DECbrouter 90 Products

Configuration and Reference Volume 2

Order Number: EK–DECB2–CG. A01

First Edition, May 1993

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

5 Routing IP

6 The IP Routing Protocols

Index

Figures

Tables

About This Guide

This section discusses the objectives, audience, organization, and conventions of this guide. It also lists related Digital publications.

Audience

This publication is intended for system administrators who will configure and maintain a DECbrouter 90.

Organization

The following chapters are contained in this guide:

- **Chapter 1, Routing Apollo Domain**–describes how to configure the DECbrouter 90T implementation of the Apollo Domain routing protcol.
- **Chapter 2, Routing AppleTalk**–describes the routing process of the AppleTalk network protocol.
- **Chapter 3, Routing CHAOSnet**–describes the DECbrouter 90T implementation of the CHAOSnet routing protocol.
- **Chapter 4, Routing DECnet**–describes Digital's implementation of DECnet Phase IV for the DECbrouter 90T product line.
- **Chapter 5, Routing IP**–introduces the DECbrouter 90T implementation of the IP protocol for its line of routing protocols, and describes an in-depth view of configuration options, IP addressing and its various protocols, and examples of well-designed networks.
- **Chapter 6, The IP Routing Protocols**–describes routing protocol options for the Internet protocol (IP) suite.

Conventions

The following conventions are used in this guide:

Convention	Meaning
`system displays`	Terminal sessions and information the system displays are printed in monospace type.
boldface	Information you enter is in boldface type. Keywords are also in boldface type.
>	Nonprinting characters are shown in angle brackets.
[ ]	Defaults are in square brackets.
COMMANDS	Commands are in uppercase letters.
italics	Command variables, worksheet values, new terms and concepts, and titles of books and periodicals are in italics.
Ctrl/X	Indicates two keys that you must press simultaneously.
Note	Provides general information about the current topic.
Caution	Provides information to prevent damage to equipment.

Related Documentation

The most important companion to this guide is the set of Digital documentation that accompanies the product. This guide refers you to the appropriate manual for additional reference material that is beyond the scope of this guide. The following manuals are shipped with the DECbrouter 90:

- *DECbrouter 90 Products Getting Started*
- *DECbrouter 90 Products Configuration and Reference, Volume 1*
- *DECbrouter 90 Products Configuration and Reference, Volume 3*
- *DECbrouter 90T Installation and Operating Information*
- *DECbrouter 90 System Error Messages*
- *DECbrouter 90 Products Command Summary*

To order these publications or additional copies of this guide, contact your sales representative. Refer to the How to Order Additional Documentation page at the back of this document for addresses and phone numbers.

Service and Support

Call your local representative for service.

Obtaining Additional Information

This section describes how to obtain additional Digital publications and includes tips for obtaining books, standards, and other information about networks and data communications that might be helpful while using Digital products.

Ordering Additional Digital Publications

To order these publications or additional copies of the *DECbrouter 90 Products Configuration and Reference* guides, contact your local representative.

Obtaining Information

This section describes how to obtain RFCs and technical standards.

Obtaining RFCs

Information about the Internet suite of protocols is contained in documents called *Requests for Comments or RFCs*. These documents are maintained by Government Systems, Inc. (GSI). You can request copies by contacting GSI directly, or you can use the TCP/IP File Transfer Protocol (FTP) to obtain an electronic copy.

Contacting GSI

You can contact GSI through mail, by telephone, or through electronic mail:

Government Systems, Incorporated
Attn: Network Information Center
14200 Park Meadow Drive, Suite 200
Chantilly, Virginia 22021

1-800-365-3642
(703) 802-4535
(703) 802-8376 (FAX)

NIC@NIC.DDN.MIL

Network address: 192.112.36.5
Root domain server: 192.112.36.4

Obtaining an Electronic Copy

To obtain an electronic copy of an RFC through FTP, complete the following step

1. At your server prompt, use the **ftp** command to connect to address *nic.ddn.mil*:

   ```
   % ftp nic.ddn.mil
   ```

The following display appears, followed by a login prompt:

```
Connected to nic.ddn.mil.
220-*****Welcome to the Network Information Center*****
    *****Login with username "anonymous" and password "guest"
    *****You may change directories to the following:
      ddn-news          - DDN Management Bulletins
      domain            - Root Domain Zone Files
      ien               - Internet Engineering Notes
      iesg              - IETF Steering Group
      ietf              - Internet Engineering Task Force
                           internet-drafts   - Internet Drafts
      netinfo           - NIC Information Files
      netprog           - Guest Software (ex. whois.c)
      protocols         - TCP-IP & OSI Documents
      rfc               - RFC Repository
      scc               - DDN Security Bulletins
220 And more.
```

2. At the login prompt, enter the word **anonymous** as your login name:

   ```
   Name (nic.ddn.mil:cindy): anonymous
   ```

   The NIC responds with this message:

   ```
   331 Guest login ok, send "guest" as password.
   Password:
   ```

3. Enter the word **guest** at the Password: prompt. The following message and ftp> prompt appear:

   ```
   230 Guest login ok, access restrictions apply.
   ftp>
   ```

4. Use the **cd** command to change directories. The following example illustrates how to change the RFC directory and obtain RFC 1158:

   ```
   ftp> cd rfc
   250 CWD command successful.
   ftp> get rfc1158.txt
   ```

5. To exit the FTP facility, enter the **quit** command at the ftp> prompt.

Obtaining Technical Standards

Following are additional sources for technical standards:

- Omnicom, 1-800-OMNICOM
- Global Engineering Documents, 2805 McGraw Ave., Irvine, CA 92714 1-800-854-7179
- American National Standards Institute, 1430 Broadway, New York, NY 10018 (212) 642-4932 or (212) 302-1286

1
Routing Apollo Domain

This chapter describes how to configure the DECbrouter 90 implementation of the Apollo Domain routing protocol. Tasks and topics described in this chapter include:

- An overview of Apollo Domain
- How to enable Apollo Domain routing
- How to configure static routes, set update timers, and define access lists

Global and interface command summaries appear at the end of the chapter.

DECbrouter 90 Implementation of Apollo Domain

The Apollo Domain routing protocol is the native-mode networking protocol for Apollo workstations. The DECbrouter 90 routing software implementation supports packet forwarding and routing for the Apollo Domain network protocols on Ethernet and serial interfaces using HDLC or X.25 encapsulation. The following restrictions apply to the DECbrouter 90 implementation:

- An IP address must be set on all media that use the ARP protocol (Ethernet, for example). This is because Domain ARP uses the same Ethernet type value as IP ARP.
- The DECbrouter 90 implementation of the Apollo Domain routing support assumes that it can use ARP to locate workstations on the local cable. Following are the versions of the Apollo operating system that support Domain ARP (D-ARP):
  - — DN3000 and DN3010 nodes need Version 9.7.4.1. This version of the operating system is available on a patch (ask for patch 186) from local Apollo field offices.
  - — DN3500, 4000, and 4500 nodes need Version 9.7.5.1 which is available on patch tape 185.
  - — Version 9.7 (which provides ARP for DN5xx-T nodes) needs Version 9.7.4.b101. No patch is available for these machines; it is provided only on a DECnet tape.

Note

Version 10.0 does not provide ARP. You must migrate to Version 10.1 and later versions of the Apollo Domain operating system before successfully operating with the DECbrouter 90. The DECbrouter 90 does not support the RTCHK and LCNODE commands and D-ARP in Apollo's 802.5 implementation.

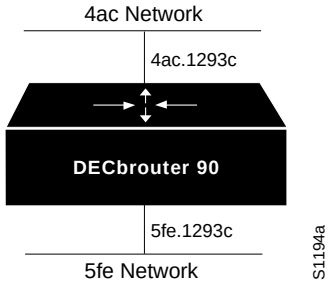
Apollo Domain Addresses

Apollo Domain addresses are 20-bit quantities, represented by five-digit hexadecimal numbers. Each host has a single address that is used for all of its network connections. An Apollo Domain host can have interfaces on more than one physical network (Ethernet, Domain Token Ring, serial line, and so on). Physical networks are identified by 32-bit numbers written in hexadecimal. These network numbers must be unique throughout an Apollo Domain internet. Because both the network number and the host address are needed to deliver traffic to a host, addresses usually are given as network numbers followed by host addresses separated with dots. An example would be as follows:

```
5fe.1293c
```

The number 5fe identifies the physical network, and the number 1293c identifies the host, as shown in Figure 1–1.

Figure 1–1 Apollo Domain Addresses



Configuring Apollo Domain Routing

There are only two commands required to enable Apollo Domain routing.

1. Use the global configuration command APOLLO ROUTING to enable routing.
2. Use the interface subcommand APOLLO NETWORK to assign Apollo routing to a specific interface.

All other configuration commands provide additional functionality or refinements. Each task is described in the following sections. These descriptions are followed by applicable EXEC commands for monitoring and debugging Apollo Domain networks. Summaries of global configuration commands and interface subcommands described here appear at the end of this chapter.

Enabling Apollo Domain Routing

To enable or disable Domain routing and specify which system-wide host address to use, use the APOLLO ROUTING global configuration command. The full syntax of this command follows.

apollo routing *address*
no apollo routing

The argument address is a unique, five-digit hexadecimal host address that you define, as described previously in the section Apollo Domain Addresses. The no

APOLLO ROUTING command disables Apollo routing. See the next section for an example of how to enable Apollo Domain routing.

Assigning the Apollo Domain Network Numbers

Apollo Domain network numbers must be assigned to the appropriate interfaces. This is done using the APOLLO NETWORK interface subcommand. The full syntax of this command follows.

apollo network number
no apollo network

The argument *number* is an eight-digit hexadecimal number. The NO APOLLO NETWORK command removes a network number from an interface.

Example

The following is a very simple example of setting up Apollo Domain routing on a router with two serial interfaces. The first step is to enable the RIP routing protocol and assign a Domain network address using the APOLLO ROUTING command. The next step is to assign network numbers to the two interfaces.

```
apollo routing 23d5a
interface serial 0
apollo network 5f
interface serial 1
apollo network 4e
```

Configuring Static Routes

Specify static routes for an Apollo Domain network with the APOLLO ROUTE global configuration command. Full syntax of this command follows.

apollo route *network network.address*
no apollo route *network network.address*

Use of this command causes packets received for the specified network to be forwarded to the specified router (whose address is *network.address*), whether or not that router is sending out dynamic routing. Use the NO APOLLO route command to remove the routes.

Example

If the router that handled traffic for network 33 had the address 45.91ac6, you would enter the following command:

```
apollo route 33 45.91ac6
```

Configuring Maximum Paths

To set the maximum number of multiple paths that the router will remember and use, use the APOLLO MAXIMUM-PATHS global configuration command. The command increases throughput by using multiple paths. It remembers higher-bandwidth routes in preference to lower-bandwidth routes. Full syntax of this command follows.

apollo maximum-paths *paths*
no apollo maximum-paths

The argument *paths* is the number of paths to be assigned. For a given destination, multiple paths of equal cost will be remembered. Output will be determined in round-robin fashion over these multiple paths at the packet level. The default value for *paths* is one; the NO APOLLO MAXIMUM-PATHS command restores this default.

Example

The following command sets three maximum paths:

```
apollo maximum-paths 3
```

The EXEC command show APOLLO ROUTE displays these additional routes and the maximum path value.

Setting Apollo Update Timers

To allow the Apollo Domain routing update timers to be set on a per-interface basis, use the APOLLO UPDATE-TIME INTERFACE subcommand.

apollo update-time *seconds*

Internal Apollo Domain routing timers are affected by the value set for the seconds argument, as follows:

- Apollo Domain routes are marked invalid if no routing updates are heard within six times the value of the update timer.
- Apollo Domain routes are removed from the routing table if no routing updates are heard within eight times the value of the update timer.
- The default value for the seconds argument is 30.
- The minimum value of the seconds argument is 10 seconds.
- The granularity of the update timer is determined by the lowest value defined.

Example

In the example below, the granularity of the update timer is 20, because that is the lowest value specified.

```
interface serial 0
apollo update-time 40
interface ethernet 0
apollo update-time 20
interface serial 1
apollo update-time 25
```

Note

Only use this command in an environment with DECbrouter 90s and Cisco Systems routers, and ensure that all timers are the same for all routers attached to the same network segment.

The EXEC command SHOW APOLLO INTERFACE displays the value of these timers.

Configuring Apollo Domain Access Lists

Apollo Domain access lists are collections of permit and deny conditions that apply to defined Apollo network and host numbers. The router sequentially tests the network and host numbers against conditions set in the access lists.

The first match determines whether the router accepts or rejects the network and host number. Because the router stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the software rejects the network and host number.

Specifying Apollo Domain Access Lists

Use the APOLLO ACCESS-LIST global configuration command to specify an access condition. The full syntax of this command follows.

apollo access-list *name* {**permit** | **deny**} [*firstnet*] *lastnet.host* [*wildcard-mask*]
no apollo access-list *name*

The argument *name* is a name defined by the network administrator for the access list. The NO APOLLO ACCESS-LIST command removes an access list; use only the name, not all the possible parameters, when you remove the list.

Choose the permit or deny condition for this list using the **permit** or **deny** keyword.

You can define Apollo access lists for one or a selected range of Apollo networks, which are defined by network number and host number separated by a dot. The optional argument *firstnet* and the argument *lastnet.host* specify a selected network range. Use the argument *lastnet.host* to specify just one network. Use -1 as a net number to specify all networks.

The optional *wildcard-mask* argument is a wildcard mask that uses the one bits to ignore the host part of the network address. Host 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. A wildcard address has a nonzero mask and thus potentially matches more than one actual address. The software examines the actual addresses, then the wildcard addresses. The order of the wildcard addresses is important because the software stops examining access list entries once it finds a match.

Protocol types and/or socket numbers are not useful in Apollo access lists. Also, note that Apollo access lists are named, not numbered as they are with other protocols.

Use the NO APOLLO ACCESS-LIST command to delete the entire access list.

Example

In the example that follows, the first line denies access to networks 3a through 3f, the second line denies access to a specific host, and the third line permits everyone else.

```
apollo access-list eng deny 3a-3f.0 fffff
apollo access-list eng deny 5fe.1293c
apollo access-list eng permit -1.0 ffff
```

Defining Access Groups

Use the APOLLO ACCESS-GROUP INTERFACE subcommand to specify the interface on which the access list is defined. Full syntax of this command follows.

apollo access-group *name*
no apollo access-group *name*

Enter the user-defined name for the access list defined by the APOLLO ACCESS-LIST global configuration command for the argument *name*.

Upon receiving and routing a packet to a controlled interface, the software checks the source network and host number of the packet against that set in the access list. If the access list permits the address, the software transmits the packet.

You can specify ranges of network numbers, along with host masks. While the masks may not be useful, they permit the host part to be ignored entirely.

Use the NO APOLLO ACCESS-GROUP command to remove the name.

Example

In this example, the access list named eng is assigned to the Ethernet interface.

```
interface ethernet 0
apollo access-group eng
```

Monitoring the Apollo Domain Network

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

Displaying Apollo Interface Parameters

Use the EXEC command SHOW APOLLO INTERFACE to display Apollo Domain parameters that have been configured on the interfaces. Enter this command at the EXEC prompt:

show apollo interface [*interface unit*]

You can specify the optional interface and unit arguments to see information for just that interface.

Following is sample output, specifying the Ethernet interface:

```
Ethernet 0 is up, line protocol is up
  Apollo address is 123A.CAFE
  Update time is 30 seconds
  Outgoing access list is not set
```

Displaying Apollo Routes

Use the EXEC command SHOW APOLLO ROUTE to display the Apollo Domain routing table. Enter this command at the EXEC prompt:

show apollo route

Following is sample output:

```
Codes: R - RIP derived, C - connected, S - static, 1 learned routes

Maximum allowed path(s) are/is 1
C Net 123A is directly connected, 0 uses, Ethernet0
C Net 123B is directly connected, 0 uses, Serial1
R Net 123C [1/0] via 123A.CAFB,  4 sec, 0 uses, Ethernet0
```

In the display, the leading character R indicates routes learned via RIP, C indicates connected entries, and S indicates statically defined entries.

Displaying Apollo Traffic Statistics

Use the EXEC command SHOW APOLLO TRAFFIC to display information on the number and type of Apollo Domain packets transmitted and received. Enter this command at the EXEC prompt:

show apollo traffic

Following is sample output. Table 1–1 describes the fields.

```
Rcvd:   8 total, 0 format errors, 0 checksum errors, 0 bad hop count,
        8 local destination, 0 multicast
Bcast:  8 received, 0 sent
Sent:   16 generated, 0 forwarded
        0 encapsulation failed, 0 no route
        0 unknown
```

Table 1–1 Show Apollo Traffic Field Descriptions

Field	Description
Format errors	Reported whenever a "bad packet" is detected (for example, corrupted header)
Checksum errors	Should not be reported; Apollo does not use a checksum
Bad hop count	Increments when a packets hop count exceeds 16
Encapsulation failed	Registered when the router is unable to encapsulate a packet
Unknown counter	Increments when packets are encountered that the router is unable to forward (for example, misconfigured helper-address or no route available)

Displaying the Apollo ARP Table

Use the EXEC command SHOW APOLLO ARP to display that portion of the ARP table that pertains to the Apollo Domain address resolution protocol. Enter this command at the EXEC prompt:

show apollo arp

Sample output follows.

```
Protocol  Address          Age (min)     Hardware Addr
Type      Interface
Apollo    123A.CAFE              -     0000.0c00.62e6
ARPA      Ethernet0
```

Debugging the Apollo Domain Network

Use the EXEC commands described in this section to troubleshoot and monitor the Apollo Domain network transactions. Generally, these commands are entered during troubleshooting sessions with Digital engineers. For each DEBUG command, there is a corresponding UNDEBUG command that turns the message logging off.

debug apollo-packet

The command DEBUG APOLLO-PACKET outputs information about packets received, transmitted, and forwarded.

debug apollo-routing

The command DEBUG APOLLO-ROUTING prints out information on Apollo Domain routing packets.

Apollo Domain Global Configuration Command Summary

The following is an alphabetical list of the Apollo Domain global configuration commands, which specify system-wide parameters for Apollo Domain support.

[**no**] **apollo access-list** *name* {**permit | deny**} [*firstnet*] *lastnet.host* [*wildcard-mask*]

Specifies Apollo Domain access condition. The argument *name* is a name defined by the network administrator for the access list.

Choose the permit or deny condition for this list using the **permit** or **deny** keyword. The optional argument *firstnet* and the argument *lastnet.host* specify a selected network range. Use the argument *lastnet.host* to specify just one network. The optional *wildcard-mask* argument is a wildcard mask that uses the one bits to ignore the host part of the network address. Host bits corresponding to wildcard mask bits set to zero are used in comparisons. The **no** version of the command deletes the access list.

[**no**] **apollo maximum-paths** *paths*

Sets the maximum number of multiple paths that the router will remember and use. The argument *paths* is the number of paths to be assigned. The default value is one, which is restored with the **no** form of the command.

[**no**] **apollo route** *network network.address*

Specifies static routes for an Apollo Domain network. Packets received for the specified network will be forwarded to the specified router, whether or not that router is sending out dynamic routing. The **no** version of the command removes the routes.

[**no**] **apollo routing** *address*

Enables or disables Domain routing and specifies which system-wide host address to use. The argument *address* is a unique, five-digit hexadecimal host address.

Apollo Domain Interface Subcommand Summary

The following Apollo Domain interface subcommands specify line-specific parameters for Apollo Domain support. These subcommands must be preceded by an INTERFACE command.

[**no**] **apollo access-group** *name*

Specifies the interface on which an Apollo Domain access list is defined. Enter the user-defined *name* for the access list defined by the APOLLO ACCESS-LIST global configuration command for the argument *name*. The **no** version of the command removes the name.

[**no**] **apollo network** *number*

Assigns Apollo Domain network numbers to the appropriate interfaces. The argument *number* is an eight-digit hexadecimal number. The **no** version of the command removes a network number.

apollo update-time *seconds*

Sets the Apollo Domain routing update timers. The argument *seconds* specifies the interval between updates.

2
Routing AppleTalk

This chapter describes the routing process of the AppleTalk network protocol. The topics and tasks described in this chapter include the following:

- An overview of the AppleTalk routing protocol.
- The DECbrouter 90 implementation of AppleTalk on both extended (also known as Phase II) and nonextended (Phase I) interfaces.
- Configuring AppleTalk routing.
- Configuring AppleTalk access list filters.
- Monitoring and debugging an AppleTalk network.

For more detailed information about the AppleTalk network systems, refer to Appendix F, "References and Recommended Reading."

DECbrouter 90 Implementation of AppleTalk

AppleTalk was designed as a client-server, or distributed, network system. In other words, users share network resources, such as files and printers, with other users. Interactions with servers are essentially transparent to the user, because the computer itself determines the location of the requested material and accesses it without requesting information from the user.

AppleTalk identifies several network entities, of which the most elemental is a *node*. A node is simply any device connected to an AppleTalk network. The most common nodes are Macintosh computers and laser printers, but many other types of computers also are capable of AppleTalk communication, including IBM PCs, Digital VAX/VMS systems and a variety of workstations. A router is considered a node on each connected network. To avoid confusion, these router nodes are referred to as *ports*. The DECbrouter 90 supports only one port per physical interface. The terms port and interface are used interchangeably in this document's discussion of AppleTalk routing. The next entity defined by AppleTalk is a *network*. An AppleTalk network is simply a single logical cable. Finally, an AppleTalk *zone* is a logical group of one or more (possibly noncontiguous) networks. These AppleTalk entities are shown in Figure 2–1.

Apple Computer has produced a variety of internetworking products with which to connect AppleTalk local area networks. Apple supports Ethernet, Token Ring, FDDITalk, and its own proprietary twisted-pair media access system (called LocalTalk). However, to allow an AppleTalk network full participation in a multiprotocol internetwork requires a multiprotocol router.

The DECbrouter 90 supports the AppleTalk network protocol (both extended and nonextended) over Ethernet, synchronous serial, and X.25 interfaces.

Figure 2–1 AppleTalk Entities

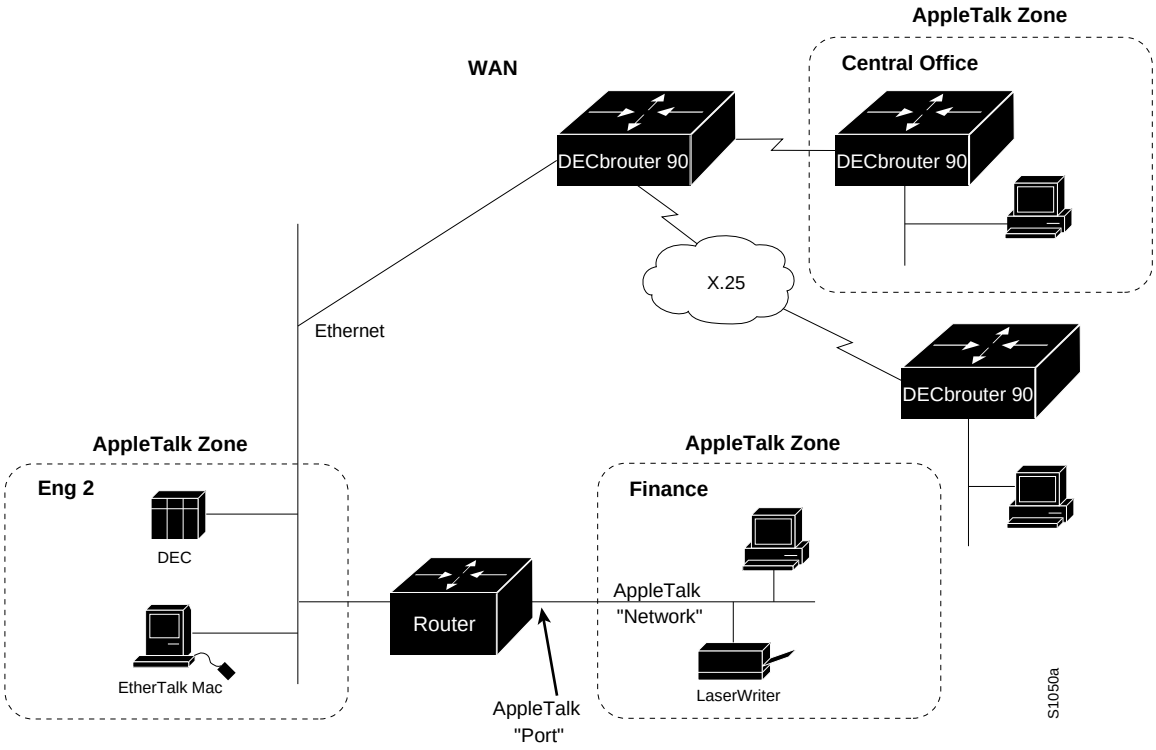
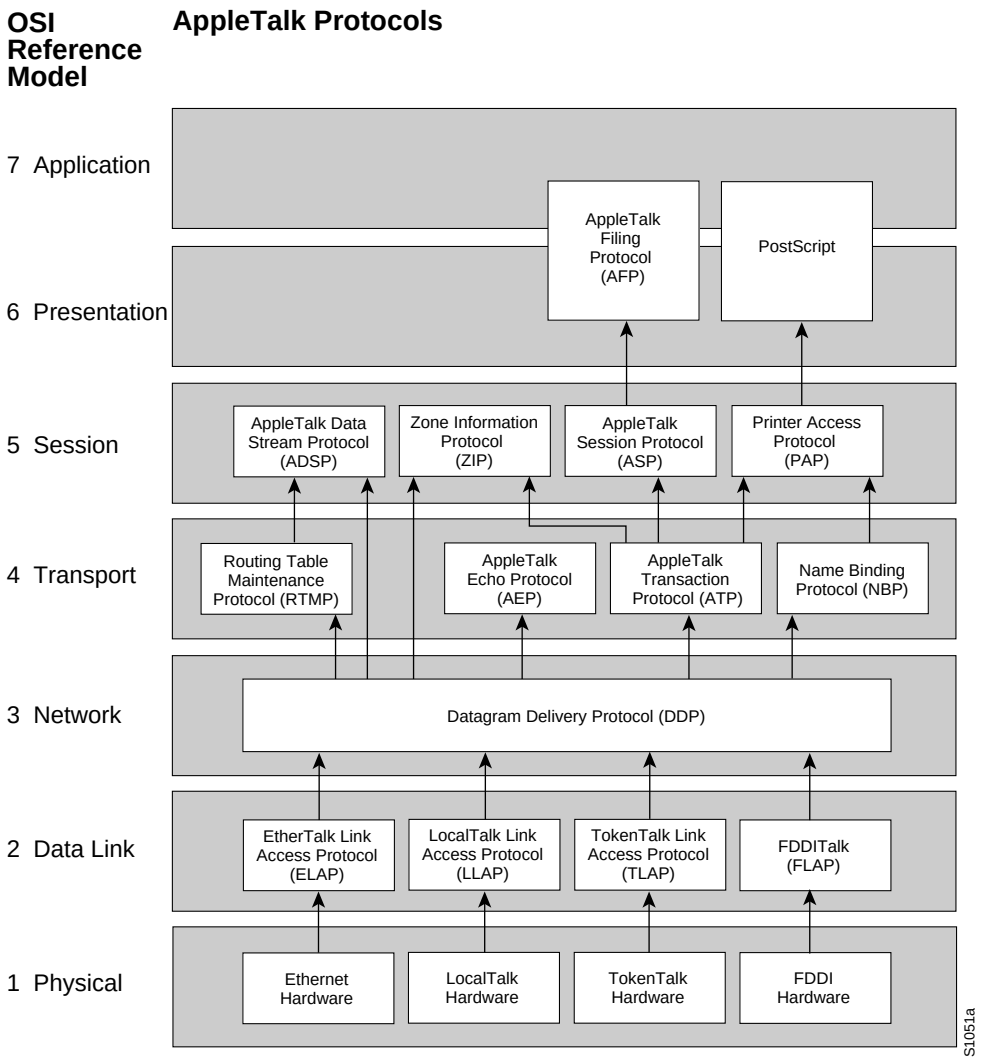


Figure 2–2 compares the AppleTalk protocols with the standard seven-layer OSI model and illustrates how AppleTalk works with a variety of physical and link access mechanisms.

The DECbrouter 90 AppleTalk implementation provides the following standard services, in addition to the ability to route any AppleTalk packet:

- AppleTalk Address Resolution Protocol (AARP)
- Datagram Delivery Protocol (DDP)
- Routing Table Maintenance Protocol (RTMP)
- Name Binding Protocol (NBP)
- AppleTalk Echo Protocol (AEP)
- AppleTalk Transaction Protocol (ATP)
- Zone Information Protocol (ZIP)

Figure 2–2 AppleTalk and the OSI Reference Model



The DECbrouter 90 AppleTalk implementation also includes the following enhancements:

- Support for EtherTalk 1.2 and EtherTalk 2.0 without the requirement for translation or transition routers
- Support for serial protocols, including SMDS, Frame Relay, X.25 and HDLC
- Configurable protocol constants
- No software limits on the number of zones or routes supported
- MacTCP support via the MacIP server
- NBP proxy service providing compatibility between the two AppleTalk standards
- IP encapsulation of AppleTalk, IPTalk, and Columbia AppleTalk Package (CAP) support
- Access control support to allow filtering of zones, routing data, and packets

- Integrated node name support to simplify AppleTalk management
- Interactive access to AEP and NBP provided via the ping router command
- Support for both configured (aka seed) and discovered port configuration
- Responder support used by Inter•Poll and other network monitoring packages
- SNMP over AppleTalk support

The DDP, RTMP, and AARP protocols provide end-to-end connectivity between internetworked nodes. NBP maps network names to AppleTalk internet addresses. NBP relies on ZIP to help determine which networks belong to which zones. File and print access is provided through AFP and PAP respectively, which work in concert with applications such as AppleShare and print servers.

Note

Apple Computer uses the name *AppleTalk* to refer the Apple Networking Architecture, whereas the actual transmission media used in AppleTalk Network are referred to as LocalTalk (Apple Computer's proprietary twisted-pair transmission medium for AppleTalk), TokenTalk (AppleTalk over Token Ring), EtherTalk (AppleTalk over Ethernet), and FDDITalk (AppleTalk over Fiber Distributed Data Interface).

AppleTalk, like many network protocols, makes no provisions for network security. The design of the AppleTalk protocol architecture requires that security measures be executed at higher application levels. The DECbrouter 90 supports AppleTalk distribution lists, allowing control of routing updates on a per interface basis. It is a security feature similar to those provided for other protocols.

Extended (Phase II) Versus Nonextended (Phase I) AppleTalk

AppleTalk was designed for local work groups. With the installation of over 1.5 million Macintosh systems in the first five years of the product's life, Apple found that some large corporations were exceeding the design limits of AppleTalk. Apple's solution was to create extended AppleTalk. The extended AppleTalk architecture increases the number of nodes per AppleTalk internetwork to over 16 million and an unlimited number of zones per cable. Apple also enhanced AppleTalk's routing capabilities and reduced the amount of network traffic generated by AppleTalk routers.

The introduction of the extended AppleTalk architecture also introduces the concept of *nonextended* and *extended* networks. Nonextended AppleTalk networks are sometimes called "Phase I," and extended networks are called "Phase II." Nonextended networks refer to the nonextended AppleTalk Ethernet 1.0 networks (explicitly removed by Apple but still supported by the DECbrouter 90), and to the nonextended serial line-based networks, including those configured using X.25 and LocalTalk.

Extended networks refer to the extended AppleTalk-compliant networks configured on Ethernet (EtherTalk 2.0), FDDI, and Token Ring media. Samples of the AppleTalk nonextended and extended network configurations can be found in the section AppleTalk Configuration Examples. The AppleTalk extended-network architecture provides extensions compatible with nonextended AppleTalk internetworks. The AppleTalk extended architecture was designed to remove the previous limits of 254 concurrently active AppleTalk nodes per cable, as well as

the previous limit of one AppleTalk zone name per cable. Extended AppleTalk contains better algorithms for choosing the best routers for traffic and is designed to minimize the amount of broadcast traffic generated for routing updates.

Another important feature in extended AppleTalk is the ability of a single AppleTalk cable to be assigned more than one network number. The size of the range of network numbers assigned to a cable determines the maximum number of concurrently active AppleTalk devices that can be supported on that cable, which is 254 devices per network number.

The DECbrouter 90 supports both extended and nonextended AppleTalk. Ethernet and serial interfaces can be configured for either extended or nonextended AppleTalk operation.

Note

Until every router in your internet supports AppleTalk Phase 2 (ATp2), you must observe the compatibility rules described in the Configuration Guidelines (Compatibility Rules). Not all end nodes must be upgraded to use the features provided by the AppleTalk enhancements.

Nonextended AppleTalk Addressing

AppleTalk addresses are 24 bits long. They consist of two components: a 16-bit network number and an 8-bit node number. The DECbrouter 90 AppleTalk software parses and displays these addresses as a sequence of two decimal numbers, first the network number, then the node number, separated by a dot. For example, node 45 on network 3 is written as 3.45. A node is any AppleTalk-speaking device attached to the network. Each enabled AppleTalk interface on a router is a node on its connected network.

AppleTalk Zones

When a router is used to join two or more AppleTalk networks into an internetwork, the component physical networks remain independent of each other. A network manager may assign to these network conceptual groupings known as *zones*.

There are two main reasons to create zones in an AppleTalk internetwork: to simplify the process of locating and selecting network devices, and to allow for the creation of departmental work groups that may exist on several different and possibly geographically separated networks.

For example, consider a large AppleTalk internetwork that may contain hundreds or thousands of shared resources and devices. Without a method of dividing this large number of resources and devices into smaller groups of devices, a user might have to scroll through hundreds or thousands of resource/device names in the Chooser to select the one resource to be used. By creating small, conceptual groups of resource and device names, users can now choose the resources they need much more quickly and easily than if they were sorting through a very long list of names.

A zone can include many networks that need not be physically co-located. A zone is not limited by geographical area. The partitioning afforded by zone names is conceptual, not physical.

Zones are defined by the network manager during router configuration. When a router is configured, each AppleTalk configured interface must be associated with exactly one zone name for nonextended networks, or one or more zone names for extended networks. Until a zone name has been assigned, AppleTalk routing features are disabled for that interface.

It is very important that routers explicitly configured with zone information be configured correctly.

Name Binding Protocol (NBP)

The name binding protocol (NBP) maps network entity names to internetwork addresses. It allows users to specify descriptive or symbolic names, while other software processes refer to numerical addresses for the same entities. With NBP, almost all user-level programs respond to names instead of numbers. When users select an AppleTalk device, they are using the NBP protocol to translate the device's entity name to the entity's network address. Numerical addresses dynamically assigned to nodes are primarily used by the router software and by network managers in the ping process (see the section The AppleTalk Ping Command.)

NBP provides four basic services for binding names to nodes and zones:

- Name registration
- Name deletion
- Name lookup
- Name confirmation

The nature of the AppleTalk addressing scheme is inherently volatile, and node addresses change frequently. Therefore, NBP associates numerical addresses with aliases that continue to reference the correct address if the address changes.

Zone Information Protocol (ZIP)

NBP uses the zone information protocol (ZIP) to determine which networks belong to which zones. The DECbrouter 90 uses ZIP to maintain the network-number-to-zone-name mapping of the AppleTalk internet.

Along with a routing table, each router maintains a data structure known as the *zone information table* (ZIT). The table provides a listing of network numbers for each network in every zone. Each entry is a *tuple* (an inseparable network number-hop number set) that matches a network number with a zone name as supplied by the network manager.

Dynamic Configuration (Discovery Mode)

AppleTalk provides for *dynamic configuration* . With dynamic configuration, not all fields of an AppleTalk address need to be specified in the configuration of a router. If there is another AppleTalk router on the network, it may be able to supply the network number and zone name. A preconfigured router on an AppleTalk network acts as a *seed router*, responding to configuration queries from other routers on its connected network.

Seed routers are routers that come up and verify the configuration with an operational router. If the configuration is valid, they start functioning. Seed routers come up even if no other routers are on the network. On the other hand, a *nonseed router* must first communicate with a seed router before it can commence operation. A nonseed router must obtain and verify the configuration

with another functioning router. The configuration of the nonseed router must match exactly with the configuration of the seed router for the nonseed router to function.

An end node always behaves in a manner similar to discovery mode. It uses any previous configuration as a starting point for initialization.

Unspecified parts of the AppleTalk address are entered as zero. Table 2–1 illustrates AppleTalk addresses that feature unspecified addressing.

Table 2–1 Examples of AppleTalk Addresses

AppleTalk Address	Description
34.5	Represents a fully qualified address (net 34, node 5)
0.5	A partially qualified address (net unspecified, node 5)
122.0	Represents net 122, node unspecified
0.0	Address is completely unspecified

Node numbers are automatically assigned by AppleTalk configured as zero. When the specified address is in use, the node randomly chooses its node number. The node will first try the node number that was its most recent address. If that number is unavailable, the node then searches for the next available address. If it reaches 254 without finding an available number, it cycles back to 1 and continues until it finds a free address. LocalTalk address restrictions are as follows: user node numbers are from 1 to 127, and server/printer node numbers are from 128 to 254. Nonextended Ethernet and extended media do not observe the server/user node distinction. The protocol reserves node numbers 0 and 255. Extended media also reserves the node number of 254.

