other networks or subnets. Such a host assumes that all hosts are on the same local Ethernet, and that it can use ARP to determine their hardware addresses.

Under proxy ARP, if a gateway receives an ARP Request for a host that is not on the same network as the ARP Request sender, the gateway evaluates whether it has the best route to that host. If the gateway does have the best route, the gateway sends an ARP Reply packet giving its own Ethernet hardware address. The host that sent the ARP Request then sends its packets to the gateway, which forwards them to the intended host.

All Cisco Systems routers and gateway servers support proxy ARP.

Using the Default Gateway

Because the protocol translator hosts do not *wiretap* or participate in routing transactions, they are independent of any routing protocol. The protocol translator can use a default gateway (router) or rely on proxy ARP to send packets to hosts not on the local subnet or network.

You set up a default gateway for the protocol translator using the **ip default-gateway** global configuration command. The command has this syntax:

ip default-gateway *address*
no ip default-gateway

The **ip default-gateway** command specifies the router the protocol translator uses to send packets to hosts not on the local network or subnet. The argument *address* is the Internet address of the router.

The protocol translator sends any packets needing the assistance of a gateway to the address you specify. If another gateway has a better route to the requested host, the default gateway sends an ICMP *Redirect* message to the protocol translator. The ICMP *Redirect* message indicates which local router the protocol translator should use.

To display the address of the default gateway, use the EXEC command **show ip redirects**. See the section “Monitoring TCP/IP” later in this chapter for more information.

Example:

This example command defines the router on Internet address 192.31.7.18 as the default router:

```
ip default-gateway 192.31.7.18
```

Using the Router Discovery Mechanism

The Cisco Router Discovery mechanism allows the protocol translator to learn the routes to other networks using routing protocols. The mechanism supports these routing protocols:

- Gateway Discovery Protocol (GDP)
- Routing Information Protocol (RIP)
- Interior Gateway Routing Protocol (IGRP)

You may configure the three protocols together in any combination. When possible, it is recommended that GDP be used, as it allows each router to specify *both* a priority and the time after which a router should be assumed down if no further packets are received. Routers discovered using IGRP are assigned an arbitrary priority of 60. Routers discovered through RIP are assigned a priority of 50. For IGRP and RIP, the software attempts to measure the time between updates, and will assume that the router is down if no updates are received for 2.5 times that interval.

Each router discovered by the protocol translator becomes a candidate for the default router. The list of candidates is scanned to select a new router with the highest priority for the following reasons:

- To make sure that no higher-priority routers have been discovered. The router is polled at five minute intervals.
- When the current default router is declared down.
- When a TCP connection is about to time out because of excessive retransmissions, the server flushes the ARP cache and the ICMP redirect cache, and picks a new default router, in an attempt to find a route to the destination that works.

Detecting Routers Using the Cisco GDP Protocol

Use this interface subcommand to configure the Router Discovery feature using the Cisco GDP routing protocol:

ip gdp gdp
no ip gdp gdp

Use the **no ip gdp gdp** command to turn the feature off.

Example:

This example command configures Router Discovery using GDP on the Ethernet 0 interface:

```
!
interface ethernet 0
ip gdp gdp
!
```

Detecting Routers Using the UNIX RIP Protocol

Use this interface subcommand to configure the Router Discovery feature using the UNIX RIP routing protocol:

ip gdp rip
no ip gdp rip

Use the **no ip gdp rip** command to turn the feature off.

Example:

This example command configures Router Discovery using RIP on the Ethernet 1 interface:

```
interface ethernet 1
ip gdp rip
```

Detecting Routers Using the Cisco IGRP Protocol

Use this interface subcommand to configure the Router Discovery feature using the Cisco IGRP routing protocol:

ip gdp igrp
no ip gdp igrp

Use the **no ip gdp igrp** command to turn the feature off.

Example:

This example command configures Router Discovery using IGRP on the Ethernet 1 interface:

```
interface ethernet 1
ip gdp igrp
```

Configuring IP Access Lists

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

Specifying Access Conditions

To specify an access condition, use the **access-list** global configuration command. The command syntax is somewhat complex:

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 is predefined; it permits any address and is the default access list for terminal lines.

The protocol translator compares the address being tested to *address*, ignoring any bits specified in *wildcard-mask*. If you use the condition keyword **permit**, a match causes the address to be accepted. If you use the condition 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 nonzero address mask and thus potentially matches more than one actual address. The protocol translator examines first the actual addresses, then the wildcard addresses. The order of the wildcard addresses is important, because the protocol translator stops examining access list entries after it finds a match.