For nonseed routers, an interface will behave as an AppleTalk end node. If zero has been specified for a network number, that interface will not route any packets until it receives its network number from a seed router.

Receipt of a routing table update informs the router of the network number for the interface on which the packet was received. Every routing table update includes the network number of the network the packet was sent on. Therefore, the router is able to determine the network number of the receiving interface.

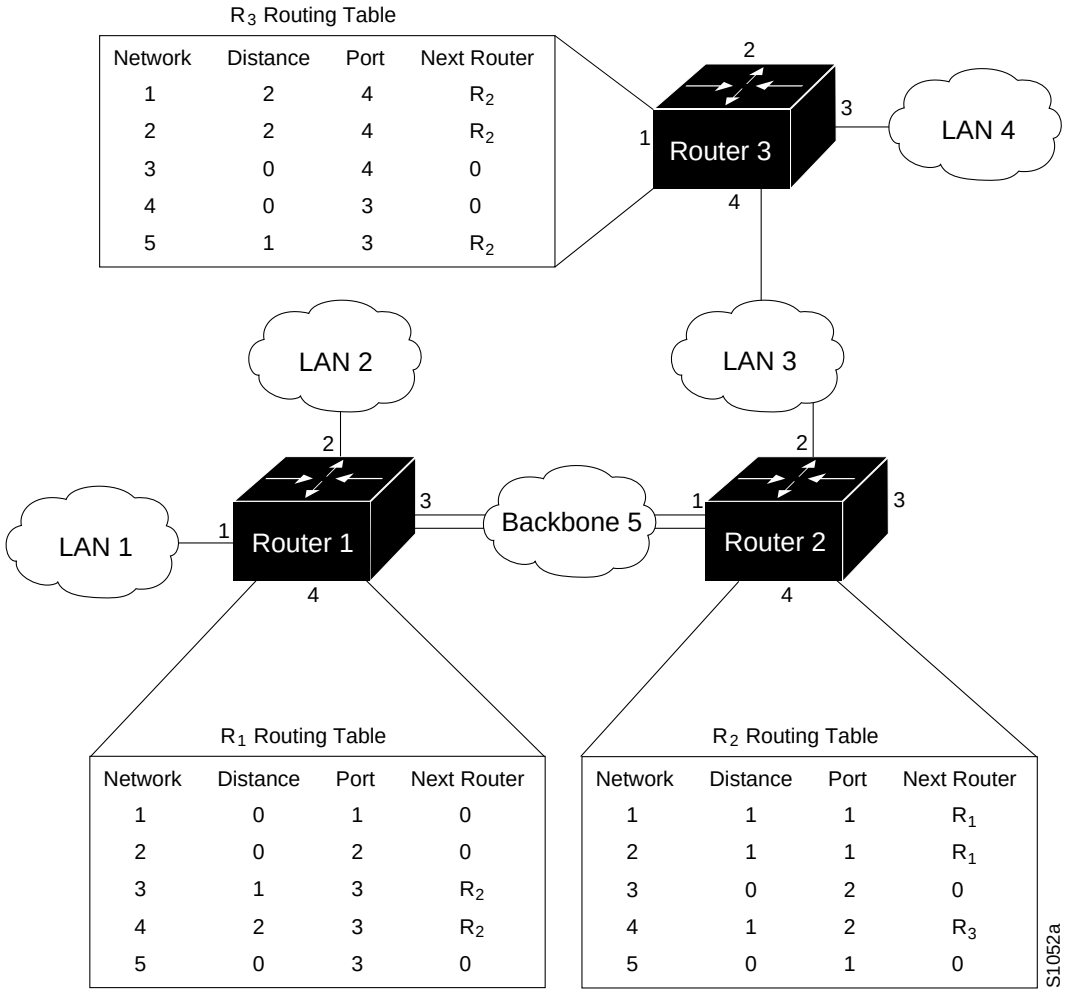
As long as one fully configured router exists on a physical network segment (or *cable*), other routers directly attached to that cable can use discovery mode to determine their configuration; they can take their information from an operational router. However, once the configuration process has stabilized for a particular AppleTalk internet, all routers thereafter should be configured as seed routers. Note that synchronous X.25 network interfaces must be explicitly configured on each router to be used as AppleTalk transports.

RTMP routing tables contain an entry for every network in the internet. Each entry includes the router port that leads to the destination network, the node ID of the next router to receive the packet, and the distance in hops to the destination network. Periodic exchange of routing tables allows the routers in an internet to ensure accurate and consistent information.

Node address information is maintained by tables appropriate to the media (usually AARP tables).

Figure 2–3 shows a sample RTMP table and the corresponding network topology.

Figure 2–3 Sample AppleTalk Routing Table



Extended AppleTalk Addressing

AppleTalk addresses, as explained in the section Nonextended AppleTalk Addressing, are composed of a 16-bit network and a n 8-bit node number. In nonextended AppleTalk, nodes within a single cable can communicate using only their 8-bit node numbers.

A node in extended AppleTalk is always identified by its network and node number. Dynamic address resolution when a router is not present includes the assignment of a random network number within a small range, as well as a node number. When a router is present in the network, a node starts up using its newly acquired address for a short period of time. It then immediately requests the range of valid network numbers from an operational router. The node then uses this to determine its actual AppleTalk address by selecting an unassigned address.

A new concept of cable *ranges* is introduced with the extended AppleTalk. Cables now have ranges of network numbers and multiple zones that can exist on them, so that a node can access anything that is in any of the zones that are on the same cable as the node itself. But the node can exist in only one zone and on only one network.

In an extended AppleTalk network, the mapping of a physical cable to a zone name is no longer valid. End nodes are expected to know the zone to which they belong or to choose from the list of available zones provided by a router. The router maintains a default zone that new nodes will use automatically if they have not chosen a zone previously.

AppleTalk Name Registration

A DECbrouter 90 with active AppleTalk interfaces register each interface separately. A unique interface name is generated by appending the interface type name and unit number to the router name. For example, if a router is named myrouter and has Appletalk enabled on Ethernet 0 in zone Engineering, the NBP registered name will be as follows:

```
myrouter.Ethernet0:DigitalRouter@Engineering
```

The NBP name is deregistered in the event that AppleTalk is disabled on an interface by configuration or due to interface errors.

Registering each interface on the router provides the AppleTalk site administrator with a positive indication that each router is properly configured and operating.

One name is registered per interface; other service types are registered once for every zone name on the router. The following display output from a SHOW APPLE NBP command illustrates this. This display shows that each interface is uniquely identified, but that only one SNMP Agent is generated per zone.

```
Net Adr Skt Name                  Type            Zone
4042   8 254 sloth.Ethernet0       DigitalRouter     Engineering
4042   8   8 sloth                 SNMP Agent      Engineering
4028   8 254 sloth.Serial1         DigitalRouter     Engineering
4035   8 254 sloth.Serial0         DigitalRouter     Engineering
4038   8   8 sloth                 SNMP Agent      Narrow Beam
```

AppleTalk Responder Support

The router answers Appletalk *responder* requests. The *listener* is installed on the Appletalk interface name registration socket.

The response packet generated supplies the bootstrap firmware version string, followed by the router operating software version string. These are displayed in the position of the Macintosh system version and the Macintosh printer driver version, respectively, in such applications as Apple's Inter•Poll™.

The response packet contains strings similar to those displayed by the SHOW VERSION EXEC command.

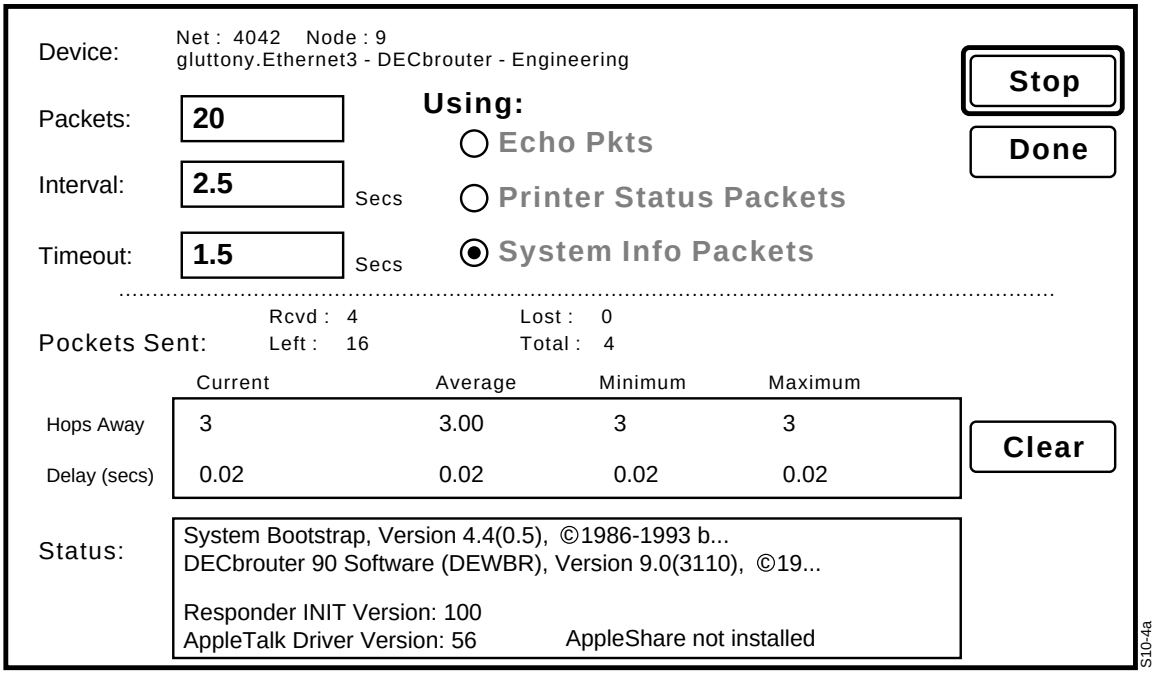
The information returned is as follows:

- The system bootstrap version (Flash ROM version).
- The currently running software version.
- The AppleTalk version—This always indicates 56, which is the first Apple Macintosh version that contained AppleTalk Phase 2 support.

- The AppleTalk responder version—This always displays 100, which indicates support of Version 1.0 responder packets.
- Finally, the router reports that AppleShare is not installed.

Figure 2–4 illustrates a typical output display for Inter•Poll that lists this information.

Figure 2–4 Sample illustration of Inter•Poll Output



Configuring AppleTalk Routing

This section provides an overview on how to configure Digital AppleTalk routing.

Configuration Overview

The AppleTalk interface configuration is different for the two types of AppleTalk interfaces: extended and nonextended.

Configuring a nonextended AppleTalk interface involves the following steps:

1. Enable AppleTalk routing with the APPLETALK ROUTING command.
2. Assign the nonextended AppleTalk addresses with the APPLETALK address interface subcommand.
3. Assign the zone name with the APPLETALK ZONE interface subcommand.

Configuring an extended AppleTalk interface involves these steps:

1. Enable AppleTalk routing with the APPLETALK ROUTING command.
2. Assign the extended AppleTalk cable range parameters with the APPLETALK CABLE-RANGE command.

3. Assign the zone name or names with the APPLETALK ZONE interface subcommand.

The software also provides commands for fine-tuning the AppleTalk network, for configuring packet filtering mechanisms, monitoring, maintaining and troubleshooting network operation. Alphabetically arranged summaries of the commands described in this chapter are also provided at the end of the chapter.

Configuration Guidelines (Compatibility Rules)

Follow these guidelines when configuring your AppleTalk network on a DECbrouter 90:

- If your AppleTalk internet contains any routers that support only nonextended AppleTalk, the following configuration restrictions must be observed. These restrictions are not enforced, but unpredictable behavior may result if they are violated. All routers in a network must support extended AppleTalk before these restrictions may be lifted.
  - — Cable ranges of only one (*666-666*, for example) are permitted.
  - — Each AppleTalk network may have only one zone associated with it.

Follow these guidelines when using the DECbrouter 90 with other vendors' AppleTalk implementations:

- A Macintosh that contains an Ethernet card must run EtherTalk Version 2.0 or later to support extended AppleTalk. A Macintosh with only a LocalTalk interface does not require any changes.
- Shiva FastPath routers must run K-Star Version 8.0 or later and be explicitly configured for extended AppleTalk.
- Apple's Internet Router software Version 2.0 supports a transition mode for translation between the nonextended AppleTalk and the extended AppleTalk on the same network. Transition mode requires the Apple upgrade utility and a special patch file from Apple.

A general understanding of the DECbrouter 90 representation of AppleTalk addresses is necessary before configuring your router. Refer to the sections DECbrouter 90 Implementation of AppleTalk, Nonextended AppleTalk Addressing, and Extended AppleTalk Addressing for more information.

Enabling AppleTalk Routing

Before you can configure AppleTalk routing, enable AppleTalk protocol processing. To do so, use the appletalk routing global configuration command. The full command syntax is as follows:

appletalk routing
no appletalk routing

The APPLETALK ROUTING configuration command enables AppleTalk protocol processing. The **no appletalk routing** disables all AppleTalk processing.

Assigning Nonextended (Phase I) AppleTalk Address

To assign AppleTalk addresses for nonextended networks, use the appletalk address interface subcommand. Its full syntax follows.

appletalk address address
no appletalk address

The argument *address* assigns AppleTalk addresses on the interfaces that will be used for the AppleTalk protocol. It assigns one AppleTalk address per interface. This step must be done before assigning zone names.

Note

Use this subcommand to configure nonextended interfaces.

The NO APPLETALK ADDRESS subcommand disables nonextended AppleTalk processing on the interface.

Example

These commands begin AppleTalk routing and assign address 1.129 to interface Ethernet 0.

```
!
appletalk routing
!
interface ethernet 0
appletalk address 1.129
!
```

Assigning a Cable Range for Extended AppleTalk (Phase II)

To assign the cable-range parameters, use the APPLETALK CABLE-RANGE interface subcommand. The full command syntax follows.

appletalk cable-range *start-end* [*network.node*]
no appletalk cable-range *start-end* [*network.node*]

This command designates an interface to be on an extended AppleTalk network. A cable range is the network numbers assigned to an extended network.

This range is specified using the argument *start-end*, which is a pair of decimal numbers between 1 and 65,279, inclusive. The starting network number should be less than or equal to the ending network number.

Specifying a cable range of 0-0 in the *start-end* argument (start = end = 0) places the interface into discovery mode, which attempts to determine cable range information from another router on that network.

The optional *network.node* argument specifies the suggested network and node number that will be used first when selecting the AppleTalk address for this interface. Note that any suggested network number must fall within the specified range of network numbers.

Use the NO APPLETALK CABLE-RANGE command to disable AppleTalk processing on the interface.

Example

This command assigns a cable range of 2-2 to the interface:

```
appletalk cable-range 2-2
```

Assigning a Zone Name

Use the APPLETALK ZONE interface subcommand to assign a zone name to an AppleTalk interface. Full command syntax for this command follows.

appletalk zone *zonename*
no appletalk zone [*zonename*]

Interfaces that are configured for seed routing or that have discovery mode disabled must have a zone name assigned before AppleTalk processing will begin.

The argument *zonename* specifies the name of the zone for the connected AppleTalk network. The argument *zonename* may include special characters from the Apple Macintosh character set. To include a special character, insert a colon and two uppercase hexadecimal characters. The hexadecimal equivalent for special characters in the Macintosh character set can be found in character tables published by Apple Computer (see Appendix D in the text *Inside AppleTalk*, 2nd edition).

Note

Due to restrictions associated with the DECbrouter 90 configuration parsing, it is not possible to define zone names with leading or trailing space characters. Although permitted by the standard, such names are not recommended because of the potential confusion that can be caused for users.

The APPLETALK ZONE command is used with both extended and nonextended configurations. Extended configurations can repeat this command to define a list of zones for the network.

The first zone specified in the list is the *default zone*. The router always uses the default zone when registering NBP names for interfaces. Computers in the network will select the zone in which they will operate from the list of zone names valid on the cable to which they are connected. If an interface is using nonextended AppleTalk, repeated execution of the zone command will replace the zone name for the interface with the newly specified zone name.

The NO APPLETALK ZONE interface subcommand deletes a zone name from a zone list or the entire zone list if none is specified. The optional zone name is ignored for nonextended AppleTalk interface configurations. The command also is ignored if the specified zone name is not in the current zone list for an interface. The list should be cleared using the NO APPLETALK ZONE interface subcommand before configuring a new zone list.

Note

The zone list is cleared automatically when APPLETALK ADDRESS or APPLETALK CABLE-RANGE commands are used. Additionally, the zone list is cleared if the APPLETALK ZONE command is used on an *existing* network; this can occur when adding zones to a set of routers until all routers are in agreement.

Examples

This command assigns the zone name Twilight to an interface:

```
appletalk zone Twilight
```

The following example shows use of the AppleTalk special characters sets by setting the zone name to Digital•zone.

```
appletalk zone Digital:A5zone
```

Setting and Resetting Discovery Mode

Discovery mode is set using the APPLETALK DISCOVERY interface subcommand. The full syntax of this command follows.

appletalk discovery
no appletalk discovery

This command resets the discovery mode and allows a new cable range to be discovered. If the port information has been discovered, and the port is operational, then this command results in the port being a valid seed port.

Use the **no appletalk discovery** command to return the software to the default (off) state.

Use the command NO APPLETALK DISCOVERY to allow the interface to be a seed port. If the interface is not operational when this command is issued, you must configure the zone names before the interface will be operational. Otherwise, the current zone list is retained as part of the configuration.

Using Discovery Mode

The DECbrouter 90 implementation of discovery mode is compliant with the mechanism defined by Apple. The network definition for a router using discovery mode is confirmed or modified to match the network configuration known by a seed router. The router in discovery mode then learns the associated zones from that router, and the port becomes operational. A seed router is required on each network.

While the port is operational, it acts like a seed router for any other routers that come on-line. However, another operational router port is still needed if the first port is restarted for any reason.

Note

It is not advisable to have all routers on a network configured with discovery mode enabled. If all routers restart simultaneously (for instance, after a power failure), the network is inaccessible until discovery mode is manually stopped via operator intervention.

Discovery mode is particularly useful while changing a network configuration or when adding a router to an existing network.

Configuring IP Encapsulation of AppleTalk Packets

Use the APPLETALK IPTALK interface subcommand to encapsulate AppleTalk in IP packets in a manner compatible with the CAP IPTalk and the Kinetics IPTalk (KIP) implementations.

appletalk iptalk *net.node zone*

This command enables IPTalk encapsulation on an interface that already has a configured IP address. The command allows AppleTalk communication with UNIX™ hosts running older versions of CAP that do not support native AppleTalk EtherTalk encapsulations. Typically, Apple Macintosh users wishing to communicate with these servers would have their connections routed through a Kinetics FastPath router running KIP (Kinetics IP) software.

This command is provided as a migration command; newer versions of CAP provide native AppleTalk EtherTalk encapsulations, and the IPTalk encapsulation is no longer required. The DECbrouter 90 implementation of IPTalk assumes that AppleTalk is already being routed on the backbone, because there is currently no LocalTalk hardware interface for the DECbrouter 90.

The DECbrouter 90 implementation of IPTalk does not support manually configured AppleTalk-to-IP address mapping (atab). The address mapping provided is the same as the Kinetics IPTalk implementation when the atab facility is not enabled. This address mapping functions as follows: The IP subnet mask used on the router Ethernet interface on which IPTalk is enabled is inverted (one's complement). This result is then masked against 255 (0xFF hexadecimal). This is then masked against the low-order 8 bits of the IP address to obtain the AppleTalk node number. The following example configuration should make this more clear:

```
interface Ethernet 0
ip address 131.108.1.118 255.255.255.0
appletalk address 20.129
appletalk zone Native AppleTalk
appletalk iptalk 30.0 UDPZone
```

In this configuration, the IP subnet mask would be inverted:

```
255.255.255.0 inverted yields: 0.0.0.255
```

Masked with 255 it yields 255, and masked with the low-order 8 bits of the interface IP address it yields 118.

This means that the AppleTalk address of the Ethernet 0 interface seen in the UDPZone zone is 30.118. This caveat should be noted, however: Should the host field of an IP subnet mask for an interface be more than 8-bits wide, it will be possible to obtain conflicting AppleTalk node numbers. For instance, consider a situation where the subnet mask for the Ethernet 0 interface above is 255.255.240.0, meaning that the host field is 12-bits wide.

Configuring IP Encapsulation DDP Socket to UDP Port Mapping

Use the global configuration subcommand APPLETALK IPTALK-BASEPORT to specify the UDP port number that is the beginning of the range of UDP ports used in mapping AppleTalk *well-known* DDP socket numbers to UDP ports. The command syntax looks like this:

appletalk iptalk-baseport *port-number*

Implementations of IPTalk prior to April 1988 mapped well-known DDP socket numbers to privileged UDP ports starting at port number 768. In April of 1988, the NIC assigned a range of UDP ports for the defined DDP well-known sockets starting at UDP port number 200 and assigned these ports the names *at-nbp*, *at-rtmp*, *at-echo*, and *at-zis*. The CAP, Release 6 and later, dynamically decides which port mapping to use. If there are no AppleTalk service entries in the */etc/services* file, CAP will use the older 768-based mapping.

This is the default UDP port mapping supported by the DECbrouter 90 implementation of IPTalk. If there are service entries in the */etc/services* file for the AppleTalk services, the router configured for IPTalk encapsulation should specify the beginning of the port mapping range with the APPLETALK IPTALK-BASEPORT command. The example configuration that follows builds upon the example for the APPLETALK IPTALK command to illustrate this concept.

```
appletalk iptalk-baseport 200
!
interface Ethernet 0
ip address 131.108.1.118 255.255.255.0
appletalk address 20.129
appletalk zone Native AppleTalk
appletalk iptalk 30.0 UDPZone
```

Checking Packet Routing Validity

Use the APPLETALK STRICT-RTMP global configuration command to enforce maximum checking of routing packets to ensure their validity. The full command syntax follows.

appletalk strict-rtmp
no appletalk strict-rtmp

The default of this command is to provide maximum checking.

Currently, strict RTMP checking consists of discarding RTMP packets arriving from routers that are not directly connected to the router performing the check. In other words, no routed RTMP packets will be accepted.

Use the NO APPLETALK STRICT-RTMP command to disable the maximum-checking mode.

Enabling and Disabling Routing Updates

Use the interface subcommand APPLETALK SEND-RTMP to allow the transmission of routing updates to be disabled for a specific interface. The full syntax of the command is as follows:

appletalk send-rtmp
no appletalk send-rtmp

This command allows a router to be placed on a network with AppleTalk routing enabled, without being seen by other AppleTalk routers on the cable. The default is to send routing updates. The NO APPLETALK SEND-RTMP command disables this default.

Changing Routing Timers

Note

You should not attempt to modify the routing timers without fully understanding the ramifications of doing so. Many other AppleTalk router vendors provide no facility for modifying their routing timers; should you adjust a DECbrouter 90's AppleTalk timers such that routing updates do not arrive at these other routers within the normal interval, it is possible to degrade or destroy AppleTalk network connectivity.

Use the global configuration command APPLETALK TIMERS to change the time intervals used in AppleTalk routing, as follows:

appletalk timers *update-interval valid-interval invalid-interval*
no appletalk timers *update-interval valid-interval invalid-interval*

The argument *update-interval* is the time, in seconds, between routing updates sent to other routers on the network. This is ten seconds by default. A route is considered suspect anytime it is older than the specified *update-interval*.

The argument *valid-interval* is the amount of time, in seconds, that the router will consider a route valid without having heard a routing update for that route. This is normally twice the update interval, 20 seconds by default. Once this period of time has elapsed without having heard a routing update for a route, the route becomes bad, and is now eligible for replacement by a path with a higher (less favorable) metric.

The argument *invalid-interval* is the amount of time, in seconds, that the route is retained after being marked as bad. During this period, routing updates include this route with a special *notify neighbor* metric. If this timer expires, the route is deleted from the routing table. By default, this timer is three times the valid interval, or 60 seconds.

Any of the timers can be specified as zero to specify the system default value.

Example

This command increases the update interval to 20 seconds, the route-valid interval to 40 seconds, and the route-invalid interval to 60 seconds.

```
appletalk timers 20 40 60
```

Assigning a Proxy Network Number

When an AppleTalk internetwork contains routers that support only nonextended AppleTalk and routers that support only extended AppleTalk, then one APPLETALK PROXY-NBP global configuration command is required for each zone in which there is a router that supports only nonextended AppleTalk. The full syntax of this command follows.

appletalk proxy-nbp *network-number zonename*
no appletalk proxy-nbp *network-number zonename*

The argument *network-number* must be a unique network number that will be advertised via this router as if it were a real network.

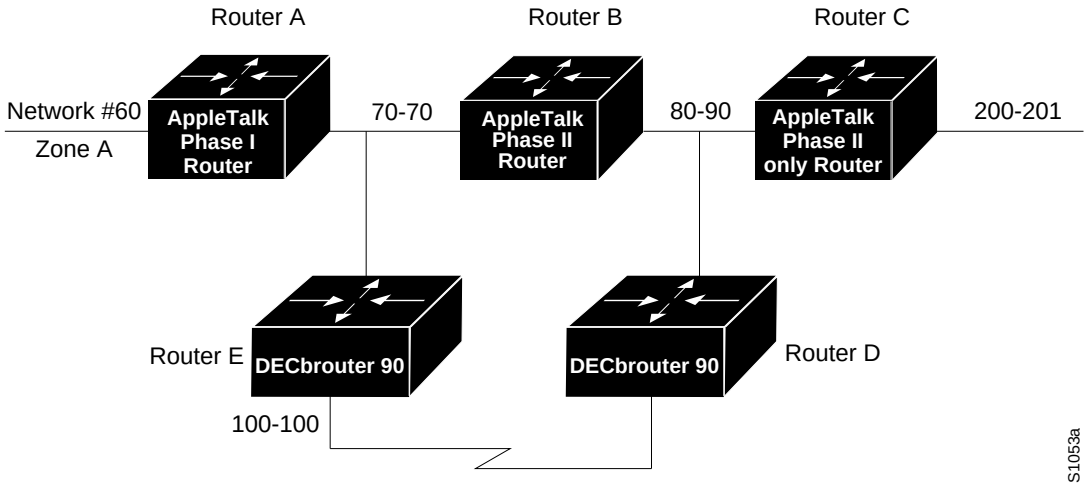
The argument *zonename* is the name of the zone requiring compatibility support.

No router can have the same network number defined as a proxy network, and no network number can be associated with a physical network.

Only one proxy is needed to support a zone, but additional proxies can be defined with different network numbers if redundancy is desired. Each proxy will generate one or more packets for each forward request it receives. All other packets sent to the proxy network are discarded. Redundant proxies increase the NBP traffic linearly.

Assume your network topology looks like the one in Figure 2–5. Also assume that router A supports only nonextended AppleTalk, that router B supports only extended AppleTalk (not in transition mode), and that router C supports only extended AppleTalk.

Figure 2–5 Example Network Topology



If router C generates an NBP hookup request for zone A, router B will convert this request to a forward request and send it to router A. Since router A supports only nonextended AppleTalk, it does not handle the forward request and ignores it. Hence, the NBP lookup from router C fails.

To work around this problem without putting a transition router adjacent to the nonextended-only router (router A), you could configure router D with an NBP proxy.

If you configured router D with an NBP proxy as follows, any forward requests received for Zone A are converted into lookup requests, and therefore, the nonextended router for Net 60 can properly respond to NBP hookup requests generated beyond router C. The following example demonstrates the command needed to describe this configuration:

```
appletalk proxy 60 A
```

Generating Checksum Verification

Use the APPLETALK CHECKSUM global configuration command to enable the generation and verification of checksums for all AppleTalk packets. The command syntax follows:

appletalk checksum
no appletalk checksum

An incoming packet with a nonzero checksum will be verified against that checksum and discarded if in error. By default, checksum verification is enabled.

The DECbrouter 90 does not check checksum on routed packets, thereby eliminating the need to disable checksum to allow operation of some networking applications.

Use the NO APPLETALK CHECKSUM command to disable checksum verifications.

Specifying the Time Interval Between AppleTalk ARP Transmissions

Use the APPLETALK ARP INTERVAL global configuration command to specify the time interval between retransmission of ARP packets, as follows:

appletalk arp {**request** | **probe**} **interval** *milliseconds*
no appletalk arp

The argument *milliseconds* specifies the interval. The minimum value is 33 milliseconds. The defaults for the *milliseconds* argument depend on the **probe** and **request** keywords as follows:

- **probe**—*milliseconds* = 200
- **request**—*milliseconds* = 1000

The keywords **request** and **probe** have the following effects:

- The **interval** *millisecond* value specified with the **request** version of this command is used when AARP is attempting to determine the hardware address for a different node, so a packet can be delivered. These **interval** *millisecond* values can be changed as desired, although the defaults are optimal for most sites.

- The **interval** *millisecond* value specified with the **probe** version is used when obtaining the address of this router (router being configured). These **interval** *millisecond* values should not be changed from the defaults because they directly modify the dynamic node assignment algorithm.

Lengthening the interval between packets permits responses from certain devices that respond more slowly, such as printers and overloaded file servers, to be received.

All values take effect immediately and are global to the router. The current values are available using the SHOW APPLE GLOBAL EXEC command.

The command NO APPLETALK ARP or a *milliseconds* value of 0 resets the defaults.

Example

This command lengthens the AppleTalk ARP retry interval to 2000 milliseconds.

```
appletalk arp request interval 2000
```

Specifying the AARP Retransmission Count

Use the APPLETALK ARP RETRANSMIT-COUNT global configuration command to specify the number of retransmissions that will be done before abandoning address negotiations and using the selected address.

appletalk arp {**request** | **probe**} **retransmit-count** *count*
no appletalk arp

The argument *count* specifies the retransmission count. The minimum value that can be specified is 1. The defaults for the *count* argument depend on the **probe** and **request** keywords as follows:

- **probe**—count = 10
- **request**—count = 5

The keywords **request** and **probe** have the following effects:

- The **retransmit-count** *count* value specified with the *request* version of this command is used when AARP is attempting to determine the hardware address for a different node, so a packet can be delivered. These **retransmit-count** *count* values can be changed as desired although the defaults are optimal for most sites.
- The **retransmit-count** *count* value specified with the **probe** version is used when obtaining the address of this router (router being configured). These **retransmit-count** *count* values should not be changed from the defaults, because they directly modify the dynamic node assignment algorithm.

All values take effect immediately and are global to the router. The current values are available using the SHOW APPLE GLOBAL EXEC command.

The command NO APPLETALK ARP or a *count* value of 0 resets the defaults.

Example

This command specifies an AARP retransmit count of 10.

```
appletalk arp request retransmit-count 10
```

AppleTalk MacIP Routing and IP Address Management Service

The DECbrouter 90 allows routing of IP datagrams to IP clients using DDP as a low-level encapsulation—commonly referred to as *MacIP*.

The MacIP address management and routing services available in the DECbrouter 90 are described in detail in Draft Internet RFC, *A Standard for the Transmission of Internet Packets over AppleTalk Networks*.

A case where MacIP services may be advantageous is in managing IP address allocations for a large, dynamic Macintosh population. There are several advantages to using the MacIP approach in this situation:

- Macintosh TCP/IP drivers can be configured in a completely standard way, regardless of location. Essentially, the dynamic properties of Appletalk address management become available for IP address allocation.
- All global parameters, such as IP subnet mask, Domain Name Services, and default routers, can be modified by the administrator in the DECbrouter 90. Macintosh IP users receive the updates by merely restarting their local TCP/IP driver.
- The administrator can monitor MacIP address allocations and packet statistics remotely by using the Telnet application to attach to the router console. This allows central administration of IP allocations in remote locations. For Internet sites, it allows remote technical assistance.

However, when evaluating the implementation of MacIP on a DECbrouter 90, there are several considerations to weigh:

- Each packet from a Macintosh client destined for an IP host, or from an IP host destined for the Macintosh client, *must* pass through the router if the client is using the router as a MacIP server. This increases traffic through the router in cases where the router is not a necessary hop. There is also a slight increase in router CPU use that is proportional to the number of packets delivered to and from active MacIP clients.
- Memory usage increases in the router, proportional to the total number of active MacIP clients (about 80 bytes per client).

Exceptions to Draft RFC

The DECbrouter 90 implementation of MacIP confirms to the September 1991 Draft RFC for MacIP, with the following exceptions:

- Fragmentation of IP datagrams that exceed the DDP MTU and that are bound for DDP clients of MacIP, is not performed.
- Routing to DDP clients outside of configured MacIP client ranges is not performed.

Configuring MacIP

Configuring MacIP support for the router involves the configuration of a few specific commands and requires several general configuration prerequisites. In general, MacIP-related configuration steps are as follows:

1. Establish a MacIP server for a specific zone.
2. Specify at least one *dynamic* or *static* resource address assignment statement for each server.

These are the only steps necessary. However, in order for MacIP to function properly, several conditions must be met:

- AppleTalk routing must be enabled on at least one interface.
- IP routing must be enabled on at least one interface.
- The MacIP zone name configured must be associated with a configured or *seeded* zone name.
- Any IP address specified in configuring a given MacIP server using an appletalk macip configuration statement must be *aliasable* to a specific IP interface on the router. Since the router is acting as a proxy for MacIP clients, it is not acceptable to use an IP address to which the router ARP module is unable to respond.

Enabling MacIP Servers

Use the APPLETALK MACIP SERVER global configuration command to establish a new MacIP server. The command syntax is as follows:

appletalk macip server *ip-address* **zone** *server-zone*
no appletalk macip server *ip-address* **zone** *server-zone*

Only one MacIP server can be configured per AppleTalk zone. A server is not registered via NBP until at least one MacIP resource is configured.

The NO APPLETALK MACIP command shuts down all active MacIP services. If entered with the keyword *server*, a specific *ip-address* and a specific *server-zone*, the particular server statement (if one exists) will be shut down and eliminated from the configuration.

Example

The following example illustrates specification of a MacIP server on interface Ethernet 0 in AppleTalk zone Engineering. Also provided are some related IP and AppleTalk configuration commands.

```
!This global statement specifies server address and zone:
appletalk macip server 131.108.1.27 zone Engineering
!
!These statements assign the address and subnet mask for Ethernet0:
interface ethernet 0
ip address 131.108.1.27 255.255.255.0
!
!These statements specify the AppleTalk zone Engineering for Ethernet0:
appletalk routing
!
interface ethernet 0
appletalk cable-range 69-69 69.128
appletalk zone Engineering
```

Note

Multiple MacIP servers can be configured for a router, but only one can be assigned to a particular zone and only one IP interface is assigned to each MacIP server. In general, the IP address assigned with this command must be *aliasable* to an existing IP interface address. For implementation simplicity, it is suggested that the address specified in this command match an existing IP interface address.

Specifying Addresses for Dynamic MacIP Clients

Use the APPLETALK MACIP DYNAMIC global configuration command to allocate a single IP address or a range of IP addresses to be assigned to *dynamic* MacIP clients by the MacIP server serving zone *server-zone*. Dynamic clients are those who accept any IP address assignment within the dynamic range specified. The command syntax is as follows:

appletalk macip dynamic *ip-address* [*ip-address*] **zone** *server-zone*
no appletalk macip dynamic *ip-address* [*ip-address*] **zone** *server-zone*

The NO APPLETALK MACIP command disables all MacIP services. If entered with the keyword **dynamic**, a specific *ip-address* range, and a specific *server-zone*, then the particular dynamic address assignment statement is eliminated from the configuration.

Example

The following example illustrates MacIP support for dynamically addressed MacIP clients with dynamically allocated IP addresses in the range of 131.108.1.28 to 131.108.1.44.

```
!This global statement specifies server address and zone:
appletalk macip server 131.108.1.27 zone Engineering
!
!This global statement specifies dynamically addressed clients:
appletalk macip dynamic 131.108.1.28 131.108.1.44 zone Engineering
!
!These statements assign the address and subnet mask for Ethernet0:
interface ethernet 0
ip address 131.108.1.27 255.255.255.0
!
!These statements specify the AppleTalk zone Engineering for Ethernet0:
appletalk routing
!
interface ethernet 0
appletalk cable-range 69-69 69.128
appletalk zone Engineering
```

Specifying Addresses for Static MacIP Clients

Use the APPLETALK MACIP STATIC global configuration command to define a range of addresses to be made available to MacIP clients who have reserved an invariant IP address. The server keeps track of these address for routing and informational purposes. The command syntax is as follows:

appletalk macip static *ip-address* [*ip-address*] **zone** *server-zone*
no appletalk macip static *ip-address* [*ip-address*] **zone** *server-zone*

The NO APPLETALK MACIP command shuts down all running MacIP services. If entered with the keyword *static*, a specific *ip-address*, and a specific *server-zone*, the particular static address assignment statement (if one exists) will be eliminated from the configuration.

Note

In general, it is recommended that you do not use fragmented address ranges if possible in configuring ranges for MacIP. However, in some cases it might be unavoidable. In such cases, use the APPLETALK MACIP STATIC command to assign a specific address or address range.

Example

The following example illustrates MacIP support for MacIP clients with statically allocated IP addresses. Addresses range from 131.108.1.50 to 131.108.1.66. Nodes have the specific addresses 131.108.1.81, 131.108.1.92 and 131.108.1.101.

```
!This global statement specifies server address and zone:
appletalk macip server 131.108.1.27 zone Engineering
!
!These global statements specify statically addressed clients:
appletalk macip static 131.108.1.50 131.108.1.66 zone Engineering
appletalk macip static 131.108.1.81 zone Engineering
appletalk macip static 131.108.1.92 zone Engineering
appletalk macip static 131.108.1.101 zone Engineering
!
!These statements assign the address and subnet mask for Ethernet0:
interface ethernet 0
ip address 131.108.1.27 255.255.255.0
!
!These statements specify the AppleTalk zone Engineering for Ethernet0:
appletalk routing
!
interface ethernet 0
appletalk cable-range 69-69 69.128
appletalk zone Engineering
```

MacIP Configuration and Address Assignment Considerations

Remember the following configuration when setting up MacIP routing:

- Static and dynamic resource statements are cumulative. The administrator can specify as many as necessary. It is desirable to specify a single all-inclusive range if possible, as opposed to several adjacent ranges; that is, 131.108.121.1 to 131.108.121.10 as opposed to 131.108.121.1 to 131.108.121.5 and 131.108.121.6 to 131.108.121.10.
- Overlapping resource ranges (that is, 131.108.121.1 to 131.108.121.5 and 131.108.121.5 to 131.108.121.10) are *not* allowed. If it is necessary to change a range in a running server, use the negative form of the resource address assignment statement (such as no appletalk macip dynamic) to delete the original range, followed by the corrected range statement.
- It is always possible to add resources to a running server as long as the new range does not overlap with one of the old ranges.

AppleTalk Access and Distribution Lists

The DECbrouter 90 AppleTalk access lists provide network security by permitting or denying certain packets access to a network interface. The DECbrouter 90 AppleTalk access lists are applicable to *zones* or *networks*; they cannot be used for specific nodes.

An *access list* is a list of AppleTalk network numbers or zones kept by the DECbrouter 90 to control access to or from specific networks or zones for a number of services.

AppleTalk access list capability supports four basic access control filter applications. Each uses AppleTalk access lists and can be defined on a per-interface basis. The supported filters are as follows:

- Packet filtering (zones are ignored)
- Routing data generation
- Routing data acceptance (zones are ignored)
- Get-zone-list handling (networks are ignored)

The subsequent discussions focus on the following topics:

- Alternative access list implementation approaches
- Command descriptions and definitions
- Configuration examples illustrating use of AppleTalk access lists

AppleTalk Access Control Methods

The DECbrouter 90 supports two general classes of AppleTalk *access control lists* (ACLs):

- True AppleTalk style ACLs (based on AppleTalk *zones*)
- IP style access lists (supported with prior software releases and based on *network number*)

The chief advantage of AppleTalk style ACLs is that they allow you to define access regardless of the existing network topology or any changes in future topologies—ostensibly because they are based on zones. A *zone* ACL is effectively a dynamic list of network numbers. The user specifies a zone name. The effect is as if the user had specified all of the network numbers belonging to that zone.

Zone-Based AppleTalk Access Control

AppleTalk style ACLs regulate the internetwork using zone names. Since zone names are the only network-level abstraction that users can access, this is the ideal control point. The DECbrouter 90 permits access and routing to be controlled using zone names—stated either explicitly or using generalized argument keywords. AppleTalk ACLs thus allow for simplified network management and greater flexibility in terms of adding segments with a reduced reconfiguration requirement for the router.

Note

Network entries and zone entries can be combined in a single list and have a cumulative effect. Network filtering is performed first, then zone filtering is applied to the result. However, for optimal performance, access lists should not include both zones and numeric network entities.

Network Number-Based AppleTalk Access Control

In general, specification of IP style ACLs to control network access is not recommended. However, it can be done by creating access lists based on network number. Such controls are not optimal, because they ignore the logical mapping provided by AppleTalk zones. Because partially obscuring a zone is not a defined facility, the list of network numbers must be configured in each secured router. When networks are added to a zone, those networks must be enumerated at each secure router.

Add to this administrative overhead the fact that anyone can add network segments (for example, finance gets a LaserWriter and installs a Cayman Gatorbox—thereby creating a new network segment), and the potential for confusion and misconfiguration is significant.

Nonetheless, the DECbrouter 90 does allow you create IP style ACLs. This feature may useful in permitting definition of network lists that control the disposition of networks that overlap, are contained by, or exactly match, a specific network range.

Note

One class of problem addressed by the use of network-number based access lists involves the potential assignment of conflicting (same) network numbers to different networks. An access control list can be used to restrict the network numbers and zones that a department can advertise, thereby limiting advertisement to an authorized set of networks. In general, zone-based ACLs are not enough in this application.

Creating AppleTalk Access Lists

To use access lists, two sets of commands are needed. The first set defines an access list. The second defines how the access list is to be used. In other words, these commands associate an access list with a specific interface or specify that it is to be used as a routing filter. To define an access list, use the ACCESS-LIST global configuration command. This command has several optional formats and supports *extended* AppleTalk addressing, as follows:

access-list *list* {**permit** | **deny network**} *network*
no access-list *list* {**permit** | **deny network**} *network*

access-list *list* {**permit** | **deny**} **cable-range** *start-end*
no access-list *list* {**permit** | **deny**} **cable-range** *start-end*

access-list *list* {**permit** | **deny**} **includes** *start-end*
no access-list *list* {**permit** | **deny**} **includes** *start-end*

access-list *list* {**permit** | **deny**} **within** *start-end*
no access-list *list* {**permit** | **deny**} **within** *start-end*

access-list *list* {**permit** | **deny**} **zone** *zonename*
no access-list *list* {**permit** | **deny**} **zone** *zonename*

no access-list *list*

access-list *list* {**permit** | **deny**} **additional-zones**
access-list *list* {**permit** | **deny**} **other-access**

The argument *list* is an integer from 600 to 699.

The argument *network* is an AppleTalk network number.

The argument *start-end* is a cable range value (decimal number from 1 to 65,279, inclusive). The starting network number should be less than or equal to the ending network number.

The argument *zonename* specifies the name of the zone for the connected AppleTalk network. It can include special characters from the Apple Macintosh character set. To include a special character, insert a colon and two uppercase hexadecimal characters.

Additional **permit** and **deny** conditions can be added to the list by issuing further ACCESS-LIST commands for that list.

Note

Unlike the access lists of other protocols, the ordering of conditions is unimportant. As a result, no network entry can overlap any other entry in a single list.

Use the no ACCESS-LIST command with the *list* number only to remove an entire access list from the configuration. Specify the optional arguments to remove a particular clause.

The following descriptions define the ACCESS-LIST command variations (specified previously) and outline the use and behavior of each:

access-list *list* {**permit** | **deny network**} *network*
no access-list *list* {**permit** | **deny network**} *network*

Specifies AppleTalk access control list (ACL) for a single network number. Affects matching nonextended networks. This rule is used when an exact match is made. Ranges of zero (in other words, same starting and ending number) do not match a network entry with that specific number.

access-list *list* {**permit** | **deny**} **cable-range** *start-end*
no access-list *list* {**permit** | **deny**} **cable-range** *start-end*

Specifies ACL for an extended network. The ACCESS-LIST command applies to extended networks with the matching starting and ending numbers. This rule is used when an exact match is to be made.

access-list *list* {**permit** | **deny**} **includes** *start-end*
no access-list *list* {**permit** | **deny**} **includes** *start-end*

Specifies ACL for any network, extended or nonextended, that overlaps any part of the range of values *start* through *end*, inclusive.

access-list *list* {**permit** | **deny**} **within** *start-end*
no access-list *list* {**permit** | **deny**} **within** *start-end*

Specifies ACL for any network, extended or nonextended, whose range of network numbers is included entirely within start through end, inclusive.

access-list *list* {**permit** | **deny**} **zone** *zonename*
no access-list *list* {**permit** | **deny**} **zone** *zonename*

Specifies ACL that applies to any network that has the specified *zonename* in its zone list.

access-list *list* {**permit** | **deny**} **additional-zones**

Specifies ACL used for zone-related checks to specify the default action for zones that were not enumerated. If not specified, the default is to deny additional zones. A **no** version is not applicable to this variation of the ACCESS-LIST command.

access-list *list* {**permit** | **deny**} **other-access**

ACL used as the default for any case that was not enumerated. If not specified, the default is to deny other access. A **no** version is not applicable to this variation of the access-list command.

Example

This example illustrates removal of all clauses associated with AppleTalk access list 610 and removal of the specific zone named Subhumans in AppleTalk access list 620.

```
no access-list 610
no access-list 620 zone Subhumans
```

Assigning an Access List to an Interface

A *packet filter*, specified through the APPLETALK ACCESS-GROUP interface subcommand, prevents any packets from being sent out an interface if the destination network has access denied. Once assigned, no packet that fails the APPLETALK ACCESS-LIST command will go out on that interface. The full syntax of this command follows.

appletalk access-group *access-list-number*
no appletalk access-group *access-list-number*

The argument *access-list-number* is the number of a predefined access list in the range of 600 to 699, inclusive. If an undefined access list is used, the rule defaults to **permit**. If a zone does not match any rule in the list, it is denied, unless permitted through the other-access option of the ACCESS-LIST global configuration command. Use the NO APPLETALK ACCESS-GROUP command to remove the list from the interface.

The EXEC command SHOW APPLE TRAFFIC displays the number of packets dropped because of access control. Refer to the section Monitoring the AppleTalk Network for more information. See the section Filtering Networks Received in Updates for an example of the use of this command.

When defining access lists for an interface, all networks within a zone should be governed by the same access control.

Filtering Networks Received in Updates

Use the APPLETALK DISTRIBUTE-LIST IN interface subcommand to filter routing updates received from other routers over the specified interface. The full syntax for this command follows.

appletalk distribute-list *access-list-number* **in**
no appletalk distribute-list *access-list-number* **in**

The argument *access-list-number* is the number of an Appletalk access list defined by a set of ACCESS-LIST commands.

When AppleTalk routing updates are received on the specified interface, each network number and range in the update is checked against the access list. Only network numbers and ranges that are permitted by the access list are inserted into the router's Appletalk routing table.

Use the **no appletalk distribute-list** *access-list-number* IN command to remove this function.

Note

Incoming routing data is checked using the network entries of the ACL. When using an AppleTalk input distribution list, the assigned access control list should not contain any zone entries since the resulting behavior is undefined.

Example

These commands cause any mention of network 10 to be ignored in routing updates arriving via Ethernet 0.

```
access-list 601 deny network 10
access-list 601 permit other-access
!
interface ethernet 0
appletalk distribute-list 601 in
```

Filtering Networks Sent Out in Updates

Use the interface configuration subcommand APPLETALK DISTRIBUTE-LIST OUT to filter routing data generated from zones or networks. The full syntax of this command follows.

appletalk distribute-list *access-list-number* **out**
no appletalk distribute-list *access-list-number* **out**

The argument *access-list-number* is the number of an AppleTalk access list defined by a set of ACCESS-LIST commands. If an undefined access list is used, the rule defaults to permit. If a zone does not match any rule in the list, it is denied unless permitted through the other-access option of the ACCESS-LIST global configuration command.

A *distribution list* is a list of AppleTalk access list numbers kept by the router that controls whether the network numbers specified by the access list are processed during the reception or transmission of routing updates. A distribution list will not prevent packets destined for a specified network number from being accepted; it will only prevent the route to the specified network from appearing in neighboring routers' AppleTalk routing tables.

Note

After performing network filtering, each network is checked for possible omission due to zone filtering.

Use the NO APPLETALK DISTRIBUTE-LIST *access-list-number* OUT command to remove this function.

Example

These commands prevent routing updates sent on Ethernet 0 from mentioning any networks in zone Admin. The APPLETALK ACCESS-GROUP command prevents packets from being sent out the interface.

```
!
access-list 601 deny zone Admin
access-list 601 permit other-access
interface Ethernet 0
appletalk distribute-list 601 out
appletalk access-group 601
```

Defining Get-Zone-List Filters

Use the APPLETALK GETZONELIST-FILTER interface subcommand to modify zone-list replies. The syntax for this command is as follows:

appletalk getzonelist-filter *access-list-number*
no appletalk getzonelist-filter *access-list-number*

The argument *access-list-number* must be in the range of 600 to 699, inclusive. If an undefined access list is used, the rule defaults to **permit**. If a zone does not match any rule in the list, it is denied, unless permitted through the additional-zones option of the ACCESS-LIST global configuration command.

Note

Using a get-zone-list (GZL) filter is not a complete replacement for anonymous network numbers. In order to prevent users from seeing a zone, all routers must implement the GZL filter. If there are other than non-DECbrouter 90 routers on the network (except Cisco routers), the GZL filter will not have a consistent effect.

Use the **no appletalk getzonelist** ACCESS-LIST-NUMBER command to remove this function.

The Macintosh's Chooser uses a GZL request to find a list of zones from which the user can select services. Any router on the same network as the Macintosh can respond with a GZL reply. The GZL filter is used to cause the router to omit certain zones in its reply. Note that this filter only changes the list of zones presented to the user. It does not change the router's behavior in routing packets or in processing routing updates. You must use the other filters to achieve those goals.

All routers on a given network should filter get-zone replies identically. Otherwise Macintosh systems will present different zone lists to the user depending upon which router responds to the request. Inconsistent filters can result in zones appearing and disappearing every few seconds when the user remains in the Chooser. If there are other than DECbrouter 90 routers on the network, the command APPLETALK GETZONELIST-FILTER is not likely to be useful unless the routers other than Digital routers (for example, Cisco routers) have a similar feature.

Note

The reply to a get-zone list request is also filtered by any APPLETALK DISTRIBUTE-LIST OUT filter in effect for the interface involved. You only need to define an APPLETALK GETZONELIST-FILTER command if you want additional filtering to be applied to GZL replies. This filter is rarely needed except to eliminate zones that do not contain user services.

Permitting Partial Zones

If any network of a zone is denied, the zone also is denied unless the AppleTalk global configuration command APPLETALK PERMIT-PARTIAL-ZONES is enabled. The command syntax is as follows:

appletalk permit-partial-zones
no appletalk permit-partial-zones

The default is for APPLETALK PERMIT-PARTIAL-ZONES to be *disabled*.

Specifying the keyword sequence **permit-partial-zones** disables the default behavior where the complete zone is access controlled if any associated network is controlled. In other words, when a specific zone is partially obscured, other (visible) networks that are not subject to access control are propagated normally when **permit-partial-zones** is enabled.

The NO APPLETALK PERMIT-PARTIAL-ZONES version of this command disables this option and restores the default condition where a complete zone is controlled if any associated network is controlled.

If **permit-partial-zones** is enabled, AppleTalk cannot maintain consistency for the nodes in the affected zones, and the results are undefined. With this option enabled, an inconsistency is created for the zone, and several assumptions made by some Appletalk protocols are no longer valid.

Requiring Specific Route Zones

Use the APPLETALK REQUIRE-ROUTE-ZONES global configuration command to prevent bogus routes (possibly generated by a broken router or corrupt packet) from causing ZIP protocol storms. The command syntax is as follows:

appletalk require-route-zones
no appletalk require-route-zones

The default is for **require-route-zones** to be enabled.

ZIP protocol storms can arise when corrupt routes are propagated and routers broadcast ZIP requests to determine the network/zone associations.

When **require-route-zones** is enabled, the router will not advertise a route to its neighboring routers until it has obtained the network/zone associations. This effectively limits the storms to a single network rather than the entire internet.

Use the NO APPLETALK REQUIRE-ROUTE-ZONES command to disable the **require-route zones** option and set the condition such that the router can advertise routes to its neighbors without having obtained the network-zone associations.

Disabling this feature enables the routing behavior associated with prior software releases; when enabled, this option *requires* that all networks have zone names prior to advertisement to neighbors.

As an alternative to disabling this option, *empty* zones can be filtered from the list presented to users while the pertinent networks can be associated with a zone name for monitoring purposes. This is done with the APPLETALK GETZONELIST-FILTER interface subcommand described earlier in this chapter.

The *user* zone lists can be configured to vary from interface to interface, but this is discouraged since AppleTalk users expect to have the same user zone lists at any end node in the internet. This kind of filtering does not prevent explicit access via programmatic methods, but should be considered a user optimization whereby unused zones are suppressed. Other forms of AppleTalk access control lists should be used to actually *secure* a zone or network.

Controlling AppleTalk Names Displayed

Two global configuration commands control the router's name-display feature: APPLETALK LOOKUP-TYPE and APPLETALK NAME-LOOKUP-INTERVAL. The names and services specified with the APPLETALK LOOKUP-TYPE command are held in a lookup cache and displayed using the SHOW APPLETALK NAME-CACHE EXEC command.

Note

Node numbers do not change very frequently, because each device keeps track of the last node number it was assigned. Typically, node numbers only change if a device is shut down for an extended period of time or is moved to a new network segment.

Setting Service Types Cached

Use the APPLETALK LOOKUP-TYPE global configuration command to specify services listed in SHOW APPLETALK NBP and SHOW APPLETALK NAME-CACHE EXEC command display. The command syntax is as follows:

appletalk lookup-type *serviceType*
no appletalk lookup-type [*serviceType*]

The argument *serviceType* is the specific AppleTalk service.

- **DigitalRouter**—Listed in SHOW APPLETALK NBP display per port
- **SNMP Agent**—Listed in SHOW APPLETALK NBP display per zone if and only if Apple's SNMP-over-DDP is enabled

Note

If AppleTalk routing is enabled, enabling SNMP will automatically enable SNMP-over-DDP. Also, if you include this entry with your list of APPLETALK LOOKUP-TYPE commands, enter it *as is*—with the space between SNMP and Agent.

- **IPGATEWAY**—Active MacIP server names
- **IPADDRESS**—Active MacIP server addresses

Other common service types, each of which can be specified in an APPLETALK LOOKUP TYPE command, include:

- **AppleRouter**—Apple internet router
- **GatorBox**—Cayman's LocalTalk gateway

- **Workstation**—System 7 Macintosh (also, the machine type is defined as an additional name by defining both, so it is possible to easily identify all user nodes)
- **FastPath**—Shiva's LocalTalk gateway
- **systemRouter**—Digital's OEM router name
- **SNMP**—Identifies node supporting IP SNMP (only done by some vendors and now considered obsolete)

Note

These non-Digital services only appear in a display generated using the SHOW APPLETALK NAME-CACHE command. Also, if a neighboring router is not a Cisco, or DECbrouter 90 router, it is possible the router will be unable to determine the name of the neighbor. This is normal behavior and there is no workaround.

As many APPLETALK LOOKUP-TYPE commands may be issued as desired. The service type **DigitalRouter** is the only type initially enabled; it cannot be disabled. The DECbrouter 90 generates requests to the network segments to which it is directly connected, so the name cache does not contain entries for *all* the selected services in a zone—only those that are *directly connected*.

The interval can be set to be as frequent or infrequent as desired.

Entries are deleted after several interval periods expire without the entry being refreshed. At each interval, a single request is sent through interfaces that have valid addresses. Refer to the description of **appletalk name-lookup-interval** that immediately follows this command for a discussion of setting the name lookup interval.

The command NO APPLETALK LOOKUP-TYPE can be used with or without the *serviceType* argument. Using the argument specifies exclusion of a specific service type from the name cache. Prevent all names (except those relating to DECbrouter 90s) from being cached by using the **no** version of this command without the argument *serviceType*.

Note

When specifying a service-type name with this command, spaces are valid (such as **SNMP Agent**). However, do not use leading or trailing spaces when entering these names.

Example

The following simple example illustrates the use of this command to specify various services to be listed in the SHOW APPLETALK NAME-CACHE display.

```
appletalk lookup GatorBox
! In addition to DigitalRouter, check for GatorBox services
appletalk lookup AppleRoute
! and Apple Internet Routers
appletalk lookup IPGATEWAY
! and MacIP servers...
appletalk lookup Workstation
! Not generally needed for any but the most inquiring minds.
! Note DigitalRouter automatically is listed
```

Setting the Service Name Lookup Interval

Use the APPLETALK NAME-LOOKUP-INTERVAL global configuration command to set the interval between service pollings by the router on its AppleTalk interfaces. The command syntax is as follows:

appletalk name-lookup-interval *intInSeconds*
no appletalk name-lookup-interval

The argument *intInSeconds* is the interval in seconds between NBP lookup pollings. The router does not query the entire zone, but instead polls only the connected networks to reduce overhead.

An interval of 0 (zero), the default, disables the appletalk lookup-type feature. All polling for services is suspended. By reentering a nonzero, positive integer value for *intInSeconds*, the appletalk lookup-type specifications in the active configuration are reinstated. A value of 0 (zero) is equivalent to NO APPLETALK NAME-LOOKUP-INTERVAL. You cannot disable the lookup of *DigitalRouter*.

Note

After disabling this parameter, the name cache is purged at the next global configuration run.

There is no limit on this value, but the recommended values are 300 (5 minutes) and 1200 (20 minutes). The smaller the interval, the more packets are generated to handle the names.

Example

The following example illustrates setting the lookup interval:

```
appletalk name-lookup-interval 1200
! Lookup names once every 20 minutes
```

AppleTalk Configuration Examples

The following examples illustrate a variety of AppleTalk configurations:

- Configuring nonextended AppleTalk networks routing over X.25
- Creating a simple configuration for an extended AppleTalk network
- Setting up extended AppleTalk networks routing over HDLC
- Initializing SNMP configuration for AppleTalk
- Configuring IPTalk
- Building and using AppleTalk access lists

Nonextended AppleTalk Routing over X.25

The configuration of X.25 networks is similar to that for HDLC encapsulation. However, you must completely and explicitly configure all network and node numbers in an X.25 environment. Note that all AppleTalk nodes within an X.25 network must be configured with the same AppleTalk network number.

X.25 configuration for AppleTalk involves mapping AppleTalk addresses to X.121 addresses, executed with the X.25 configuration subcommand **x25** map.

Each time a packet is sent to a particular AppleTalk address, that address is looked up in the X.25 map table in order to match it to an X.25 address. The packet is encapsulated in X.25 frames and sent to the X.25 node which is its destination.

The receiving node reassembles the X.25 frames if necessary, then strips the packet of X.25 framing information so that the original AppleTalk datagram can be processed.

In the configuration commands that follow, the keyword **broadcast** (as used at the end of the X25 MAP commands) has the following effect: whenever a broadcast packet is sent, assuming the broadcast flag is set, then each X.121 address specified will receive the broadcast. The X.25 protocol does not provide broadcasts; therefore, they must be simulated in this manner when using X.25 as a transport protocol for another protocol that requires broadcasts, such as AppleTalk.

If the X.121 address of the router on the far end of the X.25 network is 123456789012, and your local X.121 address is 210987654321, and the two routers are at AppleTalk addresses 7.63 and 7.25, you would configure these systems in the following way:

```
!Configuration for First Router
interface serial 0
appletalk address 7.25
appletalk zone Twilight
x25 map appletalk 7.63 123456789012 broadcast
!
!Configuration for Second Router
interface serial 0
appletalk address 7.63
appletalk zone Twilight
x25 map appletalk 7.25 210987654321 broadcast
!
```

In this example, a third router has the X.121 address 333444555666 and AppleTalk address 7.100.

```
!Configuration for Third Router
interface serial 0
appletalk address 7.100
appletalk zone Twilight
x25 map appletalk 7.25 210987654321 broadcast
x25 map appletalk 7.63 123456789012 broadcast
!
```

With the addition of the third router, both the original routers need an additional x25 map entry:

```
x25 map appletalk 7.100 333444555666 broadcast
```

Note

X.25 can be configured only as a nonextended network using the APPLETALK ADDRESS command. Logically, it is the same as a LocalTalk network, because both are *always* nonextended networks.

Extended AppleTalk Routing Network

The following example illustrates how to configure an extended AppleTalk network.

This configuration defines the zones *Empty Guf* and *Underworld* from which the router and the nodes may choose to reside. The equal cable range numbers allow compatibility with nonextended AppleTalk networks.

```
!
appletalk routing
!
interface ethernet 0
appletalk cable-range 69-69 69.128
appletalk zone Empty Guf
appletalk zone Underworld
!
```

Extended AppleTalk Routing over HDLC

AppleTalk's dynamic address assignment feature allows users and network managers to choose default network addresses. The following example illustrates configuration of two ends of a serial line for routing of AppleTalk over HDLC. An example of the interface configuration for both ends of the serial line follows.

Example

The following commands enable AppleTalk routing for interface serial 1. Assuming that a serial link is made between two different routers (both using interface serial 1), then the configuration can be the same for both ends of the connection.

```
!
interface serial 1
appletalk cable-range 1544-1544
appletalk zone Twilight
!
```

Configuring SNMP in AppleTalk Networks

For AppleTalk to enable SNMP-over-DDP, AppleTalk routing must be active before the SNMP configuration, otherwise the AppleTalk SNMP server will not be started. This is done correctly with the standard configuration handling.

However, problems can arise if AppleTalk is started manually when the SNMP server was previously configured for the router. The following example configuration sequence illustrates proper activation of SNMP and AppleTalk on a router.

Specification of the **snmp-server** global configuration commands must *follow* the APPLETALK ROUTING global configuration commands and INTERFACE subcommand specifications.

Example SNMP Configuration for an AppleTalk Router

The following example briefly illustrates the command sequence needed when starting AppleTalk routing and an SNMP server process on a router from the console.

```
!
no snmp-server
!
appletalk routing
appletalk event-logging
!
interface Ethernet 0
ip address 131.108.29.291 255.255.255.0
appletalk cable-range 29-29 29.180
appletalk zone Zombie
!
snmp-server community propellerhead RW
snmp-server trap-authentication
snmp server 131.108.2.160 propellerhead
!
```

Configuring IPTalk

IPTalk is AppleTalk *encapsulated* in IP datagrams. IPTalk is used to route across backbones and to communicate to applications on hosts that are unable to communicate via AppleTalk. The CAP is an example. The following discussion describes setting up UNIX-based systems and the DECbrouter 90 to use CAP IPTalk and other IPTalk implementations.

Note

If your system is a Sun or Digital ULTRIX system, it may be possible to directly run CAP in a mode that supports EtherTalk. In that case, your system looks like any other AppleTalk node and does not need any special IPTalk support. However, other UNIX systems for which EtherTalk support is not available in CAP must run CAP in a mode that depends upon IPTalk.

IPTalk Configuration Steps

The procedure that follows outlines the basic steps for setting up the DECbrouter 90 and UNIX hosts for operation using IPTalk implementations.

Note

The procedure that follows does not give full instructions on how to install CAP on the UNIX system. This discussion specifically addresses the required steps for setting up the UNIX system's configuration file that defines addresses and other network information. Generally, this is the only file that relies on the DECbrouter 90 address and configuration information. The rest of the setup for UNIX systems involves building the CAP software and setting up the UNIX startup scripts that make it run. These peripheral discussions are beyond the scope of this manual.

1. Set up the DECbrouter 90 for AppleTalk. The routers talk to each other and to Apple products using more standard protocols, such as EtherTalk or TokenTalk. IPTalk is needed only on an interface that will communicate with a UNIX system. You must have AppleTalk routing enabled among all of the routers that are going to use IPTalk. This includes any routers in the middle that are required for them to be able to communicate with each other. Otherwise the UNIX systems cannot communicate with each other.
2. Ensure that IP is enabled on the interface to be used to communicate with the UNIX system. Since IPTalk is AppleTalk encapsulated in IP, IP must be enabled on the router *and* on the UNIX system. This interface must be on *the same subnet* as the UNIX system.
3. Allocate an AppleTalk network number for IPTalk. A separate AppleTalk network number is needed for each IP subnet that is to run IPTalk. It is possible to have a number of UNIX machines on the same subnet. They all use the same AppleTalk network number for IPTalk. They must have their own individual node ids. It is possible for the same router to have IPTalk enabled on several interfaces. Each interface must have a different AppleTalk network number assigned for use by IPTalk, since each interface will be using a different IP subnet.
4. Determine the CAP format of the AppleTalk network number. The CAP software is based on an old convention that expresses AppleTalk network numbers as two octets (numbers from 0 to 255) separated by a dot. The Apple convention uses decimal numbers from 1 to 65,279. Use the following formula to convert between the two:
   CAP format: x.y
   Apple format: d
   - Converting from Apple to CAP: x = d/256; y = d%256
     ("/" represents truncating integer division; and % the remainder)
   - Converting from CAP to Apple: d = x * 256 + y

     Example:
     Apple format: 14087; CAP format: 55.7
5. Decide on a zone name for IPTalk. There are no special constraints on choice of zone name. The same zone name can be used for several networks. IPTalk and normal networks can be combined in the same zone if desired.

6. Decide which UDP ports you are going to use for IPTalk. The default is to use ports beginning with 768. Thus, RTMP uses port 769, NBP port 770, and so on. These are the original ports hardcoded into older versions of CAP. The only problem with using them is that the port numbers are not officially assigned by the Internet's network information center (NIC). Thus other applications could use them, possibly causing conflicts—although this is unlikely. The NIC has assigned a set of UDP ports beginning with 200. Beginning with CAP release 5.0, it became possible to configure CAP to use the officially allocated ports. If you do so, RTMP will use port 201, NBP port 202, and so on. If you decide to use these or other ports, you must configure both CAP and the DECbrouter 90 to use the same ports.

7. Enable IPTalk on each interface of the router as required. Here is an example:

```
appletalk routing
!
interface ethernet 0
ip address 128.6.7.22 255.255.255.0
! EtherTalk phase 2
appletalk cable 1792-1792 1792.22
appletalk zone MIS-Development
! IPTalk
appletalk iptalk 14087.0 MIS-UNIX
```

In this example, Ethernet 0 is configured to speak AppleTalk in two different ways:

- Through EtherTalk phase 2 using network number 1792 and zone MIS-Development
- Through IPTalk using network number 14087 and zone MIS-UNIX

Note

The node id is not specified (is left as 0) in the APPLETALK IPTALK global configuration command. The IPTalk node id is chosen automatically, based on the IP address. It is normally the host number portion of the IP address. For example, with an IP address of 128.6.7.22 and a subnet mask of 255.255.255.0, the host number is 22. Thus, the IPTalk node id is 22. If the IP host number is larger than 255, the low-order 8 bits are used, although fewer than 8 bits may be available depending on the IP subnet mask. If the mask leaves fewer bits, the node number will be quietly truncated. Be sure to use a node address that is compatible with the subnet mask. In any event, there are likely to be problems using IPTalk with host numbers larger than 255.

If you choose to use the official UDP ports (those beginning with 200), use the following command configuration line in your configuration:

```
appletalk iptalk-baseport 200
```

Note

This line is not an interface command; it can go before or after the interface commands.

8. Configure each UNIX host with the correct network number, zone name, and router.
   As an example, here are the contents of */etc/atalk.local* from a UNIX system with IP address 128.6.7.26:

```
# IPTalk on net 128.6.7.0:
# mynet mynode myzone
55.7  26      MIS-UNIX
# bridgenet bridgenode bridgeIP
55.7  22      128.6.7.22
```

   The first noncomment line defines the address of the UNIX system; the second line defines the DECbrouter 90. In both cases, the first column is 55.7, which is the AppleTalk net number chosen for use by IPTalk (in CAP format). The second column is the AppleTalk node id, which must be the same as the IP host number. The third column is the zone name on the first line and the IP address of the DECbrouter 90 on the second line.

   Note that the following must agree:

   - The first column in both lines must agree with the AppleTalk network number used in the APPLETALK IPTALK configuration command. However in */etc/atalk.local* it must be in the CAP format, and in the configuration command it must be in the Apple format.
   - The second column in both lines must agree with the IP host address of the corresponding system (the UNIX machine for the first line, the DECbrouter 90 for the second line).
   - The third column in the first line must agree with the zone name used in the APPLETALK IPTALK configuration command.
   - The third column in the second line must agree with the IP address of the DECbrouter 90.

9. Make sure that your CAP software is using the same UDP port numbers as the DECbrouter 90. Currently, CAP's default is the same as Digital's (in other words, port numbers beginning with 768). If you want to use this default, you do not need to take any further action with regard to this step. However, to use the official UDP port numbers, make sure that you have used the following global configuration command (described previously):

```
appletalk iptalk-baseport 200
```

10. On the UNIX system, add the following lines to the file /etc/services:

```
at-rtmp       201/udp
at-nbp        202/udp
at-3          203/udp
at-echo       204/udp
at-5          205/udp
at-zis        206/udp
at-7          207/udp
at-8          208/udp
```

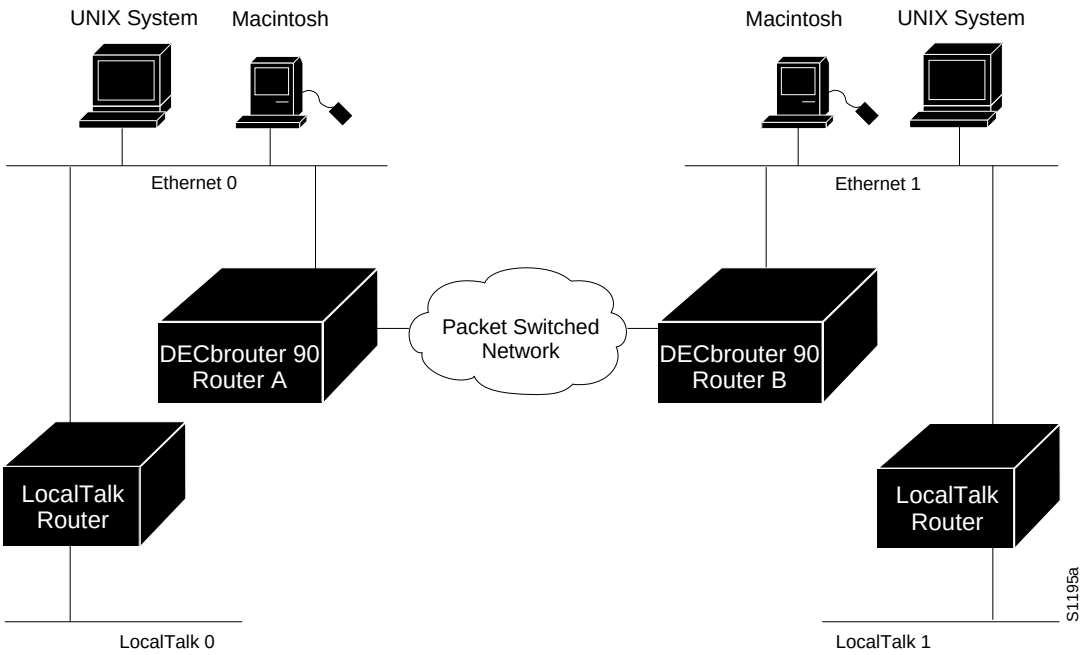
    If you are using network information services (NIS), also commonly referred to as *yellow pages*, remember to do a *make* in */var/yp* after changing */etc/services*. If you are using the default ports, for those starting with 768 you do not need the APPLETALK IPTALK-BASEPORT command, nor do you need to modify the file */etc/services*.

IPTalk Configuration Note

The installation instructions for CAP refer to KIP gateways and to the file *atalkatab*. If you use the DECbrouter 90 IPTalk support, the file *atalkatab* is neither necessary nor desirable. The DECbrouter 90 IPTalk support assumes that you wish all wide-area AppleTalk routing to be done using the normal AppleTalk routing protocols. KIP and *atalkatab* is based on a alternative routing strategy, where AppleTalk packets are passed around the Internet using IP routing. It is possible to use both strategies at the same time; however, the interaction of the two routing techniques is not well defined.

If you have routers from other vendors that support *atalkatab*, you should disable *atalkatab* support on them in order to avoid this mixed routing. The installation instructions that come with some of these products encourage you to use *atalkatab* for complex networks. When you are using the DECbrouter 90, this is not necessary. The DECbrouter 90 implementation of IPTalk integrates IPTalk into the normal AppleTalk network routing. Consider the example network diagram illustrated in Figure 2–6.

Figure 2–6 IPTalk Configuration Example



In this example, DECbrouter 90A and DECbrouter 90B must enable both standard AppleTalk (EtherTalk) and IPTalk on the Ethernets shown. They will use EtherTalk to communicate with the LocalTalk routers and Macintosh systems, and IPTalk to communicate with the UNIX systems. The LocalTalk routers also should have both EtherTalk and IPTalk enabled. IPTalk should be configured with *atalkatab* disabled. The LocalTalk routers will use IPTalk to communicate with the UNIX systems adjacent to them and EtherTalk to communicate with the rest of the AppleTalk network.