When both IP and LAT connections are allowed from a terminal line, and an IP access list is applied to that line with the **access-class** line subcommand, you also must create a LAT access list numbered the same if you want to allow any LAT connections from that terminal. This is because you can specify only one incoming and one outgoing access list number for each terminal line, and when checking LAT access lists, if the list specified does not exist, the system denies all LAT connections.

The **no access-list** command removes the specified access list.

Example 1:

The following access list example allows access for only those hosts on the three specified networks. This example 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
```

Example 2:

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 protocol translator would accept one address on subnet 48 and reject all others on that subnet. The protocol translator would accept addresses on all other network 36.0.0.0 subnets:

```
access-list 2 permit 36.48.0.3   0.0.0.0    ! allow one host
access-list 2 deny 36.48.0.0   0.0.255.255   ! block subnet 48
access-list 2 permit 36.0.0.0   0.255.255.255  ! all of net 36
```

Example 3:

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
```

Restricting Terminal Connections

To restrict incoming and outgoing connections between a particular terminal line or group of lines and the addresses in an access list, use the **access-class** line subcommand:

access-class *list* **{in|out}**

The argument *list* is an integer from 1 through 99 that specifies the defined access list. Use the keyword **in** to control which hosts may make Telnet or TCP connections into the protocol translator. These types of connections are associated with rotary groups, virtual terminals and printers, for example.

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 protocol translator:

```
access-list 12 permit 192.89.55.0   0.0.0.255
line 41 45
access-class 12 in
```

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

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 20:

```
access-list 10 permit 36.0.0.0   0.255.255.255
line 1 20
access-class 10 out
```

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

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

Configuring Extended Access Lists

Extended access lists allow finer granularity in control of connections allowed to or from a specific protocol translator port. For example, users may be restricted to only making connections to the telnet port, or incoming access to a port may be restricted to "privileged" ports on the original host.

To define an extended access list, use the extended version of the **access-list** subcommand, as follows:

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. All conditions must be met to make a match.

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 nonmatching 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 whether a connection is allowed in the first place.

Note: After an access list is initially created, any subsequent additions (possibly entered from the terminal), are placed at the *end* of the list. In other words, you cannot selectively add or remove access lists command lines from an access list.

Controlling Line Access

To restrict incoming and outgoing connections between a particular terminal line or group of lines (into a Cisco device) 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 199 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 terminal ports 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 can connect to any of them.

Example 2:

The following example defines an extended access list that permits only telnet and rlogin.

```
access-list 101 permit tcp 0.0.0.0 0.0.0.0 0.0.0.0 255.255.255.255 eq 23
access-list 101 permit tcp 0.0.0.0 0.0.0.0 0.0.0.0 255.255.255.255 eq
513
!(implicit deny of everything else)
! public terminals can only telnet and rlogin line 1 20 access-class
101 out
```

Extended access-lists also can be used with **slip access-class** *list* [**in**|**out**].

Maintaining TCP/IP

Use the privileged EXEC commands described in this section to maintain the IP caches and lines.

Clearing the ARP Cache

Use the **clear arp-cache** command to remove all noninterface entries from the Address Resolution Protocol (ARP) cache. Enter this command at the EXEC privileged-level prompt:

clear arp-cache

Clearing the Host-Name-and-Address Cache

Use the **clear host** command to remove one or all entries from the host-name-and-address cache. Enter this command at the EXEC privileged-level prompt:

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

To remove a particular entry, use the argument *name* to specify the host. To clear the entire cache, use an asterisk (*). The **show hosts** command displays the host-name-and-address cache; see the upcoming section “Displaying the Host-Name-and-Address Cache.”

Monitoring TCP/IP

This section describes the EXEC commands you use to obtain displays of IP activity.

Displaying Contents of the Access Lists

The **show access-lists** command displays the contents of the access-control lists. Enter this command at the EXEC prompt:

show access-lists

The following is sample command output:

```
Standard IP access list 1
  deny     192.31.7.18
  deny     192.31.7.50

  permit   0.0.0.0, wildcard bits 255.255.255.255
```

The permit and deny conditions are listed for each network, along with the wildcard mask, if any.

Displaying the Host-Name-and-Address Cache

Use the **show hosts** command to display information about name-and-address translation. Enter this command at the EXEC privileged-level prompt:

show hosts

The display includes the default domain name, the style of name lookup service, a list of name server addresses, and the cached list of host names and addresses. Table 1-4 briefly describes each field in this display.