If you did not enable IPTalk on the LocalTalk routers, the system still will work. However systems on LocalTalk that wished to communicate with the nearby UNIX system would have to go through the DECbrouter 90. This creates an unnecessary extra hop, since the LocalTalk routers can speak IPTalk directly to the UNIX system.

Note

In this configuration, all traffic between systems on the left and the right transit via the DECbrouter 90s using AppleTalk routing. If *atalkatab* support is enabled on the LocalTalk routers, this would establish a *hidden* path between them, unknown to the more standard AppleTalk routing protocols. In a large network, this can result in traffic taking inexplicable routes.

AppleTalk Access List Configuration Examples

The ACCESS-LIST command examples that follow illustrate several AppleTalk access-list filter variations and contrast different approaches to access control list application.

Basic Access List Example

The following is a compilation of typical access-list statements:

```
access-list 601 permit zone ZoneA
access-list 601 permit zone ZoneB
access-list 601 deny zone ZoneD
access-list 601 deny additional-zones
access-list 601 permit network 55
access-list 601 permit network 500
access-list 601 permit cable-range 900-950
access-list 601 deny includes 970-990
access-list 601 permit within 991-995
access-list 601 deny other-access
```

These separate statements combine to establish access list number 601 with the following characteristics:

- Permits routing tuples to any network that has Zone A specified in its zone list
- Permits routing tuples to any network that has Zone B specified in its zone list
- Denies routing tuples to any network that has Zone D specified in its zone list
- Blocks routing tuples to any zone not specifically enumerated
- Permits routing tuples for nonextended network 55 and allows routing of packets destined for network 55
- Permits routing tuples for nonextended network 500 and allows routing of packets destined for network 500
- Permits routing tuples for the cable range 900–950 and allows routing of AppleTalk packets destined for any network in that range

- Denies any routing tuple that has a starting or ending network number within the range 970 and 990 inclusive and prevents the routing of AppleTalk packets destined for any network in that range
- Permits any routing tuple that has both starting and ending network numbers within the range of 991 and 995 inclusive and allows routing of AppleTalk packets destined to any network in that range
- Denies ACL routing tuples for any case that was not enumerated

Note

When applying an access control list such as this with the various interface subcommands (access-group, distribute-list, or getzonelist-filter) if an undefined access list is used, it defaults to permit; if a condition being tested is not handled by the specified access list, the router denies access by default.

To illustrate how the router tests incoming routing information against its access lists and interface specifications, consider the following test responses to detected conditions. Assume that the access list clauses for 601 are applied to a particular router interface. The following outcomes result from hypothetical tests:

- If the interface access control specification is testing a zone name of Zone C, no test is successful, so the additional-zones setting, deny for the example and by default, is the result.
- If the interface access control specification is testing a zone name of Zone B, the result is permit due to an explicit match.
- If the interface access control specification is testing a zone name of Zone D, the result is deny due to an explicit match.
- If the interface access control specification is testing a cable range of 55–55, the result is the *other-access* setting, deny for the example and by default. A cable range of 55 does not match a network number of 55 for the purposes of distribution list testing. However, if this list is used as an access group, 55 does match.
- If the interface access control specification is testing a cable range of 972–980, the result is to deny by explicit match.

Distribution list filtering operates on *exact* matches when making comparisons. The comparison is between an incoming routing tuple (which considers 55 and 55–55 to be different) and the condition defined in the access control list.

The process for accepting or rejecting routing information when applying distribution lists can be further defined with some illustrative examples.

Table 2–2 through Table 2–4 list the results associated with a specific test condition. If the outcome value is *true*, the condition passes the access list specification and the DISTRIBUTE-LIST interface subcommand specification is applied.

Table 2–2 Test Condition #1: Routing Tuple of 55

Example Access List Options Configured in Router	Outcome of Test
access-list 601 permit network 55	True
access-list 601 permit cable 55–55	False
access-list 601 permit includes 55–55	True
access-list 601 permit within 55–55	True

Table 2–3 Test Condition #2: Testing Routing Tuple of 55-55

Example Access List Options Configured in Router	Outcome of Test
access-list 601 permit network 55	False
access-list 601 permit cable 55–55	True
access-list 601 permit includes 55–55	True
access-list 601 permit within 55–55	True

Table 2–4 Test Condition #3: Testing routing Tuple of 55–60

Example Access List Options Configured in Router	Outcome of Test
access-list 601 permit network 50	False
access-list 601 permit network 55	False
access-list 601 permit cable 50–55	False
access-list 601 permit cable 50–50	False
access-list 601 permit cable 50–60	True
access-list 601 permit includes 50–55	True
access-list 601 permit includes 55–55	True
access-list 601 permit includes 50–50	True
access-list 601 permit within 50–55	False
access-list 601 permit within 55–55	False
access-list 601 permit within 50–60	True

For the ACCESS-GROUPS interface subcommand specifications used to control *packet flow*, the destination network number is used and all clauses are tested as if the test condition (**network**, **cable**, **includes**, or **within**) were actually **includes**. So, for the destination network of 55, all of the preceding test outcomes are *True* (when tested with access-groups) *except* for `network 50` and `cable 50--50`.

Note

For any set of values, no condition can overlap within the *same* access list. For this purpose, 50–50 and 50 are considered overlapping. However, access control lists used for different purposes on the same interface may contain entries that overlap in the different lists.

Comparison of Alternative Segmentation Solutions

With the flexibility allowed by the DECbrouter 90 access list implementation, determining the optimal method to segment an AppleTalk environment using access control lists can be unclear. The following scenario and configuration examples illustrate how two solutions can solve a particular problem and discuss the inherent advantages of using AppleTalk-style access lists.

Consider a situation where a company's management wants to permit customers to have direct access to several corporate file servers. Access to all devices in zones named MIS and Corporate is to be permitted, but other access is discouraged because unconstrained permission might facilitate unauthorized access to sensitive engineering file servers. The solution: create appropriate access control lists to enforce access policies.

The environment for this internet comprises the following networks and zones:

- Zone: Engineering. Network numbers/cable ranges: 69–69, 3, 4160–4160, 15
- Zone: MIS. Network numbers/cable ranges: 666–777
- Zone: Corporate. Network numbers/cable ranges: 70–70, 55, 51004, 4262–4262
- Zone: World. Network numbers/cable ranges:88–88, 9, 9000–49999 (multiple networks exist in this range)

The router named Gatekeeper is placed between the World zone and the various company-specific zones. There can be an arbitrary number of routers on either side of Gatekeeper. A backbone exists on each side which connects these other routers to Gatekeeper.

For the purposes of this configuration, assume Gatekeeper is the only router that needs any access list configuration. There are two solutions, depending on the level of security desired.

A minimal configuration might be as follows (the Engineering zone is secured, but all other zones are publicly accessible):

```
int ether 0
appletalk distrib 601 out
appletalk access 601
;
access-list 601 deny zone Engineering
access-list 601 permit additional-zones
access-list 601 permit other-access
```

A more comprehensive configuration might be as follows (Corporate and MIS zones are public; all other zones are secured):

```
int ether 0
appletalk distrib 601 out
appletalk access 601
;
access-list 601 permit zone Corporate
access-list 601 permit zone MIS
access-list 601 deny additional-zones
access-list 601 deny other-access
```

Both configurations satisfy the basic goal of isolating the Engineering servers, but the second example will continue to be secure when additional zones are added in the future.

Get-Zone-List Configuration Example

The following is an example of a get-zone-list (GZL) access filter implementation. In addition to the basic configuration commands, this example also provides the following:

- A discussion of the sequence of testing and route elimination associated with this filtering mechanism
- Lists the resulting information that will be included in the GZL

A GZL reply, per AppleTalk, contains a list of all zones. This can be modified by access lists to be a list of all zones which are associated with visible network entities and not explicitly excluded by an access list. The following configuration defines an access list that is used to modify the GZL, for interface Ethernet-0:

```
access-list 601 permit zone A
access-list 601 permit zone B
access-list 601 deny net 300
access-list 601 deny includes 1-100
access-list 601 permit other-access
access-list 601 permit zone D
access-list 601 deny additional-zones

access-list 602 permit zone A
access-list 602 permit zone B
access-list 602 deny additional-zones

int ether 0
appletalk distrib 601 out
appletalk getzonelist 602
```

The discussion that follows focuses on outlining the process of removing unwanted entries from an initial AppleTalk *zone/network association table*.

For the purposes of illustration, Table 2–5 matches the access list entries with arbitrary rule numbers. These rule numbers are then used to describe the process of route elimination employed by the AppleTalk access control mechanism.

Table 2–5 GZL Filter Example Access List Rules

Access List Entry	Rule Number
access-list 601 permit zone A	1
access-list 601 permit zone B	2
access-list 601 deny net 300	3
access-list 601 deny includes 1-100	4
access-list 601 permit other-access	5
access-list 601 permit zone D	6
access-list 601 deny additional-zones	7
access-list 602 permit zone A	8
access-list 602 permit zone B	9
access-list 602 deny additional-zones	10

Table 2–6 depicts a hypothetical initial state for an AppleTalk zone-network association table. This get-zone-list is then modified with the tests described in the following discussion.

Table 2–6 Initial Zone-Network Association Table

Network Number	Zone Name	Zone-Network Association
1-5	A	a1
98-102	A	a2
300	B	a3
400	C	a4
401	A	a5
402-402	D, B	a5

The first test applied to the router is to eliminate networks that are covered by access lists. The following network-zone associations are eliminated from the get-zone-list table:

- a1 and a2 are eliminated as a result of rule 4 (listed in Table 2–5).
- a3 is eliminated by rule 3.

Table 2–7 lists the network-zone associations that remain after the first test is completed on the initial table (elimination of networks from table per access list specification).

Table 2–7 Zone-Network Association Table After Access List Applied to Network

Network Number	Zone Name	Zone-Network Association
400	C	a4
401	A	a5
402-402	D, B	a6

The next test is the application of zone filtering using the distribution list. Network-zone association a4 is eliminated from the get-zone-list table as a result of applying rule 7, because no other zone rule applied. Table 2–8 lists the network-zone associations that remain after the distribution list test is completed on the list in Table 10-5 (elimination of network 400 per deny additional-zones access list specification).

Table 2–8 Zone-Network Association Table After Distribution List Test

Network Number	Zone Name	Zone-Network Association
401	A	a5
402-402	D, B	a6

Finally, zone filtering is applied through the APPLETALK GETZONELIST-FILTER. Network-zone association *a6* is eliminated from the get-zone-list table as a result of rule 10 zone D failed to meet any other zone rule.

Thus, the get-zone-list table will contain only a single entry (of those found in the initial table)—zone A.

Monitoring the AppleTalk Network

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

Displaying AppleTalk Access List Specifications

Use the SHOW APPLE ACCESS-LISTS EXEC command to display conditions specified in AppleTalk access list configurations. The following is a sample output from the associated configuration commands:

```
AppleTalk access list 601:
        permit zone ZoneA
        permit zone ZoneB
        deny additional-zones
        permit network 55
        permit network 500
        permit cable-range 900-950
        deny includes 970-990
        permit within 991-995
        deny other-access
```

Displaying the Adjacent Routes

The SHOW APPLETALK ADJACENT-ROUTES EXEC command results in a display of routes that are directly connected or one hop away. When an AppleTalk internet has more then 600 networks, this command gives administrators a quick synopsis of the local environment.

You can use information provided in this display to determine which local routes are missing or misconfigured so that appropriate action can be taken.

To show the routing table for adjacent routes, use the SHOW APPLE ADJACENT-ROUTES EXEC command:

show apple adjacent-routes

Following is a sample display for an extended AppleTalk network:

```
Codes: R - RTMP derived, C - connected, 67 routes in internet

R Net 29-29 [1/G] via gatekeeper, 0 sec, Ethernet0, zone Engineering
C Net 2501-2501 directly connected, serial1, no zone set
C Net 4160-4160 directly connected, Ethernet0, zone Low End SW Lab
C Net 4172-4172 directly connected, serial0, zone Low End SW Lab
R Net 6160 [1/G] via urk, 0 sec, serial0, zone Low End SW Lab
```

Displaying the ARP Cache

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

show apple arp

This command displays the contents of the AARP cache. AARP establishes correspondences between network addresses 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 2–9 describes the fields.

```
Protocol  Address          Age (min)     Hardware Addr   Type      Interface
AppleTalk 4172.30                -     0000.3080.84ab  SNAP      Ethernet0
AppleTalk 4160.19               94     0000.0c00.0082  SNAP      Ethernet0
AppleTalk 4160.21               94     0000.0c00.d8db  SNAP      Ethernet0
AppleTalk 2501.117               -     0000.0c00.d8de  SNAP      Ethernet0
AppleTalk 4172.224             206     0000.3080.8453  SNAP      Ethernet0
AppleTalk 4160.150               -     0000.0c00.d8dd  SNAP      Ethernet0
```

Table 2–9 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; entries are purged once they reach four hours (240 minutes) old
Hardware Addr	LAN hardware address that corresponds to network address
Type	Type of ARP: ARPA = Ethernet-type ARP SNAP = RFC 1042 ARP

Displaying the Fast-Switching Cache

Use the SHOW APPLE CACHE command with the extended AppleTalk networks to display the current fast-switching cache. Enter this command at the EXEC prompt:

show apple cache

This display includes the current cache version number and all entries (valid or not). Valid entries are identified by an asterisk (*) in the first column.

Conditions that invalidate the fast-switching cache are as follows:

- Route deleted but not marked bad (and has been used)
- A route that has gone bad (and has been used)
- When you replace a route with a new metric (and it was used)
- When a neighbor transitions from suspect to bad
- When a node address in the AARP cache changes hardware address
- When a hardware address changes node address
- When the AARP cache gets flushed
- When an AARP entry is deleted
- When the following configuration commands are entered:
  - — After a NO APPLETALK ROUTING command
  - — After an APPLETALK ROUTE-CACHE command

— After an AppleTalk ACCESS-LIST command

- When the encapsulation for the line changes
- When a port leaves or enters operational state

Following is a sample display of the SHOW APPLE CACHE command:

```
AppleTalk Routing Cache, * = active entry, cache version is 227
 Destination Interface MAC Header
*       29.0 Ethernet0 00000C00008200000C00D8DD
*   1544.000 Ethernet0 AA000400013400000C000E8C809B84BE02
*     33.000 Ethernet0 AA000400013400000C000E8C809B84BE02
```

Displaying Global AppleTalk Information

The EXEC command SHOW APPLETALK GLOBAL displays information about the AppleTalk internetwork and specific parameters for the router. The command has this syntax:

show apple global

Following is a sample display:

```
AppleTalk global information:
  Internet is compatible with older, AT Phase1, routers.
  There are 67 routes in the internet.
  There are 25 zones defined.
  All significant events will be logged.
  ZIP resends queries every 10 seconds.
  RTMP updates are sent every 10 seconds.
  RTMP entries are considered BAD after 20 seconds.
  RTMP entries are discarded after 60 seconds.
  AARP probe retransmit count: 10, interval: 200.
  AARP request retransmit count: 5, interval: 1000.
  DDP datagrams will be checksummed.
  RTMP datagrams will be strictly checked.
  RTMP routes may not be propagated without zones.
  Alternate node address format will not be displayed.
  Access control of any networks of a zone hides the zone.
  Names of local servers will be queried every 60 seconds.
  Lookups will be generated for server types:
        appleRouter, Workstation, GatorBox
```

Displaying AppleTalk Interface Information

The SHOW APPLE INTERFACE command displays AppleTalk-specific interface information. Enter this command at the EXEC prompt:

show apple interface [*interface*]

The argument *interface* specifies an interface name and number to display a specific interface.

Information displayed by this command includes the extended AppleTalk cable ranges and the current interface mode (the network verification/discovery mode, for example).

Sample displays of the SHOW APPLE INTERFACE command follow.

Nonextended AppleTalk—Normal Operation

```
Ethernet 0 is up, line protocol is up
  AppleTalk address is 666.128, Valid
  AppleTalk zone is Underworld
```

Extended AppleTalk—Normal Operation

Depending on the configuration of the global configuration commands APPLETALK LOOKUP-TYPE and APPLETALK NAME-LOOKUP-INTERVAL, a node name can appear in this display (in addition to the node address). For instance, in the example display output below, the node name urk is listed:

```
Serial 0 is up, line protocol is up
  AppleTalk cable range is 4172-4172
  AppleTalk address is 4172.30, Valid
  AppleTalk zone is "Low End SW Lab"
  AppleTalk port configuration provided by 4172.224 (urk)
  AppleTalk discarded 117 packets due to output errors
  AppleTalk discovery mode is enabled
  AppleTalk route cache is not supported by hardware
```

Extended AppleTalk—Verification Mode

```
Ethernet 0 is up, line protocol is up
  AppleTalk routing disabled, Verifying port configuration
  AppleTalk cable range is 666-666
  AppleTalk address is 666.128, Valid
  AppleTalk zone is Underworld
```

Extended AppleTalk—Configuration Error

```
Ethernet 0 is up, line protocol is up
  AppleTalk routing disabled, Port configuration error
  AppleTalk cable range is 70-70
  AppleTalk address is 70.128, Bad
  AppleTalk zone is Empty Guf
```

When you enter the EXEC command SHOW APPLE INTERFACE with the *interface* argument, the display looks like this:

```
Ethernet 0 is up, line protocol is up
  AppleTalk cable range is 69-69
  AppleTalk address is 69.105, Valid
  AppleTalk zone is "Empty Guf"
  AppleTalk port configuration verified by 69.163
  AppleTalk discarded 3149 packets due to input errors
  AppleTalk discarded 71 packets due to output errors
  AppleTalk route cache is enabled
```

If AppleTalk routing is disabled on an interface, the display looks like this:

```
Ethernet 0 is up, line protocol is up
  AppleTalk protocol processing disabled
```

Displaying MacIP Status

Two SHOW EXEC commands provide information concerning MacIP processes:

- **show apple macip-servers**
- **show apple macip-clients**

MacIP traffic statistics are displayed under the SHOW APPLETALK TRAFFIC command.

Each SHOW command is described in the brief sections that follow.

Monitoring MacIP Servers

Use the SHOW APPLETALK MACIP-SERVERS command to get information concerning the status of the servers for a router. The command syntax is as follows:

show appletalk macip-servers

The following is a sample output for this command:

```
MACIP SERVER 1, IP 131.108.199.221,  ZONE 'S/W Test Lab' STATE is server_
upResource #1 DYNAMIC 131.108.199.1-131.108.199.10, 1/10 IP in use
Resource #2 STATIC 131.108.199.11-131.108.199.20, 0/10 IP in use
```

A listing is provided for each MacIP server on the router. The following information is listed:

- MACIP SERVER—The number (arbitrarily assigned) of the MacIP server.
- IP—The IP address specified for the MacIP server.
- ZONE—The AppleTalk server zone specified in the APPLETALK MACIP SERVER command.
- STATE—The state of the server, as described in Table 2–10.
- Resource—Lists resource specifications as defined in the APPLETALK MACIP DYNAMIC and APPLETALK MACIP STATIC configuration statements. Specifies whether the resource address is assigned dynamically or statically; identifies the IP address range associated with the resource specification; and indicates the number of active MacIP clients.

This display is very useful in determining the status of your MacIP configuration. In particular, the STATE field can help identify problems in your AppleTalk environment. The following are hints for using this information:

- If the STATE remains at resource_wait, it is possible that no resources have been assigned (with either the APPLETALK MACIP DYNAMIC or APPLETALK MACIP STATIC commands).
- If the STATE remains at zone_wait, it is possible that an incorrect *server-zone* is specified in the APPLETALK MACIP SERVER command.

In addition, SHOW MACIP-SERVERS can be used along with SHOW APPLETALK INTERFACE to identify AppleTalk problems:

1. First, you can determine the state of the MacIP server using SHOW MACIP-SERVERS. If the STATE field persistently indicates an anomalous status (something besides server_up, such as resource_wait or zone_wait), a problem exists.

2. Next, execute a SHOW APPLETALK INTERFACE command; with this command you can determine the status of AppleTalk routing and the specific interface itself.
3. If the protocol and interface are up, inspect the MacIP configuration statements for IP address and zone specification inconsistencies.

The output of the SHOW MACIP-SERVERS command provides an indication of the current state of each configured MacIP server. Each server operates according to a simple finite-state machine table, described in Table 2–10.

Table 2–10 MacIP State Table

State	Event	New State	Notes
initial	ADD_SERVER	resource_wait	"server" configured
resource_wait	TIMEOUT	resource_wait	wait for resources
resource_wait	ADD_RESOURCE	zone_wait	wait for zone seeding
zone_wait	ZONE_SEEDED	server_start	register server
zone_wait	TIMEOUT	zone_wait	wait until seeded
server_start	START_OK	reg_wait	wait for server reg
server_start	START_FAIL	del_server	couldn't start (config err?)
reg_wait	REG_OK	server_up	registration successful
reg_wait	REG_FAIL	del_server	reg. failed (duplicate IP?)
reg_wait	TIMEOUT	reg_wait	wait until register
server_up	TIMEOUT	send_confirms	NBP confirm all clients
send_confirms	CONFIRM_OK	server_up	
send_confirms	ZONE_DOWN	zone_wait	zone or IP interface down, restart
*	ADD_RESOURCE	*	ignore, except resource_wait
*	DEL_SERVER	del_server	"no server" statement (HALT)
*	DEL_RESOURCE	ck_resource	ignore
ck_resource	YES_RESOURCES	*	return to previous state
ck_resource	NO_RESOURCES	resource_wait	shutdown, wait for resources

The following are descriptions of the state functions:

- **initial**—All servers begin here.
- **resource_wait**—Wait until a client range has been configured for the server.
- **zone_wait**—Wait until the configured AppleTalk zone name for the server is up.

  Note

  The server will remain in this state if no such zone has been configured or if AppleTalk routing is not enabled.

- **server_start**—Register configured IPADDRESS, and register as IPGATEWAY. Open ATP socket to listen for IP address assignment requests. Send NBP lookup requests for existing IPADDRESSes, and automatically add clients with addresses within one of the configured client ranges.
- **server_up**—Server has registered. Enable routing to client ranges. Respond to IP address assignment requests.
- **send_confirms**—Send NBP confirm tickles active clients every minute. Delete clients that have not responded within the last five minutes. Check IP and AppleTalk interfaces used by MacIP server. If down or reconfigured, restart server.
- **del_server**—All servers end here. Deregister NBP names, purge all clients and deallocate server resources.
- **ck_resource**—Make sure there is at least one client range available. If not, deregister NBP names and return to resource_wait state.
- *—If in first column, represents "any" state. If in second column, represents a return to state from which a * state was called.

Monitoring MacIP Clients

Use the SHOW APPLE MACIP-CLIENTS commands and to get information concerning the status of the known clients. The command syntax is as follows:

show apple macip-clients

The SHOW MACIP-CLIENTS command displays the IP and DDP address of all MacIP clients and the last time the client responded to an NBP confirm request.

Clients are deleted after five minutes of not responding to NBP confirm requests on their allocated IP addresses.

The following is a sample output for this command:

```
131.108.199.1@[27001n,69a,72s] 45 secs    'S/W Test Lab'
```

The resulting display lists all known MacIP clients by IP address. Bracketed information includes the AppleTalk DDP address of the registered entity (network, node address, and socket number), followed by the time since the lase NBP confirmation and name of the zone to which this particular MacIP client is attached.

Monitoring MacIP Traffic

Use the show APPLETALK TRAFFIC command to get information concerning the status of the MacIP traffic. The command syntax is as follows:

show apple traffic

An IP alias is established for each MacIP client and for the IP address of the MacIP server, if it does not match an existing IP interface address. The client aliases can be viewed with the SHOW IP ALIASES command.

Displaying Nearby NBP Services

Use the SHOW APPLETALK NAME-CACHE EXEC command to display a list of NBP services of nearby routers or other devices that support NBP. The syntax is as follows:

show appletalk name-cache

Note

The SHOW APPLETALK NAME-CACHE command can be authorized by the administrator to display any AppleTalk services of interest in local zones, whereas the SHOW APPLETALK NBP command is used to show services registered by the router.

This is sample output for the SHOW APPLETALK NAME-CACHE command:

```
AppleTalk Name Cache:
  Net Adr Skt Name                    Type            Zone
 4160  19 254 gatekeeper              DigitalRouter     Low End SW Lab
 4160  21 254 bill                    DigitalRouter     Low End SW Lab
 4160 150 254 pag.Ethernet0           DigitalRouter     Low End SW Lab
 4172  30 254 pag.Serial0             DigitalRouter     Low End SW Lab
 4172 224 254 urk                     DigitalRouter     Low End SW Lab
 6160  69 254 urk                     DigitalRouter     Low End SW Lab
```

This information is held in the NBP name cache.

Support for names allows administrators to easily identify and determine the status of any associated device. This can be important in AppleTalk internetworks where node numbers are dynamically generated.

Note

Non-DECbrouter 90 routers will have a naming format that does not include an appended interface name. The interface (`ethernet0`) included in the derived name `pag.ethernet0` in this display refers to the router pag's view of the world—*not* the local router's view. They may be, but are not necessarily, the same. This feature allows you to determine the routers and their connected interfaces that are providing routing for any given AppleTalk network.

Displaying NBP Services Registered by Digital Routers

Use the SHOW APPLETALK NBP EXEC command to display the NBP name registration table. The command syntax is as follows:

show appletalk nbp

The following is a sample output:

```
Net  Adr Skt Name                  Type            Zone
4160 211 254 pag.Ethernet0         DigitalRouter     Low End SW Lab
4160 211   8 pag                   SNMP Agent      Low End SW Lab
4172  84 254 pag.Serial0           DigitalRouter     LES Tokenring
4172  84   8 pag                   SNMP Agent      LES Tokenring
 200  75 254 myrouter.Ethernet0    DigitalRouter     Marketing    *
```

Note

The SHOW APPLETALK NBP command is used to show services registered by the router, whereas the SHOW APPLETALK NAME-CACHE command can be authorized by the administrator to display any AppleTalk services of interest in local zones.

In this display, the fields are as follows:

- **Net**—AppleTalk network number
- **Adr**—Node address
- **Skt**—DDP socket number
- **Name**—Name of service
- **Type**—Device type, varies depending on service. The Digital service types are:
  - **DigitalRouter**—Listed in show appletalk nbp display per port
  - **SNMP Agent**—Listed in show appletalk nbp display per zone if and only if Apple's snmp-over-ddp is enabled
  - **IPGATEWAY**—Active MacIP server names
  - **IPADDRESS**—Active MacIP server addresses

If an asterisk (*) appears in the far right margin, the name registration is pending confirmation.

Displaying Neighboring Routers

The SHOW APPLE NEIGHBOR EXEC command shows all AppleTalk routers that are directly connected to any of the networks to which this router is directly connected. It is from these neighboring routers that this router obtains the AppleTalk network topology and most of the other information it needs to support the protocol. The command has this syntax:

show apple neighbor [*neighbor-address*]

The optional argument *neighbor-address* permits access to detailed statistics and other information associated with a particular neighbor.

For the command SHOW APPLE NEIGHBOR, the display looks like this:

```
AppleTalk neighbors:

 31.86, Ethernet0, uptime 133:28:06, last update 1 sec ago
 81.81, Ethernet0, uptime 267:30:28, last update 958334 secs ago
   Neighbor is down.
 29.200, Serial0, uptime 263:45:50, last update 948440 secs ago
  Neighbor has restarted 2 times in 267:59:53.
   Neighbor is down.
 17.128, Serial0, uptime 133:26:43, last update 2 secs ago
  Neighbor has restarted 1 time in 268:00:21.
 69.163, Serial1, uptime 268:00:25, last update 1 sec ago
```

Depending on the configuration of the global configuration commands APPLETALK LOOKUP-TYPE and APPLETALK NAME-LOOKUP-INTERVAL, a node name can appear in this display (as well as a node address). For instance, in the example display output below, the node names `urk`, `gatekeeper`, and `bill` are listed:

```
AppleTalk neighbors:
  4172.224     urk      Serial0, uptime 63:35:42, 1 sec
        Neighbor has restarted 2 times in 125:16:47.
  4160.19      gatekeeper       Ethernet0, uptime 125:17:53, 1 sec
  4160.21      bill     Ethernet0, uptime 13:07:55, 5 secs
        Neighbor has restarted 5 times in 89:53:09.
```

For the command SHOW APPLE NEIGHBOR 69.163, the display looks like this:

```
 Neighbor 69.163, Ethernet0, uptime 268:00:52, last update 7 secs ago
 We have sent queries for 299 nets via 214 packets.
 Last query was sent 4061 secs ago.

 We received 152 replies and 0 extended replies.
 We have received queries for 14304 nets via 4835 packets.
 We sent 157 replies and 28 extended replies.
 We received 0 ZIP notifies.
 We received 0 obsolete ZIP commands.
 We received 4 miscellaneous ZIP commands.
 We received 0 unrecognized ZIP commands.
 We have received 92943 routing updates.
 Of the 92943 valid updates, 1320 entries were invalid.
 We received 1 routing update which were very late.
 Last update had 0 extended and 2 nonextended routes.
 Last update detail: 2 old
```

If the global configuration commands APPLETALK LOOKUP-TYPE and APPLETALK-NAME-LOOKUP interval have been configured, a node name can appear in this display (as well as a node address). For instance, in the example display output below the node name urk is listed:

```
Neighbor 4172.224, Serial0, uptime 63:36:19, updated 8 secs ago
        The neighbors address is 4172.224, and named urk.
        We have sent queries for 0 nets via 0 packets.
        We received 0 replies and 0 extended replies.
        We have received queries for 143 nets via 12 packets.
        We sent 12 replies and 60 extended replies.
        We received 0 ZIP notifies.
        We received 0 obsolete ZIP commands.
        We received 4 miscellaneous ZIP commands.
        We received 0 unrecognized ZIP commands.
        We have received 44856 routing updates.
        Of the 44856 valid updates, 0 entries were invalid.
        We received 0 routing updates which were very late.
        Last update had 0 extended and 1 non-extended routes.
        Last update detail: 1 old
        The neighbor has restarted 2 times in 125:17:24.
        Cached service names for urk:
           urk:DigitalRouter@Low End SW Lab, socket 254
```

Note

The cached service names (which are used to determine the router name) and the neighbor's name (listed with its address) are only listed when **appletalk lookup-type** is enabled.

Displaying the Network Routing Table

To show the routing table for networks, use the SHOW APPLE ROUTE EXEC commands:

show apple route [*network*]
show apple route [*interface-name*]

This command displays either the full routing table or just the entry for the optionally specified *network* for both extended and nonextended AppleTalk networks. For the extended AppleTalk networks, the command also displays cable ranges information.

The optional interface-name argument specifies an interface name to report on. Displays for both nonextended and extended AppleTalk networks follow.

A sample display for a nonextended AppleTalk network:

```
Codes: R - RTMP derived, C - connected, S - static, 3 routes
C Net 258 directly connected, 1431 uses, Ethernet0, zone Twilight
R Net 6 [1/G] via 258.179, 8 sec, 0 uses, Ethernet0, zone The O
C Net 11 directly connected, 472 uses, Serial1, zone No Parking
R Net 2154 [1/G] via 258.179, 8 sec, 6892 uses, Ethernet0, zone LocalTalk
S Net 1111 via 258.144, 0 uses, Ethernet0, no zone set
[hops/state] state can be one of G:Good, S:Suspect, B:Bad
```

In the above display, the G rating after Net 6 indicates *good*. Alternate ratings are S for *suspect* and B for *bad*. These ratings are attained from the routing updates that occur at 10-second intervals. A separate and nonsynchronized event occurs at 20-second intervals, checking and flushing the ratings for particular routes that have not been updated. For each 20-second period that passes with no new routing information, a rating will slip from G to S to B; after one minute with no updates, that route will be flushed. Every time the router receives a useful update, the status of the route in question is reset to G. Useful updates are those advertising a route that is as good or better than the one currently in the table.

Following is a sample display for the extended AppleTalk network. Note the cable range display for Magnolia Estates:

```
Codes: R - RTMP derived, C - connected, 29 routes in internet

R Net 3 [1/G] via 254.163, 8 sec, Ethernet0, zone Localtalk
C Net 4 directly connected, Ethernet0, zone Twilight
C Net 6 directly connected, Ethernet0, zone Heavenly
R Net 11 [3/G] via 254.163, 8 sec, Ethernet0, zone UDP
R Net 17 [1/G] via 254.163, 8 sec, Ethernet0, zone UDP
R Net 33 [1/G] via 4.129, 1 sec, Ethernet0, zone Twilight
R Net 36 [1/G] via 254.174, 7 sec, Ethernet0, zone idontcare
R Net 55 [1/G] via 254.130, 9 sec, Ethernet0, zone Hospital
R Net 69 [1/G] via 4.129, 1 sec, Ethernet0, zone Empty Guf
R Net 70 [1/G] via 254.247, 2 sec, Ethernet0, zone Empty Guf
C Net 80 directly connected, Ethernet0, zone Light
R Net 99 [2/G] via 4.129, 1 sec, Ethernet0, zone BammBamm
C Net 254 directly connected, Ethernet0, zone Twilight
R Net 890 [2/G] via 4.129, 1 sec, Ethernet0, zone release lab
R Net 901 [2/G] via 4.129, 1 sec, Ethernet0, zone Dave's House
C Net 999-999 directly connected, Serial0, zone Magnolia Estates
R Net 2003 [4/G] via 80.129, 6 sec, Ethernet0, zone Bldg-13
R Net 2004 [2/G] via 80.129, 6 sec, Ethernet0, zone Bldg-17
R Net 2012 [2/G] via 4.130, 7 sec, Ethernet0, zone Bldg-13
R Net 2013 [3/G] via 254.163, 8 sec, Ethernet0, zone UDP
R Net 2024 [4/G] via 80.129, 3 sec, Ethernet0, zone Bldg-17
R Net 3004 [1/G] via 80.129, 3 sec, Ethernet0, zone Bldg-17
R Net 3012 [1/G] via 4.130, 5 sec, Ethernet0, zone Bldg-13
R Net 3024 [4/G] via 80.129, 3 sec, Ethernet0, zone Bldg-17
R Net 3880 [1/G] via 999.2, 0 sec, Serial0, zone Magnolia Estates
R Net 5002 [2/G] via 80.129, 3 sec, Ethernet0, zone Bldg-17
R Net 5003 [2/G] via 4.130, 5 sec, Ethernet0, zone Bldg-13
R Net 5006 [4/G] via 80.129, 3 sec, Ethernet0, zone Bldg-17
R Net 51489 [3/G] via 4.129, 8 sec, Ethernet0, zone Dave's House
```

Depending on the configuration of the global configuration commands APPLETALK LOOKUP-TYPE and APPLETALK NAME-LOOKUP-INTERVAL, a node name can appear in this display (instead of a node address). For instance, in the example display output that follows, the node name gatekeeper is listed:

```
   Codes: R - RTMP derived, C - connected, 67 routes in internet

   R Net 3 [2/G] via gatekeeper, 4 sec, Ethernet0, zone Engineering
   R Net 4 [3/G] via gatekeeper, 4 sec, Ethernet0, zone Twilight
R Net 6 [4/G] via gatekeeper, 4 sec, Ethernet0, zone Heavenly
R Net 11 [4/G] via gatekeeper, 4 sec, Ethernet0, zone UDP
R Net 12-12 [3/G] via gatekeeper, 4 sec, Ethernet0, zone UDP
R Net 17-17 [2/G] via gatekeeper, 4 sec, Ethernet0, zone Twilight
R Net 19-19 [3/G] via gatekeeper, 4 sec, Ethernet0, zone customer eng
R Net 29-29 [1/G] via gatekeeper, 4 sec, Ethernet0, zone Engineering
R Net 33 [2/G] via gatekeeper, 4 sec, Ethernet0, zone Twilight
R Net 69-69 [2/G] via gatekeeper, 4 sec, Ethernet0, zone Empty Guf
R Net 80 [3/G] via gatekeeper, 4 sec, Ethernet0, zone Light
R Net 199-199 [6/G] via gatekeeper, 4 sec, Ethernet0, zone Tir'n nag
R Net 550 [4/G] via gatekeeper, 4 sec, Ethernet0, zone outside Digital
R Net 560 [4/G] via gatekeeper, 4 sec, Ethernet0, zone outside Digital
R Net 666-666 [2/G] via gatekeeper, 4 sec, Ethernet0, zone Gates of Hell
R Net 2010 [7/G] via gatekeeper, 4 sec, Ethernet0, zone europe
R Net 2500-2500 [6/G] via gatekeeper, 4 sec, Ethernet0, zone Looking Glass
C Net 2501-2501 directly connected, Ethernet0, no zone set
R Net 3004 [3/G] via gatekeeper, 4 sec, Ethernet0, zone Bldg-17
R Net 3010 [6/G] via gatekeeper, 4 sec, Ethernet0, zone europe
```

The next sample shows the result of the SHOW APPLE ROUTE command with a specific network.