The following is sample command output:

```
Default domain is CISCO.COM
Name/address lookup uses domain service
Name servers are 255.255.255.255
Host                    Flags      Age Type   Address(es)
MILANO.CISCO.COM        (temp, OK)  0   IP    131.108.4.20
MATHOM.CISCO.COM        (temp, ??)  0   IP    192.31.7.17
DUSTBIN.CISCO.COM       (temp, OK)  0   IP    131.108.1.27
URIAH.CISCO.COM         (temp, OK)  7   IP    131.108.4.4
CLASH.CISCO.COM         (temp, OK)  3   IP    192.31.7.24
NATASHA                 (perm, OK) 22   IP    192.31.7.27
```

Table 1-4 Name and Address Cache Display Field Descriptions

Field	Description
Host	Lists the host name.
Flags	Lists the lookup service.
temp	Entered by a name server; the protocol translator may remove the entry after 72 hours of inactivity.
perm	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.

Field	Description
Age	Indicates the number of hours since the protocol translator last referred to the cache entry.
Type	Indicates whether the host has an Internet or a PAD address.
Addresses	Host address. One host may have up to six addresses.

Displaying the Mapped Internet Address

The **show ip aliases** command displays Internet addresses mapped to TCP ports (*aliases*). Enter this command at the EXEC prompt:

show ip aliases

Sample output follows:

```
  IP Address    Port
192.31.7.52     TTY35
192.31.7.53     TTY36
192.31.7.54     TTY37
192.31.7.55     TTY40
```

The display lists the IP address and corresponding port number.

Displaying the IP ARP Cache

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

show ip arp

This command displays the contents of the IP ARP (Address Resolution Protocol) cache. An Address Resolution Protocol 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. Table 1-5 describes the fields displayed. Following is sample output.

```
Protocol  Address        Age (min)    Hardware Addr   Type     Interface
Internet  131.108.1.140        137    aa00.0400.6408  ARPA     Ethernet0
Internet  131.108.1.111        156    0800.2007.8866  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
```

Table 1-5 Show IP ARP Display Field Descriptions

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

Displaying IP Interface Statistics

The **show ip interfaces** command displays the Internet configuration of the network interface. Enter this command at the privileged EXEC prompt:

show ip interfaces

Following is a sample display:

```
Ethernet 0 is up, line protocol is up
  Internet address is 131.108.12.19, subnet mask is 255.255.255.0
  Broadcast address is 255.255.255.255
  Address determined by non-volatile memory
  Helper address is not set
  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 disabled
  Gateway Discovery is disabled
  IP accounting is disabled
```

This display includes IP-specific parameters of the network interface, including the address, the subnet mask, the broadcast address, and how the address was determined. The display also includes parameters specific to the host Cisco router.

Displaying IP Redirect Messages

The **show ip redirects** command displays the current cache of ICMP Redirect messages for the system. Enter this command at the privileged EXEC prompt:

show ip redirects

The protocol translator uses these redirect entries to decide which local gateway to use for the first hop when sending an Internet packet. The command displays the Internet address of the default gateway, if any. It also shows routers learned through the *Gateway Discovery Program* described earlier in this chapter in "Using the Router Discovery Mechanism."

A sample display follows:

```
Gateway           Using                Interval  Priority
131.108.2.140      IGRP                   80       60
131.108.2.77       IGRP                   80       60
131.108.2.85       GDP                    6       100
131.108.2.75       IGRP                   81       60
131.108.2.1        IGRP                   80       60
131.108.2.28       IGRP                   90       60

Default gateway is 131.108.2.85

Host                 Gateway     Last Use  Total Uses Interface
192.31.7.17       131.108.2.140  0:01      15         Ethernet0
131.108.1.33      131.108.2.140  0:00      33         Ethernet0
```

Displaying System Traffic

The **show ip traffic** command displays a detailed breakdown of the protocol traffic through the system. Enter this command at the privileged EXEC prompt:

show ip traffic

The following is sample command output:

```
IP statistics:
  Rcvd:  321777 total, 321777 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: 9315 received, 51 sent
  Sent:  326443 generated, 0 forwarded
         59 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, 0 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: 383 total, 0 checksum errors, 304 no port
  Sent: 232 total, 0 forwarded broadcasts
TCP statistics:
  Rcvd: 312108 total, 0 checksum errors, 7 no port
  Sent: 326229 total
ARP statistics:
  Rcvd: 574 requests, 149 replies, 86 reverse, 0 other
  Sent: 268 requests, 62 replies (0 proxy), 0 reverse
```

In the output, 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 protocol translator received no reply to an ARP request and therefore did not send the datagram. Most of the other output categories are interesting only on a router.

Displaying Connection Status

The **show tcp** command displays the status of all TCP connections. Enter this command at the privileged-level EXEC prompt:

show tcp [*line-number***]**

The optional argument *line-number* restricts the report to the TCP connections for a particular line.

Following is sample output:

```
tty2 (Richard x1234), connection 1 to host RICHARD-SS2
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 131.108.12.19, 27138   Foreign host: 131.108.2.154, 23

Enqueued packets for retransmit: 0, input: 0, saved: 0

Event Timers (current time is 187173320):
Timer:       Retrans   TimeWait    AckHold    SendWnd  KeepAlive
Starts:           64          0        197          0          0
Wakeups:           0          0        137          0          0
Next:              0          0          0          0          0

iss:  170606792  snduna:  170606894  sndnxt:  170606894    sndwnd:   4096
irs:  516160000  rcvnxt:  516200441  rcvwnd:      2144  delrcvwnd:    440

SRTT: 300 ms, RTTO: 607 ms, RTV: 3 ms, KRTT: 0 ms
minRTT: 0 ms, maxRTT: 1000 ms, ACK hold: 300 ms
Flags: higher precedence, retransmission timeout

Datagrams (max data segment is 1450 bytes):
Rcvd: 239 (out of order: 0), with data: 228, total data bytes: 40440
Sent: 236 (retransmit: 0), with data: 69, total data bytes: 105
```

Debugging TCP/IP

Use the privileged EXEC debugging commands described in this section to monitor error information about IP traffic. For each **debug** command, there is a corresponding **undebug** command to disable the reports.

debug arp

The **debug arp** command enables logging of ARP-related traffic.

debug broadcast

The **debug broadcast** command enables logging of all broadcast traffic.

debug ip-icmp

The **debug ip-icmp** command enables logging of ICMP transactions.

debug ip-packet

The **debug ip-packet** command enables logging of IP packets sent and received.

debug ip-routing

The command causes packets received and decisions made by the Router Discovery process to generate debugging messages. Use this EXEC command to debug the Router Discovery process.

debug ip-tcp

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

debug ip-tcp-packet *line*

The **debug ip-tcp-packet** command enables logging of each TCP packet associated with the specified line number.

debug ip-udp

The **debug ip-udp** command enables logging of UDP packets sent and received. This command is useful for checking whether SNMP packets are being sent or received.

debug telnet

The **debug telnet** command enables logging of incoming Telnet connections. To debug outgoing connections, append the **/debug** keyword to the EXEC command **connect.**

TCP/IP Global Configuration Command Summary

Following is an alphabetically arranged summary of the TCP/IP global configuration commands.

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

Specifies an access condition. The argument *list* is an integer from 1 through 99 selected to identify one or more permit/deny conditions. Access list zero is predefined and permits any address, and is the default access list for terminal lines.

The keyword **permit** specifies that a match causes the address to be accepted. The keyword **deny** specifies that a match causes the address to be rejected.

The software compares the address being tested to that entered for the *address* argument, ignoring any bits specified in *wildcard-mask*. 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. An access list can contain an indefinite number of actual and wildcard addresses. The **no** variation of the command removes the specified access list.

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

Installs a permanent entry in the system ARP cache, which the protocol translator uses to translate 32-bit Internet Protocol addresses into 48-bit Ethernet or IEEE 802.3 hardware addresses that use the specified type of encapsulation. The argument *internet-address* is the Internet address in dotted-decimal format corresponding to the hardware address specified by argument *hardware-address*, specified as a "dotted triple" of hexadecimal digits. The argument *type-keyword* is the encapsulation keyword, and can be one of the following:

- **arpa**—Specifies standard Ethernet style ARP (RFC 826) encapsulation. This is the default encapsulation method for Ethernet interfaces.
- **snap**—Specifies ARP packets conforming to RFC 1042.
- **probe**—Specifies the HP-proprietary Probe protocol for IEEE-802.3 networks and resolves IEEE 802.3 or Ethernet local data link virtual addresses using the *Virtual Address Request and Reply* message.

The **no** keyword with the Internet address removes the specified entry from the ARP cache.

[no] ip alias *internet-address TCP-port*

Assigns an Internet address to the service provided on a TCP port. The argument *internet-address* is the Internet address for the service, and the argument *TCP-port* is the number of the TCP port.

The **no** keyword with the Internet address removes the specified address for the protocol translator.