For the command SHOW APPLE ROUTE 69, the display looks like this:

```
Codes: R - RTMP derived, C - connected, 67 routes in internet

R Net 69-69 [2/G] via gatekeeper, 0 sec, Ethernet0, zone Empty Guf
 Route installed 125:20:21, updated 0 secs ago
  Next hop: gatekeeper, 2 hops away
  Zone list provided by gatekeeper
  Route has been updated since last RTMP was sent
  Valid zones: "Empty Guf"
```

Depending on the configuration of the global configuration commands APPLETALK LOOKUP-TYPE and APPLETALK NAME-LOOKUP-INTERVAL, a node name can appear in this display (instead of a node address). For instance, in the example display output above, the node name gatekeeper is listed.

For the command SHOW APPLE ROUTE SERIAL 0, the display looks like this:

```
Codes: R - RTMP derived, C - connected, 29 routes in internet

C Net 999 directly connected, Serial0, zone Magnolia Estates
R Net 3880 [1/G] via 999.2, 3 sec, Serial0, zone Magnolia Estates
```

Displaying Information About the Sockets

The command SHOW APPLE SOCKET displays information about the process-level processing in all the sockets in the AppleTalk interface. Enter this command at the EXEC prompt:

show apple socket [*socket-number*] li

When used with the optional *socket-number* argument, it shows information about a specific socket.

The following is the output seen when no socket number is specified:

```
Socket   Name            Owner                  Waiting/Processed
    1   RTMP            AT RTMP                0 148766
    2   NIS             AT NBP                 0 156429
    4   AEP             AT Maintenance         0      0
    6   ZIP             AT ZIP                 0  13619
    8   SNMP            AT SNMP                0      0
  253   PingServ        AT Maintenance         0      0
```

When a socket is specified, only statistics for that socket are displayed, as seen in following sample output:

```
6    ZIP              AT ZIP  0   2704
```

Displaying AppleTalk Traffic Information

The EXEC command SHOW APPLE TRAFFIC displays AppleTalk-specific traffic information. The command has this syntax:

show apple traffic

The statistics it displays include the total number of packets received, categorized errors, summaries of packets received for the various AppleTalk services (for example, NBP, ZIP, DDP) and for other protocols such as Echo and ARP. Several counters have also been added to monitor extended AppleTalk activity. See Table 2–11.

Following is a sample display of extended AppleTalk activity.

```
AppleTalk statistics:
  Rcvd:  357471 total, 0 checksum errors, 264 bad hop count
         321006 local destination, 0 access denied
         0 for MacIP, 0 bad MacIP, 0 no client
         13510 port disabled, 2437 no listener
         0 ignored, 0 martians
  Bcast: 191881 received, 270406 sent
  Sent:  550293 generated, 66495 forwarded, 1840 fast forwarded
         0 forwarded from MacIP, 0 MacIP failures
         436 encapsulation failed, 0 no route, 0 no source
  DDP:   387265 long, 0 short, 0 macip, 0 bad size
  NBP:   302779 received, 0 invalid, 0 proxies
         57875 replies sent, 59947 forwards, 418674 lookups, 432 failures
  RTMP:  108454 received, 0 requests, 0 invalid, 40189 ignored
         90170 sent, 0 replies
  ATP:   0 received
  ZIP:   13619 received, 33633 sent, 32 netinfo
  Echo:  0 received, 0 discarded, 0 illegal
         0 generated, 0 replies sent
  Responder:  0 received, 0 illegal, 0 unknown
         0 replies sent, 0 failures
  AARP:  85 requests, 149 replies, 100 probes
         84 martians, 0 bad encapsulation, 0 unknown
         278 sent, 0 failures, 29 delays, 315 drops
  Lost: 0 no buffers
  Unknown: 0 packets
  Discarded: 130475 wrong encapsulation, 0 bad SNAP discriminator
```

Table 2–11 Show Apple Traffic Field Descriptions

Field	Description
checksum errors	The DDP checksum was incorrect, so these packets were discarded. The DDP checksum is verified for packets that are directed to the router. Forwarded packets do not have their checksums verified enroute.
bad hop count	Packet dropped; the packet has traveled too many hops.
local destination	The number of packets that were received for processing by the router.
access denied	Packet dropped; access list did not permit it.
no client	The number of packets that were directed to a MacIP client but that were not present. The packets were discarded.
port disabled	Packet dropped, routing disabled for port (extended AppleTalk only). Occurs because of a configuration error or a packet received while in verification/discovery mode.
no listener	The number of packets directed to a socket on the router that does not have any services associated with that socket. The packets were discarded.

(continued on next page)

Table 2–11 (Cont.) Show Apple Traffic Field Descriptions

Field	Description
ignored	The number of routing update packets that were ignored because the packet was from a misconfigured neighbor. Also, packets are ignored when routing is disabled.
martians	The number of packets that were discarded because they contained bogus information in the DDP header. What distinguishes this error from the others is that the data in the header is never valid as opposed to not being valid at a given point in time.
fast forwarded	Packets that were forwarded using data from the fast switching (route cache). These packets incur the least delay and cause the least impact with respect to the router.
encapsulation failed	Packet received for a connected network, but node's MAC address not found.
bad size	Physical packet length and claimed length disagree.
netinfo	Number of packets that requested port configuration via ZIP GetNetInfo requests. Originally, these were exclusively used during node startup, but are now used by some AppleTalk network management software packages.
unknown	Unknown AppleTalk packet type.
no buffers	Attempted packet buffer allocation failed.
wrong encapsulation	Nonextended AppleTalk packet on extended AppleTalk port, or vice versa.
bad SNAP discriminator	Extended AppleTalk packet without Apple discriminator (extended AppleTalk only). Occurs when another AppleTalk device has implemented an obsolete or incorrect packet format.

Displaying Zone Information

The SHOW APPLE ZONE command displays the zone information table and has this syntax:

show apple zone [*zonename*]

Use this command to display which networks comprise each zone for both nonextended and extended AppleTalk networks.

The argument *zonename* specifies the name of the zone you are trying display information on.

In the following sample display, notice the report of cable ranges for the extended zone Empty Guf:

```
Name                                Network(s)
Gates of Hell                       666-666
Engineering                         3 29-29 4042-4042
customer eng                        19-19
Digital IP                            4140-4140
Dave's House                        3876 3924 5007
Narrow Beam                         4013-4013 4023-4023 4037-4037 4038-4038
Low End SW Lab                      6160 4172-4172 9555-9555 4160-4160
Tir'n na'Og                         199-199
Mt. View 1                          7010-7010 7122 7142 7020-7020 7040-7040
                                    7060-7060
Mt. View 2                          7152 7050-7050
UDP                                 11 12-12
Empty Guf                           69-69
Light                               80
europe                              2010 3010 3034 5004
Bldg-13                             4032 5026 61669 3012 3025 3032 5025 5027
Bldg-17                             3004 3024 5002 5006
S/W Test Lab                        27001-27001
Dead Ringer                         4028-4028 4035-4035 4036-4036
outside Digital                       550 560 4014-4014 4020-4020
Pin Point                           25346 25344 25345-25345
```

If a specific *zonename* is specified, the display output appears as follows:

```
AppleTalk Zone Information for Digital IP:
  Valid for nets: 4140-4140
  Not associated with any interface.
  Not associated with any access list.
```

The AppleTalk Ping Command

The EXEC PING command sends Echo Protocol datagrams to other AppleTalk nodes to verify connectivity and measure round-trip times.

When the PING command prompts for a protocol, specify appletalk. Default options are indicated with carriage returns. What follows is a sample of using PING with the AppleTalk protocol. To abort a ping session, type the escape sequence (by default, type Ctrl/^X, which is done by simultaneously pressing the Ctrl, Shift, and 6 keys, letting go then pressing the X key).

Sample Session

```
Protocol [ip]: appletalk
Target Appletalk address: 1024.128
Repeat count :
Datagram size [100]:
Timeout in seconds :
Verbose [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte AppleTalk Echos to 1024.128, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 4/4/8 ms
```

Note

Only an interface that supports *HearSelf* can respond to packets generated at a local console and directed to an interface on the same router. The DECbrouter 90 only supports *HearSelf* on Ethernet.

The PING command uses the characters in Table 2–12 to indicate the success or failure of each packet in the **ping** sequence.

Table 2–12 AppleTalk Ping Characters

Character	Meaning
!	The packet was echoed successfully from the target address.
.	The timeout period expired before an echo was received from the target address.
B	Bad or malformed echo was received from the target address.
C	An echo was received with a bad DDP checksum.
E	Transmission of the echo packet to the target address failed.
R	The transmission of the echo packet to the target address failed for lack of a route to the target address.

AppleTalk NBP Ping Interface

The PING EXEC command for AppleTalk allows testing and informational lookup of NBP registered entities.

To use this privileged facility, enter **ping**) and respond to the protocol prompt with the keyword **appletalk**. Then enter the keyword *nbp* in response to the prompt `Target AppleTalk address:`

The following is an example of the sequence used to initialize the AppleTalk *nbptest* utility accessed via the PING command.

```
myrouter# ping
Protocol [ip]: appletalk
Target AppleTalk address: nbp
nbptest>
```

The *nbptest* facility is an interactive, menu-driven facility. Type `help` or `?` to see the command list. Type `quit` to return to the EXEC prompt. The sections that follow describe the subcommands available from the *nbptest* utility invoked with this command.

Help Subcommand

The HELP subcommand of the *nbptest* utility displays the available tests in a menu.

The following is a sample of the menu displayed:

```
nbptest> help
Tests are:

lookup:     lookup an NVE. prompt for name, type and zone
parms:      display/change lookup parms (ntimes, nsecs, interval)
zones:      display zones
poll:       for every zone, lookup all devices,using default
help|?:     print command list
quit:       exit nbptest
```

Parms Subcommand

The PARMS subcommand of the *nbptest* utility sets the *lookup* parameters used in subsequent lookup and poll commands.

The following is an example parameter configuration sequence.

```
nbptest> parms
maxrequests [10]: 1
maxreplies : 100
interval : 10
```

Note

If the values of the PARMS subcommand are revised, the next time this menu is activated the parms last entered appear in brackets.

In the preceding example, the number of lookup retries is set to 1, the maximum number of replies to accept for each lookup is set to 100, and the interval between each retry is set to 10 seconds.

The defaults for `maxrequests`, `maxreplies`, and `interval` are 10, 5, and 5, respectively; the current value is indicated in brackets in the prompt for each parameter.

The acceptable ranges are as follows:

- `maxrequests`—1 to 5 (integer value) requests
- `maxreplies`—1 to 500 (integer value) replies
- `interval`—1 to 60 (integer value) seconds

Lookup Subcommand

Use the LOOKUP subcommand to search for NBP entities in a specific zone. The PARAMS command can be used to adjust the lookup parameters. Nonprinting characters can be specified by entering a three-character string specifying the hexadecimal equivalent (for example, `:c5` specifies the NBP truncation wildcard).

Example

The following example sequence illustrates the specification of the PARM subcommand parameter.

```
nbptest> parms
maxrequests [10]: 1
maxreplies [5]: 100
interval : 10

nbptest> lookup
Entity name [=]:
Type of Service [ipgateway]: macintosh:c5
Zone [bldg-17]: engineering
(100n,50a,253s)[1]:   'userA:Macintosh IIcx@engineering'
(100n,16a,251s)[1]:   'userB:Macintosh II@engineering'
(200n,24a,253s)[1]:   'userC:Macintosh IIci@engineering'
(200n,36a,253s)[1]:   'userD:Macintosh IIci@engineering'
(300n,21a,252s)[1]:   'userE:Macintosh SE/30@engineering'
(300n,97a,251s)[1]:   'userF:Macintosh SE/30@engineering'
NBP lookup request timed out
Processed 6 replies, 7 events
```

The AppleTalk DDP address of the registered entity is displayed in parentheses, (network, node address, and socket number), followed by the NBP enumerator and the NBP entity string.

Note

If the values of the PARMS subcommand are revised, the next time this menu is activated, the parameters last entered appear in brackets.

Poll Subcommand

Use the POLL command to search for all devices in all zones according to the current lookup parameters. The poll command posts a lookup of the form '=:=@zone" for each zone in the AppleTalk internet.

In a large AppleTalk internetwork, the POLL subcommand will return several hundred replies and generate a large amount of network activity, so it should be used with caution.

The following is a sample output for this command:

```
poll:  sent 2 lookups
(100n,82a,252s)[1]:   'userA:Macintosh IIci@Zone one'
(200n,75a,254s)[1]:   'userB:Macintosh IIcx@Zone two'
NBP polling completed.
Processed 2 replies, 2 events
```

The AppleTalk DDP address of the registered entity is displayed in parentheses, (network, node address, and socket number), followed by the NBP enumerator and the NBP entity string.

Zones Subcommand

The ZONES subcommand displays the current zone list in the router. It is equivalent to the SHOW APPLE ZONES EXEC command and is included in *nbptest* for convenience.

The following is a sample output for this command:

```
Name                                    Network(s)
UDP                                     17 11
Heavenly                                1161 6
Hospital                                55
Bldg-17                                 82 81 14 13
CSL EtherTalk                           22
Twilight                                1544 254 36 33 4
EtherTalk                               2
Underworld                              666
Magnolia Estates                        3880 999
Light                                   80
LocalTalk                               3
Empty Guf                               69-69
Total of 12 zones
```

Debugging the AppleTalk Network

The EXEC DEBUG commands described in this section are used to troubleshoot the AppleTalk network transactions. Generally, you enter these commands during troubleshooting sessions with Digital customer engineers.

For each DEBUG command, there is a corresponding UNDEBUG command that turns the display off. Remember that some of these commands can be entered in groups that then display additional information.

debug appletalk

The DEBUG APPLETALK command debugs all startup messages and protocol routines dedicated to support startup. This command also debugs global messages such as those regarding neighbors, ports/interfaces, and configuration. The command looks at problems with parts of Appletalk that do not have their own options in other debug commands.

debug apple-aarp

The DEBUG APPLE-AARP command enables debugging of AppleTalk address resolution protocol. A side effect of enabling this option is that gleaning MAC information from datagrams is disabled.

debug apple-errors

The DEBUG APPLE-ERRORS command reports information about errors that occur. The information displayed by this command is enhanced by enabling debugging for the specific class of errors that you are interested in. This is similar to DEBUG APPLE-PACKETS.

debug apple-event

The DEBUG APPLE-EVENT command displays debugging information about AppleTalk special events, neighbors becoming reachable/unreachable, and interfaces going up/down. Only significant events (for example, neighbor and/or route changes) are logged. This command is maintained in nonvolatile memory, if present.

appletalk event-logging

The APPLETALK EVENT-LOGGING command causes logging of a subset of messages produced by DEBUG APPLETALK command. Logs significant events using the logger facility. Logged events include routing changes, zone creation, port status, and address.

debug apple-nbp

The DEBUG APPLE-NBP command enables debugging output from the name binding protocol (NBP) routines.

debug apple-packet

The DEBUG APPLE-PACKET command enables per-packet debugging output. It reports information online when a packet is received or a transmit is attempted. The command allows watching the types of packets being slow-switched. It is roughly equivalent to turning on all the other AppleTalk debugging information. There will be at least one line of debugging output per AppleTalk packet processed.

The DEBUG APPLE-PACKET command, when invoked in conjunction with the commands DEBUG APPLE-ROUTING, DEBUG APPLE-ZIP, and DEBUG APPLE-NBP, adds protocol processing information in addition to generic packet details. It reports protocol processing, and successful completion or failure information.

The DEBUG APPLE-PACKET command, when invoked in conjunction with the command DEBUG APPLE-ERRORS, reports packet level problems such as encapsulation problems. This is the case because DEBUG APPLE-ERRORS is a subset of DEBUG APPLE-PACKETS.

debug apple-routing

The DEBUG APPLE-ROUTING command enables debugging output from the routing table maintenance protocol (RTMP) routines. This command can be used to monitor acquisition of routes, aging of routing table entries, and advertisement of known routes. It also reports conflicting network numbers on the same network if the network is misconfigured.

debug apple-zip

The DEBUG APPLE-ZIP command enables debugging output from the zone information protocol (ZIP) routines. This command reports significant events such as discovery of new zones and zone list queries.

AppleTalk Global Configuration Command Summary

This section lists all the global commands used with the AppleTalk interface.

[**no**] **access-list** *list* {**permit** | **deny**} **network** *network*
[**no**] **access-list** *list* {**permit** | **deny**} **cable-range** *start-end*
[**no**] **access-list** *list* {**permit** | **deny**} **includes** *start-end*
[**no**] **access-list** *list* {**permit** | **deny**} **within** *start-end*
[**no**] **access-list** *list* {**permit** | **deny**} **zone** *zonename*
no access-list *list*
access-list *list* {**permit** | **deny**} **additional-zones**
access-list *list* {**permit** | **deny**} **other-access**

Defines an AppleTalk access list. This command has several optional formats and supports *extended* AppleTalk networks. The argument *list* is an integer from 600 to 699 and the argument *network* is an AppleTalk network number. Additional **permit** and **deny** conditions can be added to the list by issuing further ACCESS-LIST commands for that list. Use the NO ACCESS-LIST command with the *list* number only to remove an entire access list from the configuration. Specify the optional arguments to remove a particular clause.

no appletalk arp

Resets the ARP INTERVAL and ARP RETRANSMIT commands to their default values.

appletalk arp {**request | probe**} **interval** *milliseconds*

Specifies the time interval between retransmission of ARP packets. The argument *milliseconds* specifies the interval. The default is 200 when the **probe** keyword is used and 1000 when the **request** keyword is used. The minimum value is 33 milliseconds. The command NO APPLETALK ARP or a *milliseconds* value of 0 resets the default.

appletalk arp {**request | probe**} **retransmit-count** *count*

Specifies the number of retransmissions that will be done before abandoning address negotiations and using the selected address. The argument *count* specifies the retransmission count. The default is 10 when the **probe** keyword is used and 5 when the **request** keyword is used. The minimum value that can be specified is 1 (one). The command NO APPLETALK ARP or a *count* value of 0 resets the default.

[**no**] **appletalk checksum**

Enables and disables the generation and verification of checksums for all AppleTalk packets (except routed packets) when enabled. An incoming packet with a nonzero checksum will be verified against that checksum and discarded if in error. By default, checksum verification is enabled.

[**no**] **appletalk event-logging**

Causes logging of a subset of messages produced by DEBUG APPLETALK command. The **no** form of the command turns this function off. Logs significant events using the logger facility. Logged events include routing changes, zone creation, port status, and address.

appletalk iptalk-baseport *port-number*

Specifies the UDP port number, which is the beginning of the range of UDP ports used in mapping AppleTalk well-known DDP socket numbers to UDP ports. The argument *port-number* is the first UDP port number.

[**no**] **appletalk lookup-type** *serviceType*

Specifies services listed in SHOW APPLETALK NBP and SHOW APPLETALK NAME-CACHE EXEC command display. The argument *serviceType* is the specific AppleTalk service. The command NO APPLETALK LOOKUP-TYPE can be used with or without the *serviceType* argument. Using the argument specifies exclusion of a specific service type from the name cache. Prevent all names (except those relating to DECbrouter 90 routers) from being cached by using the **no** version of this command without the argument *serviceType*.

[**no**] **appletalk macip dynamic** *ip-address* [*ip-address*] **zone** *server-zone*

Allocates a single IP address or a range of IP addresses to be assigned to *dynamic* MacIP clients by the MacIP server serving zone *server-zone*. Dynamic clients are those who accept *any* IP address assignment within the dynamic range specified. The NO APPLETALK MACIP command shuts down all running MacIP services. If entered with the keyword **dynamic**, a specific *ip-address* range and a specific *server-zone*, the particular dynamic address assignment statement (if one exists) will be eliminated from the configuration.

[**no**] **appletalk macip server** *ip-address* **zone** *server-zone*

Establishes a new MacIP server. Only one MacIP server can be configured per AppleTalk zone. A server is not registered via NBP until at least one MacIP resource is configured. The NO APPLETALK MACIP command shuts down all active MacIP services. If entered with the keyword **server**, a specific *ip-address* and a specific *server-zone*, the particular server statement (if one exists) will be shut down and eliminated from the configuration.

[**no**] **appletalk macip static** *ip-address* [*ip-address*] **zone** *server-zone*

Defines a range of addresses to be made available to MacIP clients who have reserved an invariant IP address. The server keeps track of these address for routing and informational purposes. The NO APPLETALK MACIP command shuts down all running MacIP services. If entered with the keyword **static**, a specific *ip-address* and a specific *server-zone*, the particular static address assignment statement (if one exists) will be eliminated from the configuration.

[**no**] **appletalk name-lookup-interval** *intInSeconds*

Sets the interval between service pollings by the router on its AppleTalk interfaces. The argument *intInSeconds* is the interval in seconds between NBP lookup pollings. A value of zero (0) is equivalent to NO APPLETALK NAME-LOOKUP-INTERVAL. Both disable name lookup. The default is zero (0). You cannot disable lookup of **DigitalRouter**.

[**no**] **appletalk permit-partial-zones**

Allows access to zones that contain networks that do not have direct access. In other words, when a specific zone is *partially* obscured, other (visible) networks that are not subject to access control are propagated normally when **permit-partial-zones** is enabled. The default is for **appletalk permit-partial-zones** to be *disabled*. The NO APPLETALK PERMIT-PARTIAL-ZONES version of this command disables this option and restores the default condition where a complete zone is controlled if any associated network is controlled. If this command is enabled, networks for the zone are propagated, even if one or more networks are access-controlled.

[**no**] **appletalk proxy-nbp** *network-number zonename*

Required for each zone that has a nonextended-only AppleTalk router connected to a network in the zone. The argument *network-number* must be a unique network number that will be advertised via this router as if it were a real network. The argument *zonename* is the name of the zone requiring compatibility support. Only one proxy is needed to support a zone, but additional proxies can be defined with different network numbers if redundancy is desired The **no** version removes the specified network/zone association.

[**no**] **appletalk require-route-zones**

Prevents *bogus* routes (possibly generated by a broken router or corrupt packet) from causing ZIP protocol storms. The default is for **require-route-zones** to be *enabled*. When **require-route-zones** is enabled, the router will not advertise a route to its neighbors until it has obtained the network/zone associations. Use the NO APPLETALK REQUIRE-ROUTE-ZONES command to disable the **require-route zones** option and set the condition such that the router can advertise routes to its neighbors without having obtained the network-zone associations.

[**no**] **appletalk routing**

Enables or disables the AppleTalk protocol processing.

[**no**] **appletalk strict-rtmp**

Enforces maximum checking of routing packets to ensure their validity. The default of this command is to provide maximum checking. The **no** variation disables the maximum checking mode.

[**no**] **appletalk timers** *update-interval valid-interval invalid-interval*

Changes the time intervals (in seconds) used in AppleTalk routing. The argument *update-interval* is the time between routing updates sent to other routers on the network; the default is 10 seconds. The argument *valid-interval* is amount of time that the router will consider a route valid without having heard a routing update for that route; the default is 20 seconds, and the value is normally twice the update interval. The argument *invalid-interval* is the amount of time that the router will wait before marking a route invalid; the default is three times the *valid-interval*, or 60 seconds.

AppleTalk Interface Subcommand Summary

This section lists, in alphabetical order, all the interface subcommands used with AppleTalk networks.

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

Assigns an interface to an access list. The argument *access-list-number* specifies the appropriate AppleTalk access list. Use the **no** form of the command to remove the list from the interface.

[**no**] **appletalk address** *address*

Assigns AppleTalk addresses on the interfaces that will be used for the AppleTalk protocol. Use this command prior to assigning zone names. Use this subcommand to configure nonextended interfaces. The **no** version removes the specified address.

[**no**] **appletalk cable-range** *start-end* [*network.node*]

Designates an interface as being on an extended AppleTalk network. This range is specified using the *start-end parameter*, which is a pair of decimal numbers between 1 and 65279, inclusive. The starting and ending addresses should usually be assigned equal numbers. The optional *network.node* argument specifies the suggested network and node number that will be used first when selecting the AppleTalk address for this interface. The **no** version removes the specified cable range.

[**no**] **appletalk discovery**

Resets the discovery mode and allows a new cable range to be discovered. Use the **no** variation to return the software to the default (off) state.

[**no**] **appletalk distribute-list** *access-list-number* **in**

Filters input from networks. The argument *access-list-number* is the number of a predefined access list. The keyword **in** is used to filter networks received in update. The **no** version removes the specified distribution list.

[**no**] **appletalk distribute-list** *access-list-number* **out**

Filters output from networks. The argument *access-list-number* is the number of a predefined access list. The keyword **out** is used to suppress networks from being sent in updates. The **no** version removes the specified distribution list.

[**no**] **appletalk getzonelist-filter** *access-list-number*

Modifies zone-list replies. The argument *access-list-number* must be in the range of 600 to 699, inclusive. If an undefined access list is used, the rule defaults to **permit**. If a zone does not match any rule in the list, it is denied, unless permitted through the additional-zones option of the ACCESS-LIST global configuration command. Use the NO APPLETALK GETZONELIST *access-list-number* command to remove this filter. Numeric entries in the access list are ignored by this filter.

appletalk iptalk *net.node zone*

Encapsulates AppleTalk in IP packets in a manner compatible with the Columbia AppleTalk Package (CAP) IPTalk and the Kinetics IPTalk (KIP) implementations. This command enables IPTalk encapsulation on an interface that already has a configured IP address. The argument *net.node* is a network node address; the argument *zone* is the AppleTalk zone.

[**no**] **appletalk send-rtmp**

Allows a router to be placed on a net with AppleTalk enabled, but without being seen. This allows disabling of routing update. The default is to send updates. The **no** version blocks updates from being sent.

[**no**] **appletalk zone** *zonename*

Sets the zone name for the connected AppleTalk network. This command also specifies the zone name associated with the AppleTalk network for the specified interface. The argument *zonename* specifies the name of the zone for the connected AppleTalk network. The argument is ignored for nonextended AppleTalk. The command is ignored if the specified zone name is not in the zone list. The **no** form of the command deletes a zone name from a zone list or the entire zone list if none is specified. Must be specified after the APPLETALK ADDRESS or APPLETALK CABLE-RANGE command if discovery is not enabled. This command can be issued multiple times if it follows the APPLETALK CABLE-RANGE command.

3
Routing CHAOSnet

This chapter describes the DECbrouter 90 implementation of the CHAOSnet routing protocol.

DECbrouter 90 Implementation of CHAOSnet

CHAOSnet is a local area network protocol developed at the Massachusetts Institute of Technology in the mid-1970s. Several artificial intelligence workstation manufacturers use the CHAOSnet protocol in their networking software products. The DECbrouter 90 router supports full CHAOSnet routing and a small subset of CHAOSnet host functions, including the status, uptime, and dump routing table services. The router can route CHAOSnet packets over Ethernets and synchronous serial lines.

CHAOSnet Addresses

CHAOSnet addresses are 16-bit quantities written as octal numbers. The higher-order eight bits of the address are the CHAOSnet network number, and the lower-order eight bits are the host number. Following is an example of a CHAOSnet address:

```
315.124
```

The DECbrouter 90 CHAOSnet implementation assumes that the CHAOSnet network corresponds one-to-one with a subnetted Internet network. For example, CHAOSnet subnet 1 must correspond to Internet subnet 1, and CHAOSnet host 360 (octal) must correspond to Internet host 240 (decimal). Because a CHAOSnet internetwork built with DECbrouter 90 routers uses network addresses from a single underlying Internet network, the router does not route CHAOSnet packets from one Internet network to another.

To form a CHAOSnet address, the router combines the lower eight or fewer bits of the Internet subnet field with the lower eight or fewer bits of the Internet host field. This approach does not assume any particular class of Internet address or subnetting scheme. However, Digital recommends that at least eight bits of subnet identifier and eight bits of host identifier be used.

Configuring CHAOSnet Routing

To start the CHAOSnet router process, use the ROUTER CHAOS global configuration command. The command syntax is as follows:

router chaos
no router chaos

The router process routes CHAOSnet packets and sends CHAOSnet routing updates. The NO ROUTER chaos command disables CHAOSnet routing.

Example

The following commands start CHAOSnet routing on network 128.88.0.0:

```
router chaos
network 128.88.0.0
```

CHAOSnet routing does not replace Internet routing; an Internet routing protocol, such as RIP, must run concurrently with CHAOSnet routing on the router. To ensure routing table consistency, the Internet routing protocol must have a greater administrative distance than the CHAOSnet routing protocol. In addition, you must configure the CHAOSnet routing process to readvertise subnet routes derived from the Internet routing protocol.

Continuing the previous example, suppose RIP is the routing protocol running concurrently with CHAOSnet. The following router subcommand advertises RIP-derived routes to the CHAOSnet hosts on the network:

```
redistribute rip
```

See the section Redistributing Routing Information in Chapter 6 of this manual for more information on protocol-independent routing issues such as administrative distance and redistribution.

Monitoring CHAOSnet

Use the EXEC commands described in this section to monitor activity on the CHAOSnet.

show chaos-arp

The command SHOW CHAOS-ARP displays CHAOSnet-specific ARP entries as 16-bit octal addresses.

show ip route

The command SHOW IP ROUTE displays routing entries obtained from the CHAOSnet routing protocol. In the command output, CHAOSnet entries are marked by X in the first column.

show ip traffic

The command SHOW IP TRAFFIC displays statistics on CHAOSnet protocol operation.

Debugging CHAOSnet

Use the EXEC commands described in this section to display reports of problems and activity on the CHAOSnet.

debug chaos-routing

The command DEBUG CHAOS-ROUTING enables logging of CHAOSnet routing activity, including service requests.

debug chaos-packet

The command DEBUG CHAOS-PACKET enables logging of CHAOSnet packet transactions.

4

Routing DECnet

This chapter describes Digital's implementation of DECnet Phase IV for the DECbrouter 90 product line. Topics and tasks described in this chapter include:

- The DECbrouter 90 implementation of DECnet
- An overview of DECnet routing and addressing
- How to enable DECnet routing
- How to configure interarea and intra-area routing costs
- How to configure access lists
- How to set default routers and priority values
- How to fine-tune DECnet performance parameters, including fast switching

DECnet Phase V is equivalent to ISO CLNS. Support for DECnet Phase IV/Phase V conversion is discussed in this chapter.

The DECbrouter 90 Implementation of DECnet

Digital designed the DECnet stack of protocols in the 1970s as part of its Digital network architecture (DNA). DECnet support on a DECbrouter 90 includes local area and wide area DECnet Phase IV routing over Ethernet and serial lines as follows:

- The router uses HDLC framing rather than Digital's DDCMP framing for point-to-point lines. If you construct a network using both DECbrouter 90 and other Digital equipment, you must ensure that each point-to-point line has the same type of equipment on both ends.
- The DECbrouter 90 and DECnet Phase IV routers have incompatible X.25 support. As with point-to-point lines, you must use homogeneous equipment on the X.25 portion of your network.
- The DECbrouter 90 gives you additional security options through access lists.

The DECbrouter 90 can support the address translation gateway (ATG), which allows the router to participate in multiple, independent DECnet networks and to establish a user-specified address translation table for selected nodes between networks.

Digital uses some nonroutable protocols that are not part of the DECnet stack. Protocols such as MOP (discussed later in this chapter) and LAT, the Digital terminal server, cannot be routed. These protocols must be bridged.

DECnet Phase IV Addresses

DECnet Phase IV addresses are specified by an area number and a node number separated by a period. For example, 53.6 is area 53, node 6.

DECnet hosts exist as a *node* (host) in an *area*. Do not confuse the concept of *area* with an area defined by the IP, XNS, or other routing protocols. Unlike these protocols, DECnet allows for an area to span many routers, and for a single cable to have many areas attached to it. Therefore, if a host (such as a router) exists on many cables, it uses the same area/node for itself on all of them. Note how this differs from other routing protocols where each interface is given a different internetwork address.

DECnet hosts do not use manufacturer-assigned MAC layer addresses. Instead, network level addresses are embedded in the MAC layer address according to the formula that follows.

The area number is six bits long (1 through 63); the node number is 10 bits long (1 through 1023). To derive a MAC address from a DECnet node number, convert the dotted decimal address into a 16-bit number using the formula: (1024 * area) + node. This 16-bit address is appended to the address AA00.0400 in byte-swapped order, with the least significant byte first. The following example illustrates how to convert the DECnet address 12.75:

12 * 1024 + 75 = 12363 (base 10) = 304B (base 16 or hex notation)

The resulting MAC address is AA00.0400.4B30.

You also can use the EXEC SHOW INTERFACES command to obtain the MAC address once DECnet routing is enabled.

- The DECnet Phase IV protocol associates addresses with machines, not interfaces. Therefore, a router can have only one DECnet Phase IV address for each DECnet network in which it participates. A DECnet MAC-level address is simply an encoded version of the 16-bit area/node combination. This explains why Ethernet interface addresses change on a DECbrouter 90 when the DECnet protocol is enabled.
- DECnet does not have the equivalent of the IP address resolution protocol (ARP); DECnet hosts simply advertise their presence with periodic hello packets. Routers build local routing tables and hosts learn each other's addresses by listening to host hello messages. Hosts learn about nearby routers by listening to router hello messages.

You do not have to set each interface address manually; the DECnet ROUTING global configuration command automatically assigns an address to each interface for which you entered a DECnet COST configuration command. (These commands are described later in this chapter.)

The parameters in the DECbrouter 90 implementation of DECnet are a subset of the parameters you can modify in Digital's network control program (NCP). The DECbrouter 90 uses the same names, the same range of allowable values, and the same defaults wherever possible. Note that you must use the configuration commands to set DECnet parameters; the DECbrouter 90 DECnet implementation does not set parameters by communicating with NCP.

Configuring DECnet Routing

Follow these steps to start configuring your router for DECnet routing:

1. Enable DECnet routing and specify system-wide host addresses; use the DECnet ROUTING global configuration command.
2. After DECnet routing has been enabled, a cost must be assigned to each interface over which DECnet should run. This enables the interface. DECnet nodes route toward a destination using the lowest path cost, so you should base your cost values on interface throughput. Use the DECnet COST interface subcommand to set a cost value for an interface.
3. Next, specify the node type with the DECnet NODE-TYPE command. It will be either an area router—Level 1 and Level 2—or a local router routing DECnet Phase IV at Level 1 only.
4. You can alter the maximum node number and maximum area number with the optional DECnet MAX-ADDRESS and DECnet MAX-AREA commands.
5. Finally, you must specify several commands for either intra-area or interarea routing. These commands and their parameters must be chosen carefully, because in many cases, they are dependent on each other's values.

The following sections take you through these steps in detail, as well as all the optional commands for managing performance, security, and Phase IV /V conversion. The section DECnet Configuration Examples shows complete configuration examples for many common situations.

Enabling DECnet Routing

To enable or disable DECnet routing, use the DECnet ROUTING global configuration command:

DECnet routing *decnet-address*
no DECnet routing

The argument *decnet-address* takes as its value an address in DECnet format X.Y, where X is the area number and Y is the node number. There is no default router address; you must specify this parameter for DECnet operation.

Example

In this example, DECnet routing is enabled for the router in area 21 with node number 456:

```
decnet routing 21.456
```

Assigning the Cost

After DECnet routing has been enabled, you must assign a cost to each interface over which you want DECnet to run. (Assigning a cost in effect enables DECnet routing for an interface.) Most DECnet installations have an individualized routing strategy for using costs. Therefore, check the routing strategy used at your installation to ensure that costs you specify are consistent with those set for other hosts on the network.

The DECNET COST interface subcommand sets a cost value for an interface:

decnet cost *cost-value*
no decnet cost

The argument *cost-value* is an integer from 1 to 63. There is no default cost for an interface, although suggested costs are 4 for Ethernet and greater than 10 for serial links. Use the NO DECNET COST subcommand to disable DECnet routing for an interface.

Example

The following example establishes a DECnet routing process for the router at 21.456, then sets a cost of four for the Ethernet 0 interface:

```
decnet routing 21.456
interface ethernet 0
decnet cost 4
```

Specifying the Node Type

Before you use many of the global and interface configuration commands, you must specify the node type with the DECNET NODE-TYPE global configuration command. The DECNET NODE-TYPE command specifies the node type for the router.

decnet node-type {**area** | **routing-iv**}

The options are either **area** or **routing-iv**. If you specify **area**, the router participates in the DECnet routing protocol with other area routers, as described in the Digital documentation, and routes packets to and from routers in other areas. This is sometimes referred to as Level 2, or interarea, routing. An area router does not just handle interarea routing; it also acts as an intra-area or Level 1 router. If you specify **routing-iv** (the default), the router acts as an intra-area (standard DECnet Phase IV, Level 1 router) and ignores Level 2 routing packets. In this mode, it routes packets destined for other areas via the least-cost path to an interarea router, exchanging packets with other end nodes and routers in the same area.