[no] ip default-gateway *address*

Sets up a default gateway for the protocol translator and specifies the router the protocol translator uses to send packets to hosts not on the local network or subnet. The argument *address* is the Internet address of the router.

Use the **no** keyword to disable this function; an address is not specified.

[no] domain-list *name*

Defines a list of default domain names to complete unqualified host names. The argument *name* is the domain name; do not enter an initial period. Remove a name from the list using the **no** variation of the command with the domain name.

[no] ip domain-lookup

Enables or disables IP Domain Name System-based host-name-to-address translation. Enabled by default.

[no] ip domain-name *name*

Defines a default domain name the protocol translator uses to complete unqualified host names (names without a dotted domain name appended to them). The argument *name* is the domain name; do not include the initial period that separates an unqualified name from the domain name. Use the **no** keyword to disable use of the Domain Name System; a name argument is not required.

[no] ip host name [*TCP-port***]** *internet-address1* **[***internet-address2 . . . internet-address8***]**

Enters a static host-name-to-address mapping in the host cache. The argument *name i*s the host name, and the optional argument *TCP-port* is the TCP port number (in decimal). If the *argument TCP-port* is omitted, the Telnet port is assumed. You can specify up to eight Internet address arguments (the first address argument is required), separated by spaces. Remove an address map from the list using the **no** variation of the command with the host name. Remove a host cache entry using the **clear host** EXEC command.

[no] ip ipname-lookup

Specifies the IP IEN-116 Name Server host-name-to-address translation. Disabled by default.

[no] ip name-server *server-address* **[***server-address2 . . . server-address6***]**

Specifies one or more hosts that supply name information. The arguments *server-address* are the Internet addresses of up to six name servers. By default, the protocol translator uses the all-ones broadcast address (255.255.255.255).

ip tcp chunk-size *number*

Provides faster response to user interrupt characters. The argument *number* is the number of characters output before the interrupt executes. The suggested value of *number* is 80, which will typically abort output within a line or two of where the user types the interrupt character. Values of less than 50 are not recommended for reasons of efficiency.

TCP/IP Interface Subcommand Summary

Following is an alphabetically arranged summary of the TCP/IP interface subcommands.

ip address *address subnet-mask*

Sets the interface address and the subnet mask. The argument *address* is a Class A, Class B, or Class C Internet address for the interface. The argument *subnet-mask* is a mask of address bits that specifies the network portion of the address.

ip broadcast-address *address*

Changes the Internet broadcast address when the protocol translator has the nonvolatile memory option installed. The argument *address* specifies an Internet address for the protocol translator to use in all broadcasts. The default broadcast address is 255.255.255.255.

Note: The protocol translator recognizes most forms of the broadcast address, but generates only one type of address when broadcasting on its own behalf.

[no] ip gdp gdp

Configures the Router Discovery feature using the Cisco GDP routing protocol. Use the **no** keyword to turn off the feature.

[no] ip gdp igrp

Configures the Router Discovery feature using the Cisco IGRP routing protocol. Use the **no** keyword to turn off the feature.

[no] ip gdp rip

Configures the Router Discovery feature using the UNIX RIP routing protocol. Use the **no** keyword to turn off the feature.

TCP/IP Line Subcommand Summary

Following is an alphabetically arranged summary of the TCP/IP line subcommands.

access-class *list* **{in|out}**

Restricts incoming and outgoing connections between a particular terminal line or group of lines and the addresses in an access list. The argument *list* is an integer from 1 through 99 that specifies the defined access list. The keyword **in** controls which hosts may make Telnet or TCP connections into the protocol translator. The keyword **out** defines the access checks made on outgoing connections.

telnet break-on-ip

Causes the system to generate a hardware Break signal on the RS-232 line that is associated with a Telnet connection, when a Telnet Interrupt-Process (IP) command is received on that connection. A hardware Break command is generated when a Telnet Break command is received.

telnet refuse-negotiations

Suppresses negotiation of the Telnet Remote Echo and Suppress Go Ahead options. This subcommand causes Telnet to refuse to negotiate the options on incoming connections.

telnet sync-on-break

Causes the protocol translator to send a Telnet Synchronize signal when it receives a Telnet Break signal. The TCP Synchronize signal clears the data path, but will still interpret incoming commands.

telnet transparent

Causes the protocol translator to send a Return (CR) as a CR followed by a NULL instead of a CR followed by a Line Feed (LF). This subcommand is useful for coping with different interpretations of end-of-line handling in the Telnet protocol specification.