Specifying Node Numbers and Area Sizes

DECnet routers do not have the concept of aging out a route. Therefore, all possible areas or nodes must be advertised as unreachable if they cannot be reached. Since it is best to keep routing updates small, you need to indicate the default maximum possible node and area numbers that can exist in the network. The default value for a node address to be given in an update is 1023.

You can use the DECNET MAX-ADDRESS global configuration command to configure the router with a different maximum node address, as follows:

decnet max-address *value*

The argument *value* is a number, less than or equal to 1023, that represents the maximum node address possible on the network. In general, all routers on the network should use the same value for this parameter.

Example

The example that follows configures a small network (spanning just a department). The desire is to keep routing updates as small as possible, so the maximum address value is set to 300 instead of the default of 1023.

```
!
decnet max-address 300
!
```

Use DECNET MAX-AREA global configuration command to set the largest number of areas that the router can handle in its routing table. The syntax is as follows:

decnet max-area *value*

The argument *value* is an area number from 1 to 63; the default is 63. Like the DECNET MAX-ADDRESS command value, this parameter controls the sizes of internal routing tables and of messages sent to other nodes. All routers on the network should use the same maximum address value.

Example

In this example, the maximum number of areas that the router will save in its routing table is 45.

```
!
decnet max-area 45
!
```

Specifying the Maximum Route Cost for Interarea Routing

The DECNET AREA-MAX-COST global configuration command sets the maximum cost specification value for *interarea* routing. The syntax of this command follows.

decnet area-max-cost *value*

The argument *value* determines the maximum cost for a route to a distant area that the router may consider usable; the router treats as unreachable any route with a cost greater than the value you specify. A valid range for cost is from 1 to 1022; the default is 1022. This parameter is only valid for area routers. Make sure you have used the DECNET NODE-TYPE AREA command before using this command.

Example

In this example, the node type is specified as area and the maximum cost is set to 500. Any route whose cost exceeds 500 will be considered unreachable by this router.

```
!
decnet node-type area
decnet area-max-cost 500
!
```

Use the DECNET AREA-MAX-HOPS global configuration command to set the maximum hop count value for *interarea* routing as follows:

decnet area-max-hops *value*

The argument *value* determines the maximum number of hops for a usable route to a distant area. The router treats as unreachable any route with a count greater than the value you specify. A valid range for the hop count is from 1 to 30; the default is 30. This parameter is only valid for area routers. Make sure you have used the DECNET NODE-TYPE AREA command before using this command.

Example

This example sets the node type to area, then sets a maximum hop count of 21. This was done because it is a small network with relatively few routers for interarea routing, so a route with a large hop count is liable to represent a problem, not an efficient route.

```
!
decnet node-type area
decnet area-max-hops 21
!
```

Specifying the Maximum Route Cost for Intra-area Routing

The DECNET MAX-COST global configuration command sets the maximum cost specification for *intra-area* routing. The router ignores routes within the router's local area that have a cost greater than the corresponding value of this parameter. The syntax for this command follows.

decnet max-cost *value*

The argument *value* is a cost from 1 to 1022 (the default).

Example

In this example, the node type is specified as DECnet Phase IV and the maximum cost is set to 335. Any route whose cost exceeds 335 will be considered unreachable by this router.

```
!
decnet node-type routing-iv
decnet max-cost 335
!
```

Use the DECNET MAX-HOPS global configuration command to set the maximum hop count specification value for *intra-area* routing, as follows:

decnet max-hops *value*

The argument *value* is a hop count from 1 to 30 (the default). The router ignores routes that have a hop count greater than the corresponding value of this parameter.

Example

This example sets the node type to DECnet Phase IV routing, then sets a maximum hop count of 2.

```
!
decnet node-type routing-iv
decnet max-hops 2
!
```

Configuring Maximum Visits

Use the decnet MAX-VISITS global configuration command to set the limit on the number of times a packet can pass through a router.

decnet max-visits *value*

The argument *value* can vary from 1 to 63 (the default). If a packet exceeds value, the router discards the packet. Digital recommends that the value of the max-visits parameter be at least twice that of the max-hops parameter, to allow packets to still reach their destinations when routes are changing.

Example

This example of intra-area routing configuration specifies Phase IV routing, a maximum hop count of 28, and maximum number of visits of 62 (which is more than twice 28).

```
!
decnet node-type routing-iv
decnet max-hops 28
decnet max-visits 62
!
```

Configuring Path Selection

Limiting the number of *equal cost* paths can save memory on routers with limited memory or very large configurations. Additionally, in networks with a large number of multiple paths and end-systems with limited ability to cache out-of-sequence packets, performance may suffer when traffic is split between many paths.

Limiting the size of the routing table will not affect your router's ability to recover from network failures transparently, provided that you do not make the maximum number of paths too small. If more than the specified number of equal cost paths exist, and one of those paths suddenly becomes unusable, the router will discover an additional path from the paths it has been ignoring.

The first of the optional path global configuration commands, DECNET MAX-PATHS, defines the maximum number of equal cost paths to a destination that the router will keep in its routing table, with the following syntax:

decnet max-paths *value*

The argument *value* is a decimal number equal to the maximum number of equal cost paths the router will save. The highest value accepted is 31; the default value is 1.

Example

In the following example, some destinations have six equal cost paths, so the example specifies that the router will save no more than three equal cost paths.

```
!
decnet max-paths 3
!
```

The DECNET PATH-SPLIT-MODE global configuration command also helps you make decisions about equal cost paths; it specifies how the router will split the routable packets between equal cost paths. This command has two forms, as shown:

decnet path-split-mode normal
decnet path-split-mode interim

The keyword **normal** selects normal mode (the default), where equal cost paths are selected on a round-robin basis. The keyword **interim** specifies that traffic for any particular (higher-layer) session is always routed over the same path. This mode supports older implementations of DECnet (VMS Versions 4.5 and earlier) that do not support out-of-order packet caching. Other sessions may take another path, thus using equal cost paths that a router may have for a particular destination.

Altering DECnet Defaults

In general, you need not modify the DECnet parameters. However, under special circumstances or when using a specific configuration, you will see better performance if you alter some of the default parameters. This section will guide you through those special circumstances.

Adjusting Timers and the Route Cache

The router broadcasts hello messages on all interfaces with DECnet enabled. Other hosts on the network use the hello messages to identify the hosts with which they can communicate directly. The router sends hello messages every 15 seconds by default. On extremely slow serial lines, you may want to increase this value to reduce overhead on the line using the DECNET HELLO-TIMER interface subcommand.

decnet hello-timer *value*
no decnet hello-timer

The argument *value* varies from 1 to 8191 seconds; the default is 15 seconds.

Example

The following example increases the hello interval to 2 minutes (120 seconds) on interface serial 1.

```
interface serial 1
decnet hello-timer 120
```

By default, the DECbrouter 90 DECnet routing software implements fast switching of DECnet datagrams. There are times when it makes sense to disable fast switching. This is especially important when using rates slower than T1.

Fast switching uses memory space interface cards. In situations where a high-bandwidth interface is writing large amounts of information to a low-bandwidth interface, additional memory could help avoid congestion on the slow interface (also known as big-pipe/little-pipe problems). Use the NO DECNET ROUTE-CACHE interface subcommand to turn off fast switching.

decnet route-cache
no decnet route-cache

In a network where changes occur infrequently or do not need to be responded to immediately (it is small and uncomplicated, applications are not particularly sensitive to delays or occasional packet loss, or slow serial links), increasing the time between routing updates reduces the amount of unnecessary network traffic. The DECNET ROUTING-TIMER INTERFACE subcommand specifies how often the router sends routing updates that list all the hosts that the router can reach. Other routers use this information to construct local routing tables. Digital usually calls this parameter the *broadcast routing timer* because they use a different timer for serial lines; the DECbrouter 90 DECnet implementation does not make this distinction. The syntax for the DECNET ROUTING-TIMER interface subcommand follows:

decnet routing-timer *value*
no decnet routing-timer

The argument *value* specifies a time from 1 to 65535 seconds; the default is 40 seconds. The NO DECNET ROUTING-TIMER command restores this default.

Example

In the following example, a serial interface is set to broadcast routing updates every two minutes.

```
interface serial 0
decnet routing-timer 120
```

Specifying the Designated Router

The *designated* router is the router to which all end nodes on an Ethernet communicate if they do not know where else to send a packet. The designated router is chosen through an election process in which the router with the highest priority gets the job. When two or more routers on a single Ethernet in a single area share the same highest priority, the unit with the highest node number is elected. You can reset a router's priority to help ensure that it is elected designated router in its area.

Priority can be changed with the DECNET ROUTER-PRIORITY interface subcommand, as follows:

decnet router-priority *value*

The argument *value* can range from zero through 127; the default priority is 64.

Example

In the following example, interface Ethernet 1 is set to a priority of 110.

```
!
interface ethernet 1
decnet router-priority 110
!
```

Managing Traffic Using DECnet Access Lists

There are two forms of DECnet access lists: one that specifies a single address (a standard list) and one that specifies two addresses (an extended list). See the section Configuring IP Access Lists in Chapter 5 for general information about setting up access lists.

Configuring DECnet Access Lists

Use the ACCESS-LIST global configuration command to create an access list.

access-list *list* {**permit** | **deny**} *destination destination-mask*
no access-list *list*

The argument *list* is an integer you choose between 300 and 399 that uniquely identifies the access list. The **permit** and **deny** keywords decide the access control action when a match happens with the address arguments.

The standard form of the DECnet access list has a DECnet *destination* followed by a *destination-mask*, also in DECnet address format, with bits set wherever the corresponding bits in the address should be ignored. DECnet addresses are written in the form *area.node* (for example, 50.4 is area 50, node 4). All addresses and masks are in decimal.

Note

In contrast with IP masks, a DECnet mask specification of "all ones" is entered as the decimal value 1023. In IP, the equivalent is 255.

Example

This example sets up access list 300 to deny packets going out to the destination node 4.51 and permit packets destined to 2.31.

```
!
access-list 300 deny 4.51 0.0
access-list 300 permit 2.31 0.0
!
```

Configuring Extended Access Lists

Use this global configuration command to create extended access lists:

access-list *list* {**permit** | **deny**} *source source-mask destination destination-mask*
no access-list *list*

The extended form of the DECnet access list has a source DECnet address and mask pair, followed by a destination DECnet address and mask pair.

The argument *list* is an integer you choose between 300 and 399 that uniquely identifies the access list. The **permit** and **deny** keywords decide the access control action when a match happens with the address arguments.

Example

In this example, access list 301 is configured to allow traffic from any host in networks 1 and 3. It implies no other traffic will be permitted. (The end of a list contains an implicit "deny all else" statement.)

```
!
access-list 301 permit 1.0 0.1023 0.0 63.1023
access-list 301 permit 3.0 0.1023 0.0 63.1023
!
```

DECnet Connect Initiate Filtering

DECnet access lists can be used to filter *connect initiate* packets. This means that you can filter by DECnet object type, such as MAIL. The syntax for the connect initiate filter version of DECnet access lists follows.

access-list *list* {**permit** | **deny**} *source source-mask* [*destination destination-mask*] [*connect-entries*]
no access-list *list*

The argument *list* is the access list number in the range 300–399.

The argument *pair source source-mask* is the source address and mask.

The argument *pair destination destination-mask* are the optional destination address and mask.

The *connect-entries* are the optional entries used to match connect packets. These entries are as follows:

{**eq** | **neq**} [*src-object*] [*dst-object*] [*identification*]
eq any

For the **eq** | **neq** option:

- If **eq** is specified, the item matches the packet if all the specified parts of *src-object*, *dst-object*, and *identification* match data in the packet.
- If **neq** is specified, the item matches the packet if *any* of the specified parts do not match the corresponding entry in the packet.

The argument *src-object* consist of two parts:

- The keyword **src**
- The argument *obj-speckeyword src*

The argument *obj-spec* can be one of the following:

— *relop object-number*

— **exp** *regex*

— **uic** [*group,user*]—In this case the bracket symbols are literal; they must be entered.

The argument [*group, user*] is a numeric UID expression. The group and user parts can either be specified in decimal, in octal by prefixing the number with a 0, or in hex by prefixing the number with 0x. The **uic** expression is displayed in **show** displays as an octal number.

The argument *relop* can be one of the following relational operator keywords:

— **eq**—equal to

— **neq**—not equal to

— **lt**—less than

— **gt**—greater than

The argument *object-number* is a numeric DECnet object number.

The argument *regex* is a regular expression that matches a string.

Note

Regular expressions are described in Appendix D of this manual.

The argument *dst-object* consists of two parts:

- The keyword **dst**
- The argument *obj-spec* (described previously)

The argument identification can include any of the following:

- **id** *regex*
- **password** *regex*
- **account** *regex*

The argument *regex* is a regular expression that matches a string.

Connect Initiate Filter Configuration Considerations

The *obj-spec* can be one of three formats.

- In the first format, you specify a relational operator and an object number. For example:
  — code_example(eq 17)—Equal to 17
  — code_example(gt 128)—Greater than 128

- The second format is a regular expression that matches a string. For example:
  - `exp ^SYSTEM$`—This expression exactly matches the string SYSTEM.
  - `exp USER`—This expression matches any string containing the substrate USER.
- The third format matches a UIC. For example:
  - `uic [1,4]`

The second and third formats can be used separately or combined. If specified separately, they might appear as follows:

- `src uic [1,4]`
- `src exp USERNAME`

If combined, the *obj-spec* might appear as:

- `src exp USERNAME uic [1,4]`

The **id**, **password**, and **account** keywords all take regular expressions as arguments and match access information in the packet.

The special format **eq any** matches any connect packet.

Connect Initiate Filtering Examples

The following examples illustrate specification of access lists for connect initiate packet filtering.

Example 1: Match Object Number

The following example illustrates an access list for matching all connect packets for object number 27:

```
access-list 300 permit 0.0 63.1023 eq dst eq 27
```

Example 2: Match Connect Packets Except Object Number

The following example illustrates an access list for matching all connect packets *except* for the object number 17:

```
access-list 300 permit 0.0 63.1023 neq dst eq 17
```

Example 3: Match Connect Packets for Access ID

The following example illustrates an access list for matching all connect packets where the access identification was SYSTEM:

```
access-list 300 permit 0.0 63.1023 eq id ^SYSTEM$
```

Example 4: Match Connect Packets from Area 1

The following example illustrates an access list for matching all connect packets from area 1 to object number 27 where SYSTEM is the originating user:

```
access-list 300 permit 1.0 0.1023 eq src exp ^SYSTEM$ dst eq 27
```

Example 5: Match Any Connect Packet

The following example illustrates an access list for matching any connect packet:

```
access-list 300 permit 0.0 63.1023 eq any
```

Note

The configuration specification in Example 5 can be used at the end of a list to permit any packets not already matched.

When building a DECnet access list, you can consider a list as having two parts:

- The first part, which contains none of the optional connect matching codes, filters all packets, including data packets and connect initiate packets. When using an access list, all packets, including connect packets, must match the first part of the list. If you only want to filter traffic based on connect packets, use an access list command of the following form as the first entry in your list (it matches and permits all packets):

  ```
  access-list 300 permit 0.0 63.1023
  ```

- The second part of the list consists of entries that contain the optional codes to match connect initiate packets. If there are any items of this type in your access list, all connect initiate packets must *match* the specification and be permitted by an item in your list, otherwise the connect initiate packet will be rejected.

Configuring Access Groups

The DECNET ACCESS-GROUP interface subcommand applies an access list to an interface.

decnet access-group *list*

The argument *list* can be either a standard or extended DECnet access list. A standard DECnet access list applies to destination addresses in this case.

Example

The following example applies access list 389 to interface Ethernet 0.

```
!
interface ethernet 0
decnet access-group 389
!
```

Configuring In- and Out-Routing Filters

The DECNET IN-ROUTING-FILTER interface subcommand provides access control to hello messages or routing information received on this interface. Addresses that fail this test are treated as unreachable. The full syntax of the command follows.

decnet in-routing-filter *list*
no decnet in-routing-filter

The argument *list* is a standard DECnet access list.

The NO DECNET IN-ROUTING-FILTER command removes access control.

Example

In the following example, interface Ethernet 0 is set up with a DECnet in-routing filter of 321, which means that any hello messages sent from addresses that are denied in list 321 will be ignored. Additionally, all node addresses listed in received routing messages on this interface will be checked against the access list, and only routes passing the filter will be considered usable.

```
!
interface ethernet 0
decnet in-routing-filter 321
!
```

The DECNET OUT-ROUTING-FILTER interface subcommand provides access control to routing information being sent out on this interface. Addresses that fail this test are shown in the update message as unreachable.

decnet out-routing-filter *list*
no decnet out-routing-filter

The argument *list* is a standard DECnet access list.

The NO DECNET OUT-ROUTING-FILTER command removes access control.

Example

In the following example, interface Ethernet 0 is set up with a DECnet out-routing filter of 351. This filter is applied to addresses in the transmitted routing updates. Transmitted hello messages are not filtered.

```
!
interface ethernet 0
decnet out-routing-filter 351
!
```

DECnet Phase IV-to Phase-V Conversion

DECnet Phase V is OSI-compatible and conforms to the ISO 8473 (CLNP/CLNS) and ISO 9542 (ES-IS) standards. See the *DECbrouter 90 Configuration and Reference, Volume 3* for an explanation of configuring OSI CLNP routing and for a review of the terminology.

Digital has defined algorithms for mapping a subset of the Phase V address space onto the Phase IV address space and for converting Phase IV and Phase V packets back and forth. This allows a network administrator to support both Phase IV hosts in Phase V networks and Phase V hosts in Phase IV networks.

The algorithms defined by Digital perform the following tasks:

- Conversion between Phase IV and Phase V addresses
- Conversion between Phase V and Phase IV addresses
- Advertisement of Phase IV reachability in a Phase V network
- Advertisement of Phase V reachability in a Phase IV network
- Determination of when to perform the packet conversion

Note

Refer to the *DECbrouter 90 Configuration and Reference, Volume 3* for details about DECnet Phase V cluster alias support.

The DECbrouter 90 implementation differs from other Digital products in how reachability information is advertised. The DECbrouter 90 implementation allows you to add Phase V support without modifying your existing Phase IV support. It also delays converting packets from Phase IV to Phase V, while Digital's other implementations convert as soon as possible.

To enable DECnet conversion, you must configure both DECnet and ISO CLNS on your router. In addition, you must turn on conversion with the DECNET CONVERSION global configuration command. The command syntax is as follows:

decnet conversion *NSAP-prefix*
no decnet conversion

The argument *NSAP-prefix* defines the value used for the IDP when constructing NSAPs from a Phase IV address.

The command NO DECNET CONVERSION disables Phase IV-to Phase-V conversion for the router.

Example

To enable DECnet conversion on a DECbrouter 90 router with the area tag foo and Phase IV address 20.401 using an ISO IGRP router, enter the following configuration commands:

```
!
clns routing
decnet routing 20.401
decnet max-address 600
!
router iso-igrp foo
net 47.0004.004d.0014.aa00.0400.9151.00
!
decnet conversion 47.0004.004d
!
interface ethernet 0
decnet cost 4
clns router iso-igrp foo
!
```

It is essential that the area specified in the DECNET ROUTING command be the same as the local area specified on the NET command.

Note

The DECNET ROUTING command is specified with a decimal address, while the NET command address is specified in hex. In addition, the *NSAP-prefix* specified on the DECNET CONVERSION command must match one of the NETs for this router.

Designing a Network to Support Both Phase IV and Phase V

Digital Phase V hosts can use either Phase IV or Phase V packet format. A Digital Phase V host chooses the format to use based on the type of router hello packets that it sees.

DECbrouter 90 routers with conversion enabled advertise reachability to both Phase IV hosts and Phase V hosts in both Phase IV and Phase V routing updates.

DECbrouter 90 routers always attempt to deliver packets in their native format *first*. However, if a Phase IV packet arrives at a router with conversion turned on and the router does not have a Phase IV path to the destination address, the router will convert the Phase IV address to Phase V and look in the Phase V routing table. If a path is found there, the packet will be converted to Phase V and delivered.

If a Phase V packet arrives at a router with conversion turned on and the router does not have a Phase V path to the destination address, the router will convert the Phase V address to Phase IV and look in the Phase IV routing table. If a path is found there, the packets will be converted to Phase IV and delivered.

In addition, all packets to a Phase IV host will be delivered in Phase IV format.

The following guidelines should help you design a network that simultaneously supports DECnet Phase IV and Phase V:

- Host connectivity across multiple areas is only possible if a Level 2 path exists for which every Level 2 router in the path supports a common protocol: Phase IV or Phase V. If not all routers support both protocols, those routers that do *must* have conversion enabled.

- Host connectivity across a single area is only possible if a Level 1 path exists for which every Level 1 router in the path supports a common protocol: Phase IV or Phase V. If not all routers support both protocols, those routers that do must *have* conversion enabled.
- The Level 2 backbone *must* have conversion enabled in all Level 2 routers supporting an area that needs conversion.

DECnet Configuration Examples

This section includes configuration examples showing many common DECnet configuration activities.

Establishing Routing; Setting Interfaces; Maximum Address Space

The configuration subcommands in the example that follows establish DECnet routing on a DECbrouter 90. The first line establishes DECnet routing for a specific address. The second line sets the maximum address space at 1023 addresses. The second section sets a cost of four for the Ethernet 0 interface. The third section sets a cost of ten for the serial 1 interface.

Example

```
!
decnet routing 4.27
decnet max-address 1023
interface ethernet 0
decnet cost 4
interface serial 1
decnet cost 10
!
```

Level 1 and Level 2 Routing; Designated Router

In the first part of this configuration, the router is being set up with an area and node address in the first line, then it is being designated a Level 2 (area) router. In the lines that follow, the two serial interfaces are given costs of four.

Example 1

```
!
decnet routing 6.10
decnet node area
!
interface serial 0
decnet cost 4
interface serial 1
decnet cost 4
!
```

To ensure that a specific router is elected the designated router, assign it the highest possible net address and give it a high router priority as shown in the next example.

Example 2

```
!
decnet routing 6.1023
decnet node area
!
interface ethernet 0
decnet cost 4
decnet router-priority 127
!
interface serial 0
decnet cost 20
!
```

In the third example, the router is a Level 1 router in area 7. The serial links are slower than in the previous example (9.6 vs 56 kbps), so they have a higher cost.

Example 3

```
!
decnet routing 7.12
decnet node routing-iv
!
interface ethernet 0
decnet cost 4
interface serial 1
decnet cost 25
interface serial 0
decnet cost 25
!
```

Phase IV to Phase V Conversion

This example begins by enabling DECnet routing with a specific address of 54.6. It then specifies the area with the name *Field* (as in Field Offices) with the ROUTER ISO-IGRP command. Specification of the ISO IGRP routing process is followed by specification of the NET command, which assigns an address to the routing process.

At this point you have set the stage for the DECNET CONVERSION command, which specifies the NSAP prefix address to be used when converting Phase IV addresses to Phase V.

After you have enabled the conversion, you need to name the specific interfaces that you want to route DECnet packets. In this example, the interface Ethernet 0 is enabled, with a cost of ten. The CLNS ROUTER ISO-IGRP command with the *Field* area is needed to specify that the interface will be using ISO-IGRP and Phase V CLNS and that it is part of area *Field*.

You could follow this interface specification with other interface specifications such as Ethernet 0, serial 0, and so on, with the same three commands. You also could go on to specify access lists and other special commands for these specific interfaces.

Example

```
!
decnet routing 54.6
clns routing
router iso-igrp Field
net 47.0006.0200.0000.0000.0100.0036.AA00.0400.06D8.00
decnet conversion 47.0006.0200.0000.0000.0100
interface ethernet 0
decnet cost 10
clns router iso-igrp Field
!
```

Note

Make sure that the area you specify in the DECNET CONVERSION command (54) is the same as the area you specified for the CLNS network (36). Also note that the DECnet area is specified in *decimal*, and the CLNS area is specified in *hexadecimal*.

The Address Translation Gateway

The address translation gateway (ATG) allows a DECbrouter 90 to route traffic for multiple independent DECnet networks and to establish a user-specified address translation for selected nodes between networks. This allows connectivity between DECnet networks that might be otherwise not connectable due to address conflicts between them. The ATG allows you to define multiple DECnet networks and to map between them. This can be done over all media types.

ATG Command Syntax

The ATG configuration commands are basically a modification to the standard DECnet global configuration commands.

The general syntax of the DECnet ATG command follows.

decnet *network-number keywords*

The argument *network-number* specifies the network number in the range 0 through 3, and the argument *keywords* is one of the configuration keywords (**area-max-cost**, for example). Commands without the *network-number* modifier apply to *network 0*.

You also can establish a translation entry to translate a virtual DECnet address to a real DECnet address by using this global configuration command:

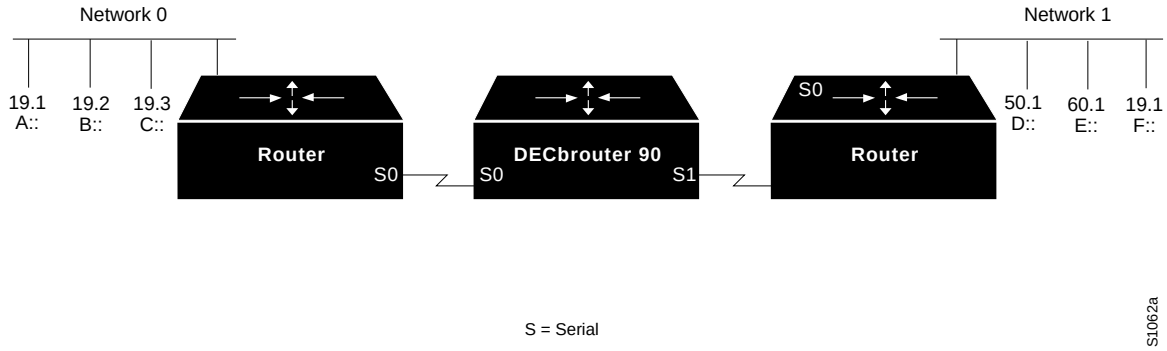
decnet *first-network* **map** *virtual-address second-network real-address*

The arguments *first-network* and *second-network* are DECnet network numbers in the range 0 through 3. The arguments *virtual-address* and *real-address* are specified as numeric DECnet addresses (10.5, for example).

ATG Configuration Examples

In Figure 4–1, the DECbrouter 90 is connected to two DECnet networks using Ethernet. The examples following Figure 4–1 refer to the configuration in the figure.

Figure 4–1 ATG Configuration Example



Example

In Network 0, the router is configured at address 19.4 and is a Level 1 router. In Network 1, the router is configured at address 50.5 and is an area router. At this point, no routing information is exchanged between the two networks. Each network in the router has a separate routing table.

```
!
decnet 0 routing 19.4
decnet 0 node routing-iv
interface serial 0
decnet 0 cost 1
!
decnet 1 routing 50.5
decnet 1 node area
interface serial 1
decnet 1 cost 1
!
```

To establish a translation map, enter these commands:

```
decnet 0 map 19.5 1 50.1
decnet 0 map 19.6 1 19.1
decnet 1 map 47.1 0 19.1
decnet 1 map 47.2 0 19.3
```

Packets in Network 0 sent to address 19.5 will be routed to Network 1, and the destination address will be translated to 50.1. Similarly, packets sent to address 19.6 in Network 0 will be routed to Network 1 as 19.1; packets sent to address 47.1 in Network 1 will be routed to Network 0 as 19.1; and packets sent to 47.2 in Network 1 will be sent to Network 0 as 19.3. Table 4–1 defines the parameters for the packet exchange translation map.

Table 4–1 A Packet Exchange Between Nodes A and D

Source			Destination	
A packet addressed as:	19.1	⟶	19.5	received on Ethernet0
Translates to:	47.1	⟶	50.1	and is transmitted out Ethernet1
A reply packet:	50.1	⟶	47.1	received on Ethernet1
Translates to:	19.5	⟶	19.1	and is transmitted on Ethernet0

Network 0 uses a block of addresses from its area to map the remote nodes. In Network 0, the router will advertise nodes 19.5 and 19.6. These nodes must not already exist in Network 0.

Network 1 uses another area for the address translation. Since the router will be advertising the availability of area 47, that area should not already exist in Network 1, because DECnet area fragmentation could occur.

Only nodes that exist in the maps on both networks will be able to communicate directly. Network 0 node 19.1 will be able to communicate with Network 1 node 50.1 (as 19.5), but will not be able to communicate directly with Network 1 node 60.1.

When naming nodes, use the appropriate address in each network. See the lists that follow for examples.

Network 0 VMS NCP Command File Sample

```
$ MCR NCP
define node 19.1 name A
define node 19.2 name B
define node 19.3 name C
define node 19.4 name GS
define node 19.5 name D
define node 19.6 name F
```

Network 1 VMS NCP Command File Sample

```
$ MCR NCP
define node 50.1 name D
define node 50.5 name GS
define node 60.1 name E
define node 19.1 name F
define node 47.1 name A
define node 47.2 name C
```

As an additional feature and security caution, DECnet "Poor Man's Routing" can be used between nodes outside of the translation map as long as those nodes have access to nodes that are in the map, so that a user on node B could issue the following VMS command:

```
$ dir A::D::E::
```

When a "Poor Man's Routing" connection is made between two networks, only the two adjacent nodes between the networks will have any direct knowledge about the other network. Application-level network access may then be specified to route through the connection.

Note

The DECbrouter 90 does not support "Poor Man's Routing" directly; the intermediate nodes must be VMS systems with "Poor Man's Routing" enabled in FAL.

Limitations of the ATG

Keep the following limitations in mind when configuring the address translation gateway:

- Both nodes that wish to communicate across the ATG must exist in the translation map. Other nodes outside of the map will see route advertisements for the mapped address but will be unable to communicate with them. An unmapped node trying to communicate with a mapped node will always get the message `"Node unreachable."` This can be confusing if another nearby node can communicate with mapped nodes because it is also a mapped node.
- Managing a large map can be tedious. Configuration errors will likely cause unpredictable network behavior.
- Third-party DECnet applications could fail if they pass node number information in a data stream (most likely a sign of a poorly designed application).
- Routing information for mapped addresses is static and does not reflect the reachability of the actual node in the destination network.

DECnet Monitoring Commands

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

Displaying DECnet Status

Use the SHOW DECNET INTERFACE command to display the DECnet status and configuration for all interfaces. Enter this command at the EXEC prompt:

show decnet interface [*interface unit*]

When the optional arguments *interface* and *unit* are specified, the relevant information for that particular interface is displayed.

In the following sample output, no specific interface was named, so you see information on all interfaces.

```
Global DECnet parameters for network 0:
  Local address is 19.15, node type is area
  Maximum node is 350, maximum area is 63, maximum visits is 63
  Maximum paths is 1, path split mode is normal
  Local maximum cost is 1022, maximum hops is 30
  Area maximum cost is 1022, maximum hops is 30
Ethernet 0 is up, line protocol is up
  Interface cost is 2, priority is 126, DECnet network: 0
  We are the designated router
  Sending HELLOs every 15 seconds, routing updates 40 seconds
  Smallest router blocksize seen is 576 bytes
  Routing input list is not set, output list is not set
  Access list is not set
  DECnet fast switching is enabled
Serial 0 is up, line protocol is up
  Interface cost is 5, priority is 126, DECnet network: 0
  Sending HELLOs every 15 seconds, routing updates 40 seconds
  Smallest router blocksize seen is 1498 bytes
  Routing input list is not set, output list is not set
  Access list is not set
  DECnet fast switching is enabled
Serial 1 is up, line protocol is up
  DECnet protocol processing disabled
```

Displaying the DECnet Address Mapping Information

Use the SHOW DECNET MAP command to display the address mapping information used by the DECnet address translation gateway. Enter this command at the EXEC prompt:

show decnet map

Displaying the DECnet Routing Table

Use the SHOW DECNET ROUTE command to display the DECnet routing table. Enter this command at the EXEC prompt:

show decnet route [*decnet-address*]

The optional argument *decnet-address* is a DECnet address and, when specified, the first hop route to that address is displayed. This command may show several routes for a destination when equal cost paths have been set with the DECNET MAX-PATHS command, and when there is more than one equal cost path to a destination. The currently selected route is indicated by an asterisk in the first column of the output. In interim mode, the selected route will never appear to change.

In the following sample output, a DECnet address name was not specified, so the entire routing table is displayed:

```
  Node       Cost  Hops   Next Hop to Node       Expires  Prio
*(Area)        0     0     (Local) ->19.15
*19.16         2     1   Ethernet0 ->19.16           44     64    V
*19.17         1     1   Serial0   ->19.17           31    125    VA
 19.17         2     1   Ethernet0 ->19.17           31    125    VA
*19.22         2     1   Ethernet0 ->19.22           41
```

In the displays:

- The Expires field displays how many seconds from now this entry expires.
- The Prio field is the router priority of this node.
- The V indicates that this is an adjacent Level 1 router; VA or A indicates that this is an adjacent Level 2 (area) router.
- An area node exists on the same local (0 hops) cable.

Displaying DECnet Traffic Statistics

The SHOW DECNET TRAFFIC command shows the DECnet traffic statistics, including datagrams sent, received, and forwarded. Enter this command at the EXEC prompt:

show decnet traffic

Following is sample output:

```
Total: 92275748 received, 758 format errors, 0 unimplemented
       0 not a gateway, 0 no memory, 689 no routing vector
HELLOs: 13113448 received, 26 bad, 15042 other area, 1842481 sent
Level 1 routing: 3919281 received, 0 bad, 580109 other area, 1485567 sent
Level 2 routing: 794130 received, 0 not primary router, 1140858 sent
Data: 73868022 received, 0 not long format, 68 too many visits
      73852256 forwarded, 0 mapped, 10880 returned, 0 converted
      0 access control failed, 10880 no route, 0 encapsulation failed
      0 inactive network, 0 incomplete map
```

In the displays:

- Total: displays the totals of packet types received.
  - The received field is the total of all types of DECnet packets received.
  - The format errors field lists the number of packets that appeared to be DECnet, but were formatted incorrectly. The number in the received field includes these packets.
  - The unimplemented field reports the number of incoming packets that are DECnet control packets, and how many specify a service that the router does not implement, including services implemented to forward Level 1 and Level 2 routing information, and router and end-system Hello packets.
  - The field labeled not a gateway reports the total number of packets received while not routing DECnet.
  - The field labeled no memory is a catch-all that records transaction attempts when the system has run out of memory.
  - The field labeled no routing vector indicates that either a routing update came in from another router when the router did not have an adjacency for it, or it had no routing vector for the type of routing update. Execute the DEBUG DECNET-ROUTING command (in the section Debugging DECnet to display additional information.
- HELLOs: displays the number of Hello messages received and sent.
  - The received field displays the total number of Hello messages received. All protocol types are included.

  - The bad field displays the total number of "bad" Hello messages received. Invoke the EXEC command DEBUG DECNET to display more information about why the Hello message was judged as bad.
  - The other area field displays the total number of Hello messages received from nodes on other areas when the router is a Level 1 router only.
  - The sent field displays the total number of Hello messages sent.
- Level 1 routing:displays the Level 1 routing updates received and sent.
  - The received field displays the total number of Level 1 routing updates received.
  - The bad field displays the total number of Level 1 updates received that were judged to be bad.
  - The other area field displays the total number of Level 1 updates from nodes in other areas.
  - The sent field displays the total number of Level 1 updates sent.
- Level 2 routing: displays the Level 2 routing updates received and sent.
  - The received field displays the total number of Level 2 updates received.
  - The field labeled not primary router should always be zero.
  - The sent field displays the total number of Level 2 updates sent.
- Data: displays the number of data packets received and sent.
  - The received field displays the total number of noncontrol (data) packets received.
  - The field labeled *not long format* displays the number of packets received that are not in the long DECnet format. This number should always be zero. If it is not, investigate the source of the improperly formatted packets.
  - The field labeled too many visits lists the number of packets received that have visited too many routers and have been flushed.
  - The forwarded field lists the total number of packets forwarded.
  - The mapped field displays the total number of ATG packets mapped.
  - The returned field lists the total number of packets returned to the sender at the sender's request.
  - The converted field displays the number of Phase IV packets converted to Phase V packets.
  - The field labeled access control failed lists the packets dropped because access control required it.
  - The no route field lists the total packets dropped because the router did not know where to forward them.
  - The field labeled encapsulation failed lists the number of packets that could not be encapsulated. This usually happens when there are entries missing in a map for a public data network, such as X.25 or Frame Relay. This can also occur if an interface is set for an encapsulation for which there is no defined DECnet encapsulation (such as PPP on serial interfaces).

— The field labeled `inactive network` displays the number of packets that appear to come from an unknown interface, or that ATG returned because they did not make sense.

— The field labeled `incomplete map` counts the number of packets that failed address translation. This usually means a node that is not in the ATG map is trying to access a node in another network advertised by the ATG.

Debugging DECnet

Use the EXEC commands described in this section to troubleshoot and monitor the DECnet network transactions. For each DEBUG command, there is a corresponding UNDEBUG command that turns the message logging off. Generally, you enter these commands with Digital customer engineers during troubleshooting sessions.

debug decnet-connects

The DEBUG DECNET-CONNECTS command enables logging of all connect packets that are filtered (permitted or denied) by DECnet access lists. When using connect packet filtering, it may be useful to start with the following basic access list:

```
access-list 300 permit 0.0 63.1023
access-list 300 permit 0.0 63.1023 eq any
```

This allows you to log all connect packets transmitted on interfaces to which you add this list with the ACCESS-GROUP configuration command. This will allow you to determine those elements on which your connect packets must be filtered.
Consider the following DEBUG DECNET-CONNECT display:

```
DNET: list 300 item #2 matched src=19.403 dst=19.309 on Ethernet0: permitted
      srcname="RICK" srcuic=[0,017]
      dstobj=42 ID="USER"
```

Here a packet matched the second item in access list 300. The source DECnet address was 19.403, the destination address was 19.309. The packet was permitted and transmitted on interface Ethernet0. The packet had a source object string RICK and UIC [0,017]. The destination was object 42. The user specified an ID of USER.

Note

Packet password and account information is not logged in the debug decnet-connects message, nor is it displayed by the SHOW ACCESS EXEC command. If you specify password or account information in your access list, these will be viewable by anyone with access to your router's configuration.

debug decnet-packets

The DEBUG DECNET-PACKETS command enables logging of all DECnet routing updates and Hello packets.

debug decnet-routing

The DEBUG DECNET-ROUTING command enables logging of all changes made to the DECnet routing table; that is, new routes, routes that change cost, and routes that expire.

DECnet Global Configuration Command Summary

This section provides an alphabetically arranged summary of all the DECnet global interface commands. These commands may appear anyplace in the configuration file.

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

Creates an access list. The argument *list* is an integer you choose between 300 and 399 that uniquely identifies the access *list*. The **permit** and **deny** keywords decide the access control action when a match happens with the address arguments. The argument pair *destination destination-mask* are the optional destination address and mask. The *no* form of the command deletes access *list*s.

[**no**] **access-list** *list* {**permit | deny**} *source source-mask destination destination-mask*

Creates an extended access list. The extended form of the DECnet access list has a source DECnet address and mask pair followed by a destination DECnet address and mask pair. The argument *list* is an integer you choose between 300 and 399 that uniquely identifies the access list. The **permit** and **deny** keywords decide the access control action when a match happens with the address arguments. The **no** form of the command deletes access lists.

[**no**] **access-list** *list* {**permit | deny**} *source source-mask* [*destination destination-mask*] [*connect-entries*]

DECnet access lists can be used to filter connect initiate packets. The argument *list* is the access list number in the range 300–399. The argument pair *source source-mask* is the source address and mask. The argument pair *destination destination-mask* are the optional destination address and mask. The *connect-entries* are the optional entries used to match connect packets. The **no** form of the command deletes access lists.

decnet area-max-cost *value*

Sets the maximum cost specification value for *interarea* routing. The argument *value* determines the maximum cost for a route to a distant area that the router may consider usable; the router treats as unreachable any route with a cost greater than the value you specify. A valid range for cost is from 1 to 1022; the default is 1022. This parameter is only valid for area routes.

decnet area-max-hops *value*

Sets the maximum hop count specification value for *interarea* routing. The argument *value* determines the maximum number of hops for a route to a distant area that the router may consider usable; the router treats as unreachable any route with a count greater than the value you specify. A valid range for the hop count is from 1 to 30; the default is 30. This parameter is only valid for area routes.

[**no**] **decnet conversion** *NSAP-prefix*

Enables DECnet conversion. The argument *NSAP-prefix* defines the value used for the IDP when constructing NSAPs from a Phase IV address. The command NO DECNET CONVERSION disables Phase IV/V conversion on the router.

decnet *network-number keywords*

Specifies ATG. The argument *network-number* specifies the network number in the range 0 through 3, and the argument *keywords* is one of the configuration keywords. Commands without the *network-number* modifier apply to "network 0."

decnet *first-network* **map** *virtual-address second-network real-address*

Establishes a translation entry to translate a virtual DECnet address to a real DECnet address. The arguments *first-network* and *second-network* are DECnet network numbers in the range zero through three. The arguments *virtual-address* and *real-address* are specified as numeric DECnet addresses.

decnet max-address *value*

Determines the largest node number specification allowed in the current area. The argument *value* is a node number from 1 to 1023; the default is 255. This parameter controls the sizes of internal routing tables and of messages sent to other nodes.

decnet max-area *value*

Sets the largest area number specification that the router can handle. The max-area keyword takes as its value an area number from 1 to 63; the default is 63.

decnet max-cost *value*

Sets the maximum cost specification for *intra-area* routing. The router ignores routes within the local area that have a cost greater than the corresponding value of this parameter. The argument *value* is a cost from 1 to 1022; the default is 1022.

decnet max-hops *value*

Sets the maximum hop count specification value for *intra-area* routing. The router ignores routes within the local area that have a hop count greater than the corresponding value of this parameter. The argument *value* is a hop count from 1 to 30; the default is 30.

decnet max-paths *value*

Defines the maximum number of equal cost paths to a destination that can be kept by the router. The argument *value* specifies the maximum number of equal cost paths, which is limited to 31. The default value is one, which specifies no multiple paths.

decnet max-visits *value*

Sets the limit on the number of times a packet can pass through a router. The argument *value* is a number from 1 to 63; the default value is 63.

decnet node-type {**area** | **routing-iv**}

Specifies the node type for the router. This command takes another keyword, **area** or **routing-iv**, as its value. If you specify **area**, the router exchanges traffic directly with routers in other areas, and participates in the interarea (Level 2) routing protocol, as well as acting as a intra-area (Level 1) router for its local area. If you specify **routing-iv** (the default), the router acts as an intra-area router, and routes packets out of the area by taking the least cost path to an interarea router.

decnet path-split-mode {**normal** | **interim**}

Sets the mode for splitting the routes between equal cost paths. The keyword **normal** selects the normal mode, where equal cost paths are selected on a round-robin basis. The normal mode is the default. The keyword **interim** selects an interim mode, where traffic for any particular higher-level session is always routed over the same path. This mode supports older implementations of DECnet (VMS Versions 4.5 and earlier) that do not support out-of-order packet caching.

[**no**] **decnet routing** *decnet-address*

Enables or disables DECnet routing. The argument *decnet-address* takes as its value an address in DECnet format X.Y, where X is the area number and Y is the node number. There is no default router address; you must specify this parameter for DECnet operation.

DECnet Interface Subcommand Summary

This section provides an alphabetically arranged summary of the DECnet interface subcommands. These commands follow an INTERFACE command.

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

Applies or removes an access list. The argument *list* can be either a standard or extended DECnet access list. A standard DECnet access list applies to destination addresses in this case.

[**no**] **decnet cost** *cost-value*

Sets or removes a cost value for an interface. The argument *cost-value* is an integer from 1 to 63. There is no default cost for an interface, although a suggested cost for Ethernet is 4, and all hosts on the same cable must share the same value. Use the NO DECNET COST subcommand to disable DECnet routing for an interface.

[**no**] *decnet hello-timer value*

Specifies how often the router sends Hello messages. This keyword takes as its value a time from 1 to 8.191 seconds; the default is 15 seconds. The **no** form of the command restores the default.

[**no**] **decnet in-routing filter** *list*

Provides access control to Hello messages or routing information received on this interface. Addresses that fail this test are treated as unreachable. The argument *list* is a standard DECnet access list. The **no** form of the command removes access control.

[**no**] **decnet out-routing-filter** *list*

Provides access control to routing information being sent out on this interface. Addresses that fail this test are shown in the update message as unreachable. The argument *list* is a standard DECnet access list. The **no** form of the command removes access control.

[**no**] **decnet route-cache**

Fast switching and the route cache are normally enabled. If you want to disable fast switching, use the **no** form of the command.

[**no**] **decnet router-priority** *value*

Sets a priority value for use in determining the default router. The argument *value* is a number from 0 to 127; the default is 64. The **no** form of the command restores the default.

[**no**] **decnet routing-timer** *value*

Specifies how often the router sends routing messages. The argument *value* is a time from 1 to 65535 seconds; the default is 40 seconds. The **no** form of the command restores the default.

5
Routing IP

This chapter begins with an introduction to the DECbrouter 90 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. It covers the following specific tasks and topics:

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

See Chapter 6 for information on the various routing protocols, how they have evolved, and how they are best used in complex internetworks.

The DECbrouter 90 Implementation of IP

The DECbrouter 90 implementation of TCP/IP provides all major services contained in the various protocol specifications. The DECbrouter 90 also provides the TCP and UDP *little services* called Echo and Discard. These services are described in RFC 862 and RFC 863.

The DECbrouter 90 supports both TCP and UDP at the transport layer, for maximum flexibility in services. Some DECbrouter 90 global and interface commands require UDP packets to be sent (see the section Configuring ICMP and Other IP Services.) The DECbrouter 90 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 do not have to initially enable IP routing. All 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 should disable it. You should perform the steps that follow to configure individual interfaces and other options.

1. Enter an address for the interface on which you will be routing IP using the IP ADDRESS interface subcommand.
2. Consider addressing options and broadcast packet handling, using commands described in the Setting IP Interface Addresses and Broadcasting in the Internet sections.
3. Optionally, configure packet sizes and other performance parameters as well as ICMP and other IP services. Information for these tasks is in the section Configuring ICMP and Other IP Services.

4. Configure access lists and other security options, if desired.
5. Configure routing. The IP routing protocols are discussed in Chapter 6.

Each task is described in the following sections; they are followed by descriptions of the EXEC commands needed to maintain, monitor, and debug an IP network. Summaries of the global configuration commands, interface subcommands, and line 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 *DECbrouter 90 Configuration and Reference, Volume 3*. The default setting is to perform IP routing.

Assigning IP Addresses

The official description of Internet addresses is found in RFC 1166, "Internet Numbers." The Defense Data Network (DDN) 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 it a network number or range of addresses appropriate to the number of hosts on its network.

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 type of 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.

Address Classes and Formats

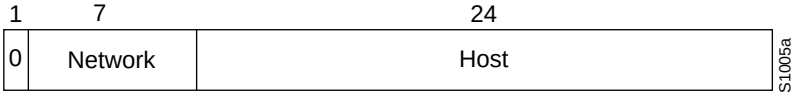
As described in RFC 1020, Internet addresses are 32-bit quantities and are divided into five classes. The classes differ in the number of bits allocated to the *network* and *host* portions of the address. For this discussion, consider a network to be a collection of devices (hosts) that have the same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

Note

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

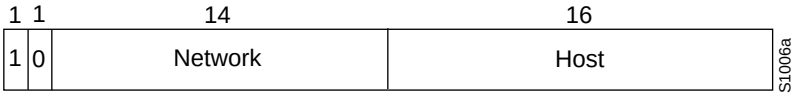
The Class A Internet address format allocates the highest 8 bits to the network field and sets the highest-order bit to 0 (zero). The remaining 24 bits form the host field. Only 126 Class A networks can exist, but each Class A network can have almost 17 million hosts (16,777,214). Figure 5–1 illustrates the Class A address format.

Figure 5–1 Class A Internet Address Format



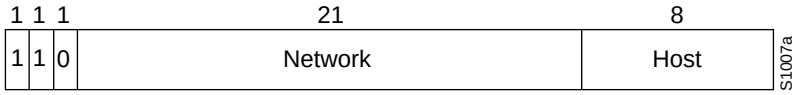
The Class B Internet address format allocates the highest 16 bits to the network field and sets the 2 highest-order bits to 1,0. The remaining 16 bits form the host field. Over 16,000 Class B networks can exist, and each Class B network can have over 65,000 hosts (65,534). Figure 5–2 illustrates the Class B address format.

Figure 5–2 Class B Internet Address Format



The Class C Internet address format allocates the highest 24 bits to the network field and sets the 3 highest-order bits to 1,1, and 0. The remaining 8 bits form the host field. Over two million Class C networks can exist, and each Class C network can have up to 254 hosts. Figure 5–3 illustrates the Class C address format.

Figure 5–3 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 4 highest-order bits are set to 1,1,1, and 0.

The Class E Internet address format is reserved for future use. In Class E addresses, the 4 highest-order bits are set to 1,1,1,and 1. The router currently ignores Class D and Class E Internet addresses, except the global broadcast address 255.255.255.255.

Allowable Internet Addresses

Some Internet addresses are reserved for special uses and cannot be used for host, subnet, or network addresses. Table 5–1 lists ranges of Internet addresses and shows which addresses are reserved and which are available for use.

Table 5–1 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 or host 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. Sending a packet to all specified hosts on a network is called a *broadcast*, which is described in the section Broadcasting in the Internet in this chapter. You also can find general information on broadcasts in Chapter 6 of this manual.

Note

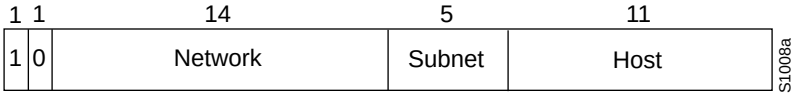
Because of these conventions, do not use an Internet address with all zeros or all ones in the host portion for your router address.

You can either configure the router's routing table manually or specify that a 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, described in the section Setting IP Interface Addresses in this chapter.)

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 contiguous leftmost bits in the host portion of the network addresses for a subnet field. For example, Figure 5–4 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 5–4 A Class B Address with a 5-Bit Subnet Field



Note

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 all subnets of a network be physically contiguous. In other words, the network must be set up so that traffic between any two subnets does not cross another network. This restriction applies to all IP routing protocols except OSPF. With OSPF you can route traffic between two subnets that are not physically contiguous.

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 a network address. 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 5–4, the subnet mask is 255.255.248.0. Table 5–2 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 5–2 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

Note

These masks are only relevant if you assume that the leftmost bits of the host portion are used contiguously. In order to function, the subnet bits must be contiguous, which is a convention employed by most IP networks.

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

The two required arguments are *address*, which is an IP address, and *mask*, 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 in this chapter 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 can use this command to specify additional secondary IP addresses by including the keyword **secondary** after the IP address and subnet mask.

Example

In the example that follows, 131.108.1.27 is the primary address and 192.31.7.17 and 192.31.8.17 are secondary addresses 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
ip address 192.31.8.17 255.255.255.0 secondary
```

Using Subnet Zero

Subnetting with a subnet address of zero generally is 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

The default is for this command to be disabled.

Example

In this example, subnet zero is enabled for the router:

```
ip subnet-zero
```

Local and Network Addresses: Address Resolution

A device in the Internet can 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 first must 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 Ethernet, 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.

Use the EXEC command SHOW ARP 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 also can 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 them, 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 IP 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. 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 nonstatic 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 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 in more detail later 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 ARP PROBE command, the router would send three (two for **probe**) 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 4 hours (14,400 seconds). A value of 0 seconds sets no timeout; then the cache entries are never cleared.

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, PROBE, Entry Timeout: 14400 sec
```

The following example illustrates how to set the ARP timeout to 12,000 seconds 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 IP PROXY-ARP interface subcommand enables proxy ARP on the interface. The full command syntax for this command follows.

ip proxy-arp
no ip proxy-arp

Proxy ARP is enabled by default.

Address Resolution Using Probe

The router can be made to use the Probe protocol (in addition to ARP) whenever it attempts to resolve an IEEE-802.3 or Ethernet local data link address. Use the ARP PROBE interface subcommand to enable use of the Probe protocol. 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 syntax for this command, which enables or disables Probe for IEEE-802.3 and Ethernet networks, is as follows:

arp probe
no arp probe

The other options of the ARP command are discussed in the section Address Resolution Using ARP in this chapter. This command is disabled by default.

Reverse Address Resolution Using RARP and BootP

Reverse ARP (RARP) is defined in RFC 903. If a router does not know the IP address of one of its Ethernet interfaces, it will try RARP during startup 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 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 nonvolatile memory uses both 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 RARP, 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.

The DECbrouter 90 supports 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.

For example, if the network address is 128.1.0.0, 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), the address 128.1.5.255 specifies all hosts on subnet 5 of network 128.1.0.0, another directed broadcast.

The IP DIRECTED-BROADCAST interface subcommand is used to enable forwarding of directed broadcasts on an interface. The full syntax of this command follows.

ip directed-broadcast
no ip directed-broadcast

The default is to forward directed broadcasts. Disable forwarding of directed broadcasts with the NO IP DIRECTED-BROADCAST subcommand.

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 also can 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 or addresses.

If the router does not have nonvolatile memory, and you want to specify the broadcast address to use before the router has been configured, you can change the Internet broadcast address by setting bits 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 5–3 shows the combined effect of setting bits 10 and 14.

Table 5–3 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 Getting Started Guide.

UDP Broadcasts

Network hosts occasionally employ user datagram protocol (UDP) broadcasts to determine address, configuration, and name information. If such a host is on a network segment that does not include a server host, UDP broadcasts are not forwarded; therefore, no answer or reply is received.

Note

UDP is an alternative transport for TCP connectionless networks. UDP is defined in RFC 768.

To correct this situation, configure the interface of your router to forward certain classes of UDP broadcasts to a helper address. See the description of the IP HELPER-ADDRESS interface subcommand and the IP FORWARD-PROTOCOL global configuration commands in this chapter for more information.

Forwarding 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 the interface. 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. The **no** version disables the forwarding of broadcast packets to specific addresses.

Example

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

```
interface ethernet 0
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 as follows:

ip forward-protocol {**udp** | **nd**} [*port*]
no ip forward-protocol {**udp** | **nd**} [*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, the following 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 following example uses the IP FORWARD-PROTOCOL command to specify forwarding of UDP only, then defines a helper address.

```
ip forward-protocol udp
!
interface ethernet 0
ip helper-address 131.120.1.0
```

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. The 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 can now be flooded. Only one copy of the packet will be put on each network segment.

The IP FORWARD-PROTOCOL SPANNING-TREE command uses the database created by the bridging spanning-tree protocol.

Note

Bridging must be configured on each interface that is to participate in the flooding in order to support this capability.

If an interface does not have bridging configured, it still will 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 actual bridging is desired, you can configure a type-code bridging filter that will deny all packet types from being bridged. Refer to the *DECbrouter 90 Configuration and Reference, Volume 3* for more information about using access lists to filter bridged traffic. 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):

- The packet must be a MAC-level broadcast.
- The packet must be an IP-level broadcast.
- The packet 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 packet's 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. The destination address 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 (including intelligent bridges), because they are Layer 2 devices, forward broadcasts to all network segments, thus propagating all broadcast storms.

The best solution to the broadcast storm 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 DECbrouter 90, can accept and interpret all possible forms of broadcast addresses.

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 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*. 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 it does not recognize, it sends an ICMP *Protocol Unreachable* message to the source.

If the router receives a datagram that it cannot 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 sending these messages. The full syntax for this command follows.

ip unreachables
no ip unreachables

The default is to send unreachable messages. The NO IP UNREACHABLES subcommand disables sending ICMP unreachable messages on an interface.

Generating Redirect Messages

The 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. It does so because the originating host presumably could 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 sending these messages, as follows:

ip redirects
no ip redirects

The default is to send redirects. The **no** version disables sending redirect messages.

Setting and Adjusting Packet Sizes

All interfaces have a default maximum packet size or MTU. You can set the IP maximum transmission unit (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*
ip mtu *bytes*

The default 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.

Note

Changing the **mtu** value with the MTU interface subcommand can affect the value for the IP MTU interface subcommand. If the current value specified with the IP MTU interface subcommands is the same as the value specified with the MTU interface subcommand, when you change the value for the MTU interface subcommand, the value for **ip mtu** is automatically modified to match the new MTU interface subcommand value. However, the reverse is not true. In other words, changing the value for the IP MTU subcommand has no effect on the value for the MTU interface subcommand.

Example

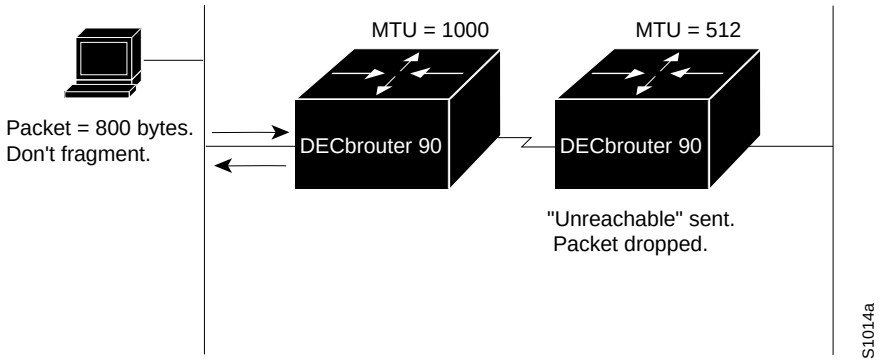
In the following example, the maximum IP packet size for the first serial interface is set to 300 bytes.

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

MTU Path Discovery

The IP MTU Path Discovery mechanism is enabled 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 it requires fragmentation (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 5–5.

Figure 5–5 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-byte 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 also would include the MTU of the next-hop network link in the low-order bits of an unused header field.

MTU Path Discovery also is 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.

Using 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* message to the source of the ICMP *Echo* message. See the section IP Ping Command 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 follows:

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

The argument *name* is the host name, and the argument *address* is the associated IP address. Up to eight addresses can be bound to a host name. The **no** version removes names-to-address mapping.

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

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
```

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* 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

This command defines dec.com as the default name.

```
ip domain-name dec.com
```

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

By default, the IP Domain Name System (DNS)-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 default is for DNS lookup to be enabled. The **no** version disables DNS host-name lookup.

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

The default is for IEN-116 lookup to be disabled. Name service is disabled by default; use the IP IPNAME-LOOKUP command to enable name service.

HP Probe Proxy Support

HP Probe Proxy support allows a router to respond to HP Probe Proxy Name requests. These are typically used at sites that have HP equipment and are already using HP Probe. Use the interface subcommand IP PROBE PROXY to enable or disable HP Probe Proxy, as follows:

ip probe proxy
no ip probe proxy

This command is disabled by default. To use the proxy service, you must first enter the host name of the HP host into the host table through the global configuration command IP HP-HOST. The full syntax is as 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 its 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 in 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 this example, several domain names are added to a list:

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

Example 2

This example 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:

1. Create a list.
2. Apply the list to interfaces to implement a policy.

You can use access lists in several ways:

- To control the transmission of packets on an interface
- To control virtual terminal line access
- To restrict contents of routing updates

The software supports two styles of access lists for IP:

- The standard IP access lists use source addresses for matching operations.
- Extended IP access lists use source and destination addresses for matching operations, as well as optional protocol type information.

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. The full command syntax follows.

access-list *list* {**permit** | **deny**} *source source-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 source address being tested to *source*, ignoring any bits specified in *source-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 *source* and *source-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 router examines first the actual address, then the wildcard (*source-mask*) 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.

Implicit Masks

There are *implicit* masks in IP access lists. For instance, if you omit the mask from an associated IP host address access list specification, 0.0.0.0 is assumed to be the mask. Consider the following example configuration:

```
access-list 1 permit 0.0.0.0
access-list 1 permit 131.108.0.0
access-list 1 deny 0.0.0.0 255.255.255.255
```

For this example, the following masks are implied in the first two lines:

```
access-list 1 permit 0.0.0.0 0.0.0.0
access-list 1 permit 131.108.0.0 0.0.0.0
```

The last line in the configuration (using the deny keyword) can be left off, because IP access lists implicitly *deny* all other access. This is equivalent to finishing the access list with the following command statement:

```
access-list 1 deny 0.0.0.0 255.255.255.255
```

Example

The following access list only allows access for 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. Any hosts with a source address that does not match the access list statements will be rejected.

```
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
! (Note: all other access implicitly denied)
```

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 through 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 of source address bits to be ignored and is also in dotted-decimal format. 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 used 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 created initially, 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 a specific access list.

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 always will 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 ordinarily would 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
established
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.1.2 eq 25
interface serial 0
ip access-group 101
interface ethernet 0
ip access-group 102
```

This is a complex example, designed to show the power of all the options just discussed. The IP ACCESS-GROUP interface subcommand is described later in this chapter.

Controlling Line Access

To restrict incoming and outgoing connections between a particular virtual terminal line (into a DECbrouter 90) and the addresses in an access list, use the ACCESS-CLASS line configuration subcommand. Full command syntax for this command is as 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

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

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 IP ACCESS-GROUP interface subcommand, as follows:

ip access-group *list*
no 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 source 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*. The **no** version removes the access group specified.

Example

The following example applies list 101:

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

Configuring the IP Security Option (IPSO)

All aspects of the IP security option (IPSO) are set up using configuration commands. The DECbrouter 90 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 is an early version of RFC 1108. The following list summarizes the differences between the DECbrouter 90 implementation and RFC 1108:

- DIA Authority is equivalent to SCI Authority.
- The DECbrouter 90 supports SCI.
- The DECbrouter 90 does not support DOE Authority keyword.
- The DECbrouter 90 only accepts a four-byte IPSO.

The following list describes some of the abilities of the 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.
- Reorders 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 5–4 lists the level keywords used by the DECbrouter 90 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 5–5 lists the authority keywords used by the DECbrouter 90 software and their corresponding bit patterns.
- **Label**—a combination of a security level and an authority or authorities.

Table 5–4 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 5–5 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.

ip security
no ip security

Use one of the IP SECURITY commands, described in the following sections, 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 are listed in Table 5–4 and Table 5–5.

Example

The following example sets 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* [*authority1...*] **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 proper subset of *authority2*. That is, *authority1* specifies those authority bits that are required on a datagram, while *authority2* specifies the required bits plus any optional authorities that also can be included. If the *authority1* more of the authority bits in *authority2*.

Example

The following example specifies 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 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 this example, 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 as follows:

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 a 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 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 procedure is performed after all security tests in the router have been passed. This command 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 follows:

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. Default value usages include the following:

- 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; otherwise it is off.
- The default value for **add** is *on* if the interface is not unclassified Genser; otherwise it is *off*.

Table 5–6 provides a list of all default values.

Table 5–6 Default Security Keyword Values

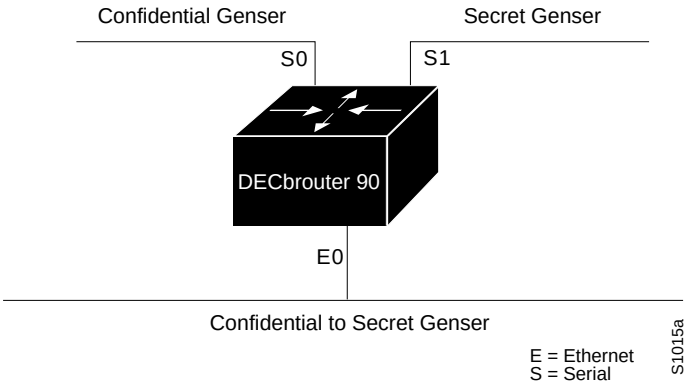
Type	Level	Authority	Implicit	Add
None	None	None	On	Off
Dedicated	Unclassified	Genser	On	Off
Dedicated	Any	Any	Off	On
Multilevel	Any	Any	Off	On

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 interfaces are presented. These interfaces are running at security levels of Confidential Genser, Secret Genser, and Confidential to Secret Genser, as shown in Figure 5–6.

Figure 5–6 IPSO Security Levels



Example 1

The following commands set up interfaces for the configuration in Figure 5–6.

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

It is possible for the setup to be much more complex.

Example 2

In this example, there are devices on serial 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 also is a host on Ethernet 0 that requires the security option to be the first option present, and this condition also must be specified. The new configuration follows.

```
interface serial 0
ip security dedicated confidential genser
ip security implicit-labelling
ip security strip
interface serial 1
ip security dedicated secret genser
ip security extended-allowed
!
interface ethernet 0
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 also is reported to the sending host when allowed by the configuration. 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. See Table 5–7.

Table 5–7 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 and their meanings are:

- "ICMP Parameter problem, code 0" — Error at pointer.
- "ICMP Parameter problem, code 1"— Missing option.
- "ICMP Parameter problem, code 2" — See Note that follows.
- "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 the router received a datagram containing a maximum length IP header, but no security option. 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 Account 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 the *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, 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. The default is zero (0), which is equivalent to the **no** version of the command.

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 in this chapter for more options on monitoring your network's accounting.

Special IP Configurations

This section discusses how to configure static routes and 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 is to perform 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 as follows:

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.

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 serial interfaces using HDLC encapsulation can be unnumbered. It is not possible to use this subcommand with X.25 interfaces.
- You cannot use the PING command to determine whether the interface is up, because the interface has no address. Simple network management protocol (SNMP) can 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 that has an IP address, not another unnumbered interface.
- You must include a NO IP ADDRESS command for the serial interface.

Note

Using an unnumbered serial line between different major networks requires special care. Any routing protocol running across the serial line must not advertise subnet information.

Example

In this example, 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
no ip address
ip unnumbered ethernet 0
```

Enabling Fast Switching

The IP ROUTE-CACHE interface subcommand controls the use of a high-speed switching cache for IP routing. The route cache is enabled by default and allows outgoing packets to be load balanced 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.

The DECbrouter 90 generally offers 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 best choice.

Compressing TCP Headers

You can compress the headers of your TCP/IP packets in order to reduce the size of your packets. TCP header compression is only supported on serial lines using HDLC encapsulation. RFC 1144 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 (usually using terminals) tends to use small packets, while file transfers use large packets. This feature only compresses the TCP header, 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**]
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. This means that fast interfaces like T1 can overload the router. Think about your network's traffic characteristics before using this command. See the section Monitoring the IP Network in this chapter for more information on commands for monitoring your compressed traffic.

The IP TCP COMPRESSION-CONNECTIONS 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. The command syntax is as follows:

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 of ten cache entries.

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

Configuration Examples

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

Configuring Serial Interfaces

In the following example, the second serial interface is given interface 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 (one 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 interface subcommand, so the default protocols (TFTP, DNS, IEN-116, Time, NetBIOS, and BootP) will be flooded.

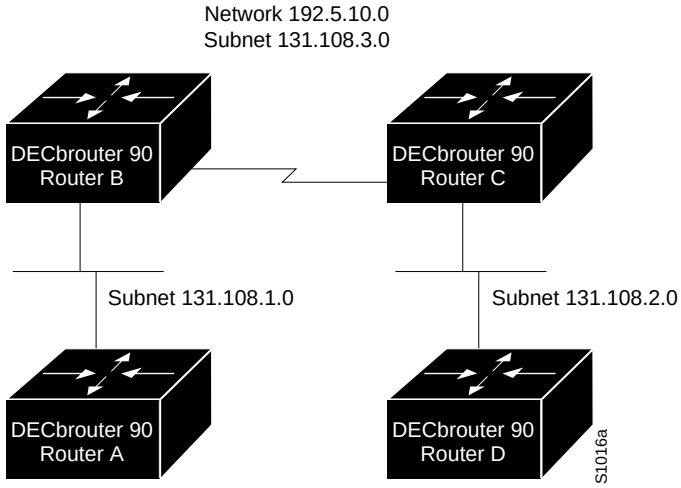
Example

```
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 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 following example, networks 131 and 192 are separated by a backbone, as shown in Figure 5–7. The two networks are brought into the same logical network through the use of secondary addresses.

Figure 5–7 Creating a Network from Separated Subnets



Example—Router B

```
interface serial 0
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 serial 0
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 that follows changes some of the ICMP defaults for the first Ethernet interface. Disabling the sending of redirects could mean that you do not think your routers on this segment will ever have to send a redirect. Lowering the error-processing load on your router would increase efficiency. Disabling the unreachables messages will have a secondary effect—it also will 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—you would be disabling options that your router would be unlikely to use anyway.

Example

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

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 that follows establishes a domain list with several alternate domain names.

Example

```
ip domain-list dec.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 last line of the list shows that the router would accept addresses on all other network 36.0.0.0 subnets.

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
ip access-group 2
```

Configuring Extended Access Lists

In this example, 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
ip 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. This command also clears the IP route 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 remove 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. The host name entries will not be removed from NVRAM, but will be cleared in running memory.

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 also can 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
```

A listing is available at both the user and privileged levels. The display will show relevant commands for each level.

Displaying the 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 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 5–8 describes the fields seen.

```
Protocol  Address          Age (min)     Hardware Addr   Type      Interface
Internet  131.108.62.192         187     0800.2010.a3b6  ARPA      Ethernet0
Internet  131.108.62.245          68     0800.200e.28f8  ARPA      Ethernet0
Internet  131.108.1.140          139     0000.0c01.2812  ARPA      Ethernet0
Internet  131.108.62.160         187     0800.200e.4dab  ARPA      Ethernet0
Internet  131.108.1.111           27     0800.2007.8866  ARPA      Ethernet0
Internet  131.108.1.117          119     0000.0c00.f346  ARPA      Ethernet0
Internet  131.108.1.115           28     0000.0c01.0509  ARPA      Ethernet0
Internet  131.108.1.77             1     0800.200e.57ce  ARPA      Ethernet0
Internet  192.31.7.29            225     aa00.0400.0234  ARPA      Ethernet0
Internet  192.31.7.17            118     2424.c01f.0711  ARPA      Ethernet0
Internet  192.31.7.18            135     0000.0c01.2817  ARPA      Ethernet0
Internet  192.31.7.21            119     2424.c01f.0715  ARPA      Ethernet0
Internet  131.108.1.33             1     0800.2008.c52e  ARPA      Ethernet0
Internet  131.108.62.1              -     0000.0c00.750f  ARPA      Ethernet0
Internet  131.108.31.35          119     0800.2010.8c5b  ARPA      Ethernet0
Internet  131.108.62.7            14     0000.0c00.33ce  ARPA      Ethernet0
Internet  131.108.1.55           155     0800.200e.e443  ARPA      Ethernet0
```

Table 5–8 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 a MAC address that corresponds to network address
Type	Type of encapsulation: ARPA = Ethernet SNAP = RFC 1042 ISO1 = IEEE 802.3

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 the 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 HOSTS at the user-level (or privileged-level) prompt.

Following is sample output:

```
show hosts
Default domain is DEC.COM
Name/address lookup uses domain service
Name servers are 255.255.255.255
Host                    Flags      Age Type   Address(es)
SLAG.DEC.COM            (temp, OK)  1   IP    131.108.4.10
CHAR.DEC.COM            (temp, OK)  8   IP    192.31.7.50
CHAOS.DEC.COM           (temp, OK)  8   IP    131.108.1.115
DIRT.DEC.COM            (temp, EX)  8   IP    131.108.1.111
DUSTBIN.DEC.COM         (temp, EX)  0   IP    131.108.1.27
DREGS.DEC.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. 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.
- The Address(es) field shows the address of the host. One host may have up to eight addresses.

Displaying the Route Cache

The SHOW IP CACHE command displays the routing table cache that is used to fast-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
```

Example

The SHOW IP CACHE display shows MAC headers up to 46 bytes, or 92 characters, long to accommodate the SMDS header.

```
router> show ip cache
Hash Destination Interface MACHeader
62/0 131.108.173.32 Serial0
      000000990604C12001730032FFFFC12001730020FFFF0
 40300000300010000000000000000AAAA030000000800
```

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 0, and so on).
- 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 sample output that follows was obtained by specifying the serial 0 interface. Table 5–9 describes the fields seen.

```
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
```

Table 5–9 Show IP Interface Field Descriptions

Field	Description
Broadcast Address	Shows the broadcast address.
Helper Address	Specifies a helper address, if one has been set.
Outgoing access list	Indicates whether or not the interface has an outgoing access list set.
Proxy ARP	Indicates whether Proxy ARP is enabled for the interface.
Security Level	Specifies the IPSO security level set for this interface.
ICMP redirects	Specifies whether redirects will be sent on this interface.
ICMP unreachables	Specifies whether unreachable messages will be sent on this interface.

(continued on next page)

Table 5–9 (Cont.) Show IP Interface Field Descriptions

Field	Description
ICMP mask replies	Specifies whether mask replies will be sent on this interface.
IP fast switching	Specifies whether fast switching has been enabled for this interface. It is generally enabled on serial interfaces, such as this one.
Gateway Discovery	Specifies whether the discovery process has been enabled for this interface. It is generally disabled on serial interfaces, such as this one.
IP accounting	Specifies whether IP accounting is enabled for this interface and what the threshold (maximum number of entries) is.
TCP/IP header compression	Indicates whether compression is enabled or disabled.
Probe proxy name	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, Serial1
```

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 had no ARP request entry 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. Table 5–10 describes the fields seen.

```
TCP/IP header compression statistics:
  Interface Serial1: (passive, compressing)
    Rcvd:    4060 total, 2891 compressed, 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
```

Table 5–10 Show IP TCP Header Compression

Field	Description
Rcvd:	
total	Total number of TCP packets received.
compressed	Total number of TCP packets compressed.
errors	Unknown packets.
dropped	Number of packets dropped due to invalid compression.
buffer copies	Number of packets that had to be copied into bigger buffers for decompression.
buffer failures	Number of packets dropped due to a lack of buffers.
Sent:	
total	Total number of TCP packets sent.
compressed	Total number of TCP packets compressed.
bytes saved	Number of bytes reduced.

(continued on next page)

Table 5–10 (Cont.) Show IP TCP Header Compression

Field	Description
bytes sent	Number of bytes sent.
efficiency improvement factor	Improvement in line efficiency because of TCP header compression.
Connect:	
number of slots	Size of the cache.
long searches	Indicates the number of times the software had to look to find a match.
misses	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.
hit ratio	Percentage of times the software found a match and was able to compress the header.
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.

IP Ping Command

The privileged-mode 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 two PING command outputs for IP. The first PING command is the simplest form of the command (specified with the destination address in line with the PING command). This version sends five 100-byte ICMP echoes. The second version provides a complete prompt sequence in verbose mode.

Sample Session 1

```
gw# ping 131.108.62.102
Type escape sequence to abort.
Sending 5 100-byte ICMP Echos to 131.108.62.102, timeout is 2 seconds:
.....
Success rate is 0 percent

gw# ping
Protocol [ip]:
Target IP address: 131.108.1.27
Repeat count <LEVEL>(text}):
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:
..!!!
Success rate is 60 percent, round-trip min/avg/max = 4/6/12 ms
```

The PING command uses the notation shown in Table 5–11 to indicate the responses it sees.

Table 5–11 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.
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, type Ctrl/^, X, which is done by simultaneously pressing the Ctrl, Shift, and 6 keys, letting go, then pressing the x key).

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 traveled 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 also can specify the router address to use as the source address for ping packets. Here, it 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
```

IP Trace Command

The EXEC command TRACE allows you to discover the routing path your router's packets are taking through your network.

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 is exceeded, or when the user interrupts the trace with the escape sequence. By default, to invoke the escape sequence, type Ctrl/^, X—done by simultaneously pressing the Ctrl, Shift, and 6 keys, letting go, then pressing the x key.

Common Trace Problems

Due to bugs in the IP implementation of various hosts and routers, the TRACE command may behave in odd ways. Not all destinations will respond correctly 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 reuse 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 time out on responses 6 through 11.

Tracing IP Routes

When tracing IP routes, you can set the following TRACE command parameters:

- 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 normally will 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 can 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—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 it can be set to a higher value to suppress the display of known hops.
- Maximum Time to Live [30]—The largest TTL value that can be used. The default is 30. The trace command terminates when the destination is reached or when this value is reached.
- Port Number—The destination port used by the UDP probe messages. The default is 33,434.
- Loose, Strict, Record, Timestamp, Verbose—IP header options. You can 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—Allows you to specify a list of nodes that must be traversed when going to the destination.
- Strict Source Routing—Allows you to specify a list of nodes that must be the only nodes traversed when going to the destination.
- Record—Allows you to specify the number of hops to leave room for.
- Timestamp—Allows you to 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.

Table 5–12 describes the output from the trace test.

Table 5–12 Trace Test Characters

Char	Meaning
nn msec	For each node, the round-trip time (in nn milliseconds) for the specified number of probes
*	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 a corresponding UNDEBUG command that turns the display off. In general, you need use these commands only during troubleshooting sessions with Digital personnel, because display of debugging messages can impact the operation of the router.

debug arp

The DEBUG ARP command enables logging of ARP 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 also can 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 ip-tcp-header-compression

The DEBUG IP-HEADER-COMPRESSION command enables logging of TCP header compression.

debug probe

Debugging information, including information about HP Probe Proxy Requests, is available through DEBUG PROBE.

IP Global Configuration Command Summary

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

[**no**] **access-list** *list* {**permit | deny**} *source source-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 *source* is a 32-bit, dotted-decimal notation IP address to which the router compares the source address being tested. The argument *source-mask* is 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* is mask bits for the source address in 32-bit, dotted-decimal notation. The arguments *destination* and *destination-mask* are 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 Ethernet. The optional keyword **alias** indicates that the router should respond to ARP requests as if it were the owner of the specified IP address. The **no** version removes the specified entry from the ARP cache.

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

Specifies a set of filters 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. The default is zero (0), which is equivalent to the **no** version of the command.

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

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

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

Defines a list of default domain names to complete unqualified host names. The argument *name* is 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 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 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. Use the **no** version of the command to prevent flooding of IP addresses.

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

Allows you to specify which protocols and ports the router will forward. 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.

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

Defines a static host-name-to-address mapping in the host cache. The argument *name* is the host name; the argument *TCP-port-number* is a TCP port number—Telnet by default (port 23); and the argument *address1* [*address2...address8*] represents associated IP addresses (up to eight can be specified). The **no** version removes the name-to-address mapping.

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

Enables or disables 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 enabled by default; the **no** variation of the command restores the default.

[no] ip name-server *server-address1* [*server-address2. . . server-address6*]

Specifies the address of the name server to use for name and address resolution. The arguments *server-address* are the Internet addresses of up to six name servers. 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 hardware addresses. The keyword **arpa**, which is the default, specifies standard Ethernet style ARP (RFC 826), **probe** specifies the HP 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 to 14,400 seconds. A value of zero seconds sets no timeout.

[**no**] **ip 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** *mask* [*secondary*]

Sets an IP address for an interface. The two required arguments are an IP address (*address*) and the subnet mask (*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 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. This is disabled by default.

[no] ip proxy-arp

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

[no] ip redirects

Enables or disables sending ICMP redirects on the interface. ICMP redirects are normally sent.

[no] ip route-cache [cbus]

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 add

Adds a basic security option to all datagrams leaving the router on the specified interface.

[no] ip security *arguments*

Controls the use of the Internet IP security option.

[no] ip security dedicated *level authority* [*authority...*]

Sets or unsets the requested level of classification and authority on the interface. See Table 13-4 and Table 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-first

Prioritizes the presence of security options on a datagram.

[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 Table 5–4 and Table 5–5 for the *level* and *authority* arguments.

ip security multilevel *level1* [*authority1...*] **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 Table 5–4 and Table 5–5 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 cannot be another unnumbered interface or the interface itself.

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

Enables or disables 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*.

6
The IP Routing Protocols

This chapter describes routing protocol options for the Internet protocol (IP) suite. Chapter 5 contains all the information you need for configuring IP. This chapter focuses on IP routing protocols. Other protocol stacks—DECnet, Novell, and Apollo–are described in their own chapters. Topics in this chapter include:

- Introduction to IP routing protocols
- Starting the routing process for a particular protocol
- Configuring static and dynamic routing
- Configuring the supported IP protocols—IGRP, OSPF, RIP, Hello, EGP, and BGP
- Configuring multiprotocol operations, including redistribution of information from one protocol to another
- Filtering incoming and outgoing updates on the interface
- Enabling IS-IS, specifying preferred routes, and filtering information
- Importing IS-IS routes and routes learned by other IP routing protocols
- Generating a default route and summarizing address ranges
- Configuring Network Entity Titles
- Specifying router level support, designated routers, and interface circuit types
- Configuring IS-IS link state metrics
- Setting advertising and retransmission intervals
- Configuring integrated IS-IS authentication passwords
- Monitoring and debugging IS-IS

DECbrouter 90 Supported Routing Protocols

Routing is the process of determining where to send data packets destined for addresses outside the local network. Routers gather and maintain routing information to enable the transmission and receipt of such data packets. Conceptually, routing information takes the form of entries in a routing table, with one entry for each identified route. The router can create and maintain the routing table dynamically to accommodate network changes whenever they occur.

Note

It is traditional when discussing IP routing protocols to refer to routers as *gateways*. For this reason, many IP routing protocols contain the word *gateway* as part of their name. Keep in mind that a gateway is a generic layer 3 and above device that connects one software stack to another. So an X.25 gateway, an electronic mail (layer 7) gateway, and a router are all—in a computer science sense—gateways.

Interior and Exterior Protocols ICMP

The routing protocols are broadly divided into two classes: interior gateway protocols (IGPs) and exterior gateway protocols (EGPs). Supported interior routing protocols include the routing information protocol (RIP), Hello, the interior gateway routing protocol (IGRP), and the open shortest path first (OSPF) protocol. Interior protocols are used for routing networks that are under a common network administration. The exterior routing protocols include the exterior gateway protocol (EGP) and the border gateway protocol (BGP). Exterior protocols are used to exchange routing information between networks that do not share a common administration.

- RIP is the routing protocol used by the routed process on Berkeley-derived UNIX systems. Many networks use RIP; it works well for small, isolated, and topologically simple networks.
- Hello is an older interior routing protocol used in the early National Science Foundation (NSF) backbone network.
- IGRP, developed by Cisco Systems, focuses on large networks with complex topology and segments having different bandwidth and delay characteristics.
- OSPF, developed by the OSPF working group of the Internet Engineering Task Force (IETF), is an IGP based on *link state* technology.
- EGP is the original exterior protocol and is still used primarily in the DDN (Defense Data Network) and NSFnet (National Science Foundation Network).
- BGP is a more recent exterior routing protocol that solves some of EGP's failings.

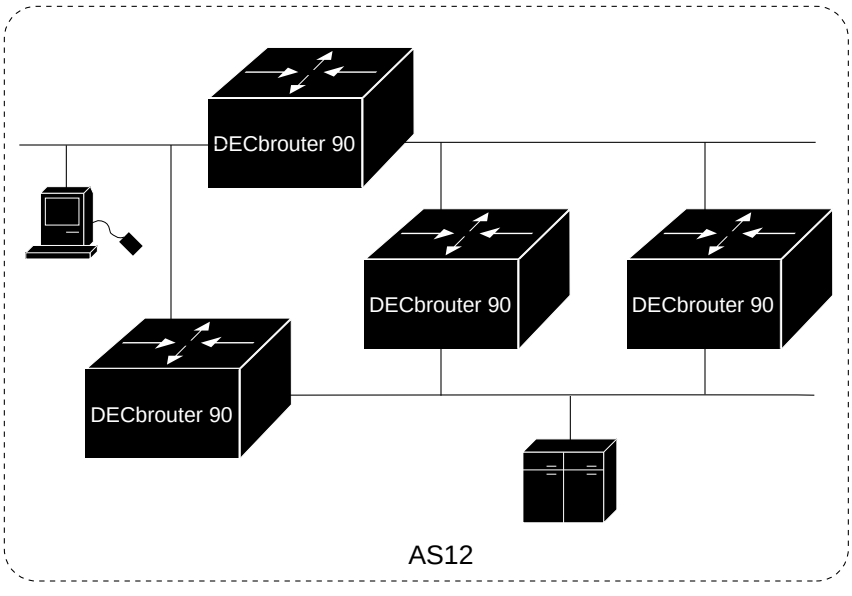
The DECbrouter 90 offers many protocol-independent routing features. For example, subnetting lets you divide a network into logical subparts. Load balancing lets you split network traffic over parallel paths, which provides greater overall throughput and reliability. Because the router can avoid routing loops, you can implement general network topologies. Notification of disabled interfaces eliminates network *black holes*. Static routing table entries can provide routing information when dynamically obtained entries are not available.

Autonomous Systems

An autonomous system (AS) is a collection of networks under a common administration that share a common routing strategy (see Figure 6–1). An autonomous system can comprise one or many networks, and each network may or may not have an internal structure (subnetting). The autonomous system number, which is assigned by the Network Information Center (NIC), is a 16-bit decimal number that uniquely identifies the autonomous system. An assigned AS number is required when running EGP or BGP. All routers that belong to

an autonomous system must be configured with the same autonomous system number.

Figure 6–1 Autonomous System 12 Contains Four Routers



Multiple Routing Protocols

The multiple routing protocol support in the DECbrouter 90 was designed for connecting networks that might use different routing protocols. It is possible, for example, to run RIP on one subnetted network, IGRP on another subnetted network, and to exchange the routing information in a controlled fashion. The routing protocols available today were not designed to interoperate with one another, so each protocol collects different types of information and reacts to topology changes in its own way. For example, RIP uses a hop count metric and IGRP uses a five-valued vector of metric information. In the case where routing information is being exchanged between different networks that use different routing protocols, there are many configuration options to enable and to filter the exchange of routing information. See the section Redistributing Routing Information in this chapter.

Multiple IP Routing Processes

The DECbrouter 90 can handle simultaneous operation of up to 30 dynamic IP routing processes.

The combination of routing processes on a router can consist of the following protocols (with the limits noted):

- Any number of IGRP routing processes
- Any number of OSPF routing processes
- Any number of EGP routing processes
- One BGP routing process
- One RIP routing process

- One CHAOS routing process
- One Hello routing process

Configuration Overview

Each routing protocol must be configured separately. Therefore, most of the configuration information discussed in this chapter appears in protocol-specific subsections. The interior routing protocols are listed first, followed by the exterior protocols. With any routing protocol, follow these basic steps:

1. Create the routing process with one of the ROUTER commands.
2. Configure one or more NETWORK router subcommands.
3. Configure the protocol specifics.

The next sections provide a review of the two protocol classes and how they are configured, followed by sections that explain how to configure each of the routing protocols. EXEC-level commands for monitoring the IP routing operations also are provided; these are explained at the end of this chapter, along with alphabetical summaries of the configuration commands.

Configuring the Interior Routing Protocols

All IP routing protocols must have a list of networks specified by the NETWORK router subcommand before routing activities can begin. The routing process will listen to updates from other routers on these networks and will broadcast its own routing information on those same networks. In addition, the routing process only advertises the (sub)nets listed in the NETWORK command. The IGRP routing protocol also has an autonomous system number, usually assigned by the NIC.

Configuring the Exterior Routing Protocols

The exterior routing protocols require three sets of information before routing can begin:

- A list of neighbor (or peer) routers with which to exchange routing information. This list is created with the NEIGHBOR router subcommand.
- A list of networks to advertise as directly reachable, created with the NETWORK router subcommand.
- The AS number of the local router.

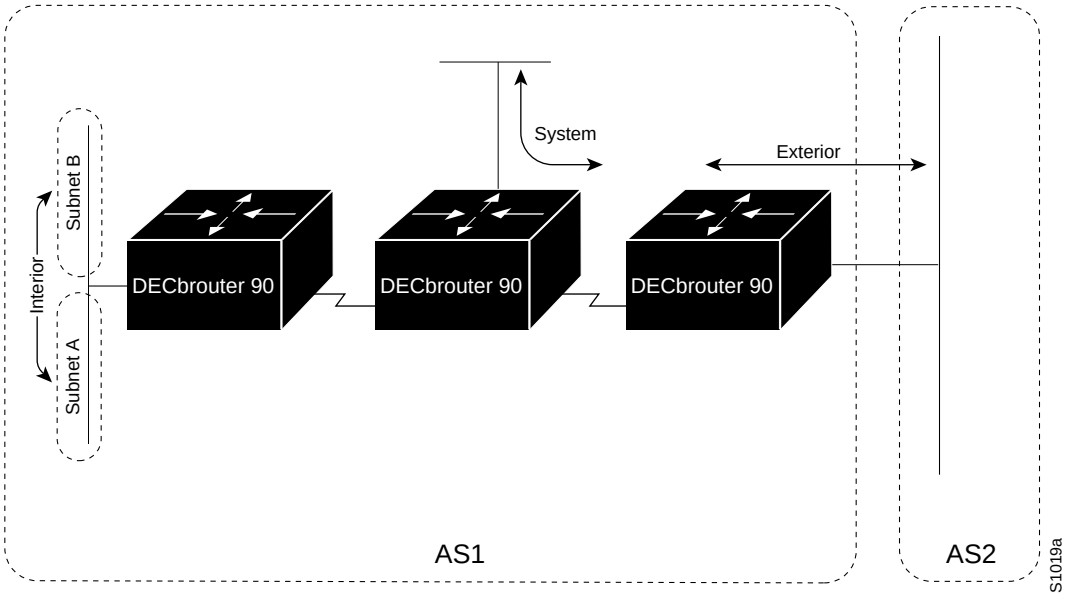
Configuring IGRP

Cisco Systems designed the interior gateway routing protocol (IGRP) for routing in an autonomous system containing arbitrarily complex topology and media with diverse bandwidth and delay characteristics. IGRP can advertise all connected and IGRP-derived networks for a particular autonomous system. The DECbrouter 90 fully supports IGRP.

Interior, System, and Exterior Routes

IGRP advertises three types of routes: interior, system, and exterior, as shown in Figure 6–2. Interior routes are routes between subnets in the network attached to a router interface. If the network attached to a router is not subnetted, IGRP does not advertise interior routes.

Figure 6–2 Interior, System, and Exterior Routes



System routes are routes to networks within the autonomous system. The router derives system routes from directly connected network interfaces and system route information provided by other IGRP-speaking routers. System routes do not include subnetting information. Exterior routes are routes to networks outside the autonomous system that are considered when identifying a gateway of last resort (see Choosing the Gateway of Last Resort in this chapter).

Creating the IGRP Routing Process

To create the routing process, use the ROUTER global configuration command. The full syntax of this command follows.

router igrp *autonomous-system*
no router igrp *autonomous-system*

The argument *autonomous-system* identifies the routes to other IGRP routers and is used to tag the routing information. Use the NO ROUTER IGRP command to shut down the routing process on the AS specified by the *autonomous-system* argument.

Next, specify the list of networks with the NETWORK router configuration subcommand.

The full syntax of this command is as follows:

network *network-number*
no network *network-number*

The argument *network-number* is a network number in dotted IP notation (of directly connected networks). Note that this number must not contain subnet information. You can specify multiple NETWORK subcommands.

The NETWORK router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

Use the NO NETWORK command with the network number to remove a network from the list.

Example

In this example, a router is configured for IGRP and assigned to AS 109. In the next two lines, two NETWORK commands indicate the networks directly connected to the router.

```
router igrp 109
network 131.108.0.0
network 192.31.7.0
```

Unequal-Cost Load Balancing

IGRP has been enhanced to simultaneously use an asymmetric set of paths for a given destination. This feature is known as *unequal-cost load balancing*. The following general rules apply to IGRP unequal-cost load balancing:

- IGRP will accept up to four paths for a given destination network.
- The next router must be closer (have a smaller metric value) to the destination than this router.
- The metric must be within the configured *variance* of the local best metric.

Note

By using *variance*, the router can balance traffic across *all* feasible paths and can immediately converge to a new path if one of the paths should fail.

IGRP Variance Command

IGRP can balance traffic across multiple routes that have different metrics. The amount of load balancing that is performed can be controlled with the VARIANCE router subcommand. The command syntax is as follows:

variance *multiplier*
no variance

The argument *multiplier* defines the range of metric values that will be accepted for load balancing. Acceptable values are nonzero, positive integers. By default, the amount of variance is set to one (equal-cost load balancing). The **no** version resets variance to one.

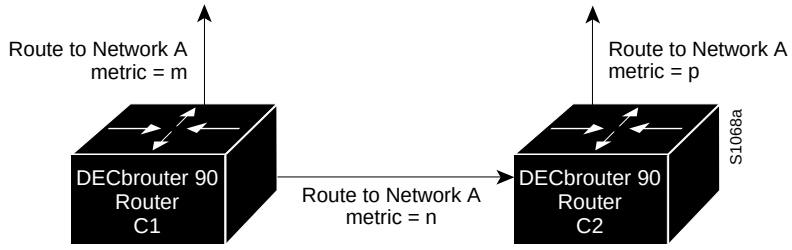
This value is used in the procedure for determining the *feasibility* of a potential route. A route is *feasible* if the next router in the path is closer to the destination than the current router and if the metric for the entire path is *within* the variance. Only paths that are feasible can be used for load balancing and included in the routing table. The two feasibility conditions are:

1. The local best metric must be greater than the best metric learned from the next router.
2. The *multiplier* times the local best metric for the destination must be larger than the metric through the next router

If both these conditions are met, the route is deemed feasible and can be added to the routing table.

Figure 6–3 illustrates the process of determining IGRP path feasibility.

Figure 6–3 Determining IGRP Path Feasibility



The feasibility test would work as follows:

Assume that C1 already has a route to Network A with metric m and has just received an update about Network A from C2. The best metric at C2 is p. The metric that C1 would use through C2 is n.

- If *m* is greater than *p*, then the first condition is met.
- If the *multiplier* times *m* is greater than or equal to *n*, then the second condition is met.

If both conditions are met, the route will be included in C1's routing table. A maximum of four paths can be in the routing table for a single destination. If there are more than four feasible paths, the four best feasible paths are used.

These conditions limit the number of cases in which load balancing can occur, but ensure that the dynamics of the network will remain stable.

Choosing the Gateway of Last Resort

The router chooses a *gateway of last resort* from the list of exterior routes that IGRP provides. The router uses the gateway (router) of last resort if it does not have a better route for a packet and the destination is not a connected network. If the autonomous system has more than one connection to an external network, different routers may choose different exterior routers as the gateway of last resort.

IGRP Metric Information

IGRP uses several types of metric information. For each path through an autonomous system, IGRP records the segment with the lowest bandwidth, the accumulated delay, the smallest maximum transmission unit (MTU), and the reliability and load.

The IGRP metric is a 32-bit quantity that is a sum of the segment delays and the lowest segment bandwidth (scaled and inverted) for a given route. For a network of homogeneous media, this metric reduces to a hop count. For a network of mixed media (FDDI, Ethernet, and serial lines running from 9600 bps to T1 rates), the route with the lowest metric reflects the most desirable path to a destination.

IGRP Updates

A router running IGRP sends an update broadcast every 90 seconds. It declares a route inaccessible if it does not receive an update from the first router in the route within three update periods (270 seconds). After seven update periods (630 seconds), the router removes the route from the routing table. IGRP uses flash update and poison reverse to speed up the convergence of the routing algorithm.

Configuring the OSPF Routing Protocol

This section describes the open shortest path first (OSPF) routing protocol and provides the following specific discussions:

- Overview of the OSPF routing environment and conventions
- DECbrouter 90 support of OSPF
- Steps in configuring OSPF for DECbrouter 90 and details about Digital's OSPF configuration commands

Note

The introductory information in this section summarizes descriptions and definitions provided in the Internet specification of OSPF Version 2 in RFC 1247. This synopsis focuses on linking the DECbrouter 90 configuration command environment to the generalized OSPF capabilities; however, if you are configuring an OSPF-based internetworking scheme, refer to RFC 1247 for specific details. In general, the emphasis in this section is on the DECbrouter 90 implementation of OSPF.

The OSPF SHOW command descriptions, DEBUG command definitions, and OSPF configuration examples are provided in subsequent sections of this chapter, along with similar descriptions and examples for other IP routing protocols.

The OSPF Routing Protocol

OSPF is an interior gateway protocol (IGP). As such, OSPF distributes routing information between routers belonging to a single AS. For the purposes of OSPF, an autonomous system is essentially a group of routers exchanging routing information via a common routing protocol. The OSPF protocol is based on shortest-path-first, or *link-state*, technology. This is a departure from the Bellman-Ford base distance vector technology used by traditional IP routing protocols such as IGRP.

The OSPF protocol was developed by the OSPF working group of the Internet Engineering Task Force (IETF). It has been designed expressly for the Internet environment and includes explicit support for IP subnetting, type of service (TOS)-based routing, and tagging of externally derived routing information. OSPF also provides for packet authentication and uses IP multicast when sending/receiving packets. The OSPF (Version 2) protocol is documented in the Internet RFC 1247.

Note

The DECbrouter 90 currently supports only TOS 0.

The OSPF Routing Domain and Areas

OSPF allows collections of contiguous networks and hosts to be grouped together. Such a group, together with the routers that have interfaces to any one of the included networks, is called an area. Each area runs a separate copy of the basic shortest-path-first routing algorithm. This means that each area has its own topological database.

The topology of an area is invisible from outside the area. Conversely, routers internal to a given area know nothing of the detailed topology external to the area. This isolation of knowledge enables the protocol to effect a marked reduction in routing traffic as compared to treating the entire autonomous system as a single SPF domain.

With the introduction of areas, it is no longer true that all routers in the AS have an identical topological database. A router actually has a separate topological database for each area to which it is connected. Routers connected to multiple areas are called *area border routers*. Two routers belonging to the same area have, for that area, identical area topological databases.

Routing in the autonomous system takes place on two levels, depending on whether the source and destination of a packet reside in the same area (intra-area routing is used) or different areas (interarea routing is used). In intra-area routing, the packet is routed solely on information obtained within the area; no routing information obtained from outside the area can be used. This protects intra-area routing from the injection of bad routing information.

OSPF Backbones

Every OSPF routing domain AS must have a *backbone*. The backbone is a special OSPF area that must have an area ID of 0.0.0.0 (or simply 0). It consists of those networks not contained in any specific area, their attached routers, and those routers that belong to multiple areas. The backbone must be contiguous. Each router's interface that is configured in Area 0 must be reachable via other routers where each interface in the path is configured as being in Area 0.

However, it is possible to define areas in such a way that the backbone is no longer contiguous—where the continuity between routers is broken. In this case, you must establish backbone continuity by configuring *virtual links*. Virtual links are useful when the backbone area is either purposefully partitioned or when restoring inadvertent breaks in backbone continuity.

OSPF Router Classifications

When the AS is split into OSPF areas, the routers are further divided according to function into the following four overlapping categories:

- **Internal**—A router with all directly connected networks belonging to the same area. Routers with only backbone interfaces also belong to this category. These routers run a single copy of the basic routing algorithm.
- **Area border**—A router that attaches to multiple areas. Area border routers run multiple copies of the basic algorithm, one copy for each attached area and an additional copy for the backbone. Area border routers condense the topological information of their attached areas for distribution to the backbone. The backbone in turn distributes the information to the other areas.
- **Backbone**—A router that has an interface to the backbone. This includes all routers that interface to more than one area (area border routers). However, backbone routers are not required to be area border routers. Routers with all interfaces connected to the backbone are considered to be internal.
- **AS boundary**—A router that exchanges routing information with routers belonging to other ASes. Such a router has AS external routes that are advertised throughout the AS. The path to each AS boundary router is known by every router in the AS. This classification is completely independent of the previous classifications; AS boundary routers can be internal or area border routers, and may or may not participate in the backbone.

OSPF Routing Conventions

All OSPF routing implementations adhere to an essentially common set of rules and conventions. The following descriptions outline these conventions and, where applicable, specific characteristics of the DECbrouter 90 implementation.

OSPF Physical Network Support

OSPF routing implementations support the following types of physical networks:

- Point-to-point—A network that joins a single pair of routers, such as a 56-kbps serial line connecting a remote site to a main campus.
- Broadcast—Networks supporting more than two attached routers together that can broadcast a single physical message to all of the attached routers. Neighboring routers are discovered dynamically on these nets using OSPF's *Hello protocol*. The Hello protocol itself takes advantage of the broadcast capability. The protocol makes further use of multicast capabilities, if they exist. Ethernet is an example of a broadcast network type.
- Nonbroadcast—Networks supporting more than two routers, but having no broadcast capability. Neighboring routers are also discovered on these networks using OSPF's Hello protocol. However, due to the lack of broadcast capability, some configuration information is necessary for the correct operation of the Hello protocol. On these networks, OSPF protocol packets that are normally multicast need to be sent to each neighboring router, in turn. An X.25 Public Data Network (PDN) is an example of a nonbroadcast network.

Support for IP Subnetting with OSPF

OSPF attaches an IP address mask to each advertised route. The mask indicates the range of addresses being described by the particular route. For example, a summary advertisement for the destination 128.185.0.0 with a mask of 255.255.0.0 actually is describing a single route to the collection of destinations 128.185.0.0 to 128.185.255.255. Including the mask with each advertised destination enables the implementation of *variable-length subnet masks*.

The OSPF *area* concept is modeled after an IP subnetted network. OSPF areas have been loosely defined to be a collection of networks. More realistically, an OSPF area is a list of address ranges. Each address range is defined as an *address*/*mask* pair. Many separate networks can be contained in a single address range, just as a subnetted network is composed of many separate subnets. Area border routers then summarize the area contents (for distribution to the backbone) by advertising a single route for each address range. The cost of the route is the minimum cost to any of the networks falling in the specified range.

Intra-Area Routing

When a source and destination reside within the same area, routing is said to be *intra-area*. The router sends *Hello* packets to its neighbors, and in turn receives their Hello packets.

The router will attempt to form *adjacencies* with some of its newly acquired neighbors. Topological databases are synchronized between pairs of *adjacent routers*. On multiaccess networks, the *designated router* determines which routers should become adjacent.

Adjacencies control the distribution of routing protocol packets. Routing protocol packets are sent and received only on adjacencies. In particular, distribution of topological database updates proceeds along adjacencies.

Interarea Routing

When a packet's source and destination reside in different areas, routing is *interarea*. For interarea routing, *area border routers* use the same basic routing strategy as with intra-area routing, but also inject additional routing information into an area. This additional information is a distillation of the rest of the AS's topology.

External Routing

Routers that have information regarding other ASes can flood this information throughout an AS. This external routing information is distributed verbatim to every participating router. There is one exception: external routing information is not flooded into *stub areas* (described in the next section).

To use external routing information, the path to all routers advertising external information must be known throughout the AS (not including stub areas). For that reason, the locations of AS boundary routers are summarized by nonstub area border routers.

OSPF Support of Stub Areas

OSPF allows certain areas to be configured as *stub areas*. A stub area can be defined as any area that does not allow the advertisement of external routes. In other words, external advertisements are not flooded into or throughout stub areas; routing to AS external destinations in these areas is based on a per-area default only. This reduces the topological database size and the memory requirements associated with a stub area's internal routers.

In order to take advantage of the OSPF stub area support, *default routing* must be used in the stub area.

Neighbors and Adjacency

OSPF creates adjacencies between *neighboring routers* to facilitate exchange of routing information. *Neighboring routers* are two routers that have interfaces to a common network. On multiaccess networks, neighbors are dynamically discovered by OSPF's Hello protocol. An *adjacency* is a relationship formed between selected neighboring routers for the purpose of exchanging routing information.

Not every pair of neighboring routers becomes adjacent. Instead, adjacencies are established with some subset of the router's neighbors. Routers connected by point-to-point networks and virtual links always become adjacent. On multiaccess networks, all routers become adjacent to both the designated router and the backup designated router (defined later in this section).

The Hello Protocol

The Hello protocol is responsible for establishing and maintaining neighbor relationships. It also ensures that communication between neighbors is bidirectional. Hello packets are sent periodically out all router interfaces. Bidirectional communication is indicated when the router sees itself listed in the neighbor's Hello Packet.

On multiaccess networks, the Hello protocol elects a designated router for the network. Among other things, the designated router controls what adjacencies will be formed over the network (see the next section).

Designated Routers

Every multiaccess network has a *designated router* that performs two main functions for the routing protocol:

- Originates a *network links advertisement* on behalf of the network. This advertisement lists the set of routers, including the designated router, currently attached to the network.
- Becomes adjacent to all other routers on the network. Since the link-state databases are synchronized across adjacencies (through adjacency initialization and the flooding procedure), the designated router plays a central part in the synchronization process.

Virtual Links

The backbone area (area ID = 0) *cannot* be disconnected from any area, or some areas of the AS will become unreachable. To establish or maintain connectivity of the backbone, *virtual links* can be configured through nonbackbone areas. Virtual links connect separate components of the backbone (for example, two areas of the same AS). The two endpoints of a virtual link are *area border routers*. The virtual link must be configured in both routers. The configuration information in each router consists of the other virtual endpoint (the other area border router), and the nonbackbone area the two routers have in common (called the *transit area*).

Note

Virtual links cannot be configured through stub areas.

The DECbrouter 90 OSPF Implementation

The sections that follow detail the specific router and interface subcommands used to enable and configure the OSPF IP routing protocol on the DECbrouter 90. The DECbrouter 90 implementation conforms to the OSPF Version 2 specifications detailed in Internet RFC 1247. The list that follows outlines key features supported in the DECbrouter 90 OSPF implementation.

- **Stub areas**—Definition of stub areas is supported using the STUB and DEFAULT-COST options of the AREA router subcommand.
- **Route redistribution**—Routes learned via any IP routing protocol can be redistributed into any other IP routing protocol. At the intradomain level, this means that OSPF can import routes learned via IGRP, RIP, and Hello. OSPF routes also can be exported into IGRP, RIP, and Hello. At the interdomain level, OSPF can import routes learned via EGP and BGP. OSPF routes can be exported into EGP and BGP. Redistribution is enabled with the REDISTRIBUTE and DEFAULT-INFORMATION router subcommands
- **Authentication**—Authentication among neighboring routers within an area is supported using two commands. An AREA router subcommand enables authentication, while the IP OSPF AUTHENTICATION-KEY interface subcommand specifies the password used in a specific area or on a specific interface by OSPF routers.
- **Router interface parameters**—Router interface parameters are configured using interface subcommands. Configurable parameters supported include interface output cost, retransmission interval, interface transmit delay, router priority, Hello protocol interval, router "dead" interval, and authentication key. Each of these is configured using various options of the IP OSPF interface subcommand.
- **Nonbroadcast networks**—Digital routers support OSPF over nonbroadcast networks using the NEIGHBOR router subcommand. The NEIGHBOR subcommand allows specification of three parameters:
  - — The priority for a neighboring router
  - — The nonbroadcast poll interval
  - — The interface through which the neighbor is reachable
- **Virtual links**—Virtual links are created using the VIRTUAL-LINK option of the AREA router subcommand.

Steps in Configuring OSPF Routing

Configuring an OSPF routing process involves the following general steps. Step 1 is mandatory; the other steps are optional but may be required for your specific application.

1. Enable OSPF using the ROUTER OSPF global command and NETWORK router subcommand.
2. Specify route redistribution, as needed (addressed separately with redistribution discussion for all IP routing protocols).
3. Set any OSPF area parameters, as needed.

4. Set interface parameters, as needed.
5. For nonbroadcast networks, specify appropriate neighbor parameters and the router's polling interval, as required.

The commands for each of these steps are detailed in the sections that follow.

Enabling the OSPF Routing Processes and Defining Areas

To start an OSPF routing process, you must enable OSPF in the router, specify the range of IP addresses to be associated with the routing process, and assign area IDs to be associated with that range of IP addresses. Two commands are associated with this initial step: ROUTER OSPF (a global configuration command) and NETWORK (a router subcommand).

Enabling OSPF Routing

The first step in setting up OSPF support on a router is to enable OSPF using the ROUTER OSPF global configuration command. The syntax for this command follows.

router ospf *ospf-process-id* **no router ospf** *ospf-process-id*

The argument *ospf-process-id* is an internally used identification parameter. It is locally assigned and can be any positive integer. A unique value is assigned for each OSPF routing process. You can specify multiple OSPF routing processes in each router.

The NO ROUTER OSPF command must be specified with the *ospf-process-id* and terminates individual OSPF routing processes in the router.

Defining OSPF on Networks and Assigning Area IDs

Use the NETWORK router subcommand to define the interfaces on which OSPF runs and to define the area ID for those interfaces. The general syntax for this command is as follows:

network *address wildcard-mask* **area** *area-id*
no network *address wildcard-mask* **area** *area-id*

The NETWORK router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

The *address* and *wildcard-mask* arguments together allow you to define one or multiple interfaces to be associated with a specific OSPF area using a single command. The argument *address* is formed as an IP address. The argument *wildcard-mask* is an IP-address-type mask that includes "don't care" bits.

The keyword/variable argument pair **area** *area-id* specifies an area to be associated with the OSPF address range as defined in the same NETWORK command. The argument *area-id* can be specified as either a decimal value or as an IP address. If you intend to associate areas with IP subnets, you can specify a subnet address as the *area-id*.

Note

Any individual interface can only be attached to a single area. If the address ranges specified for different areas overlap, the router will adopt the first area in the NETWORK subcommand list and ignore the subsequent overlapping portions. In general, it is recommended that you devise address ranges that do not overlap in order to avoid inadvertent conflicts.

The NO NETWORK router subcommand must be specified with the complete address range and area ID; it disables OSPF routing for interfaces defined with the *address wildcard-mask* pair.

Example

The ROUTER OSPF and NETWORK commands defined in this section control the assignment of area IDs to specific address ranges. The following example illustrates the assignment of four area IDs to four IP address ranges.

In the following example, OSPF routing process 109 is initialized, and four OSPF areas are defined: 10.9.50.0, 2, 3, and 0. Areas 10.9.50.0, 2, and 3 mask specific address ranges, while Area 0 enables OSPF for *all other* networks.

```
router ospf 109
network 131.108.20.0  0.0.0.255 area 10.9.50.0
network 131.108.0.0  0.0.255.255 area 2
network 0.0.0.0  255.255.255.255 area 0
!
! Interface Ethernet0 is in area 10.9.50.0:
interface Ethernet 0
ip address 131.108.20.5 255.255.255.0
!
! Interface Serial 0 is in area 2:
interface Serial 0
ip address 131.108.1.5 255.255.255.0
!
! Interface Serial 1 is in area 0:
interface Serial 1
ip address 131.109.1.1 255.255.255.0
```

Each NETWORK subcommand is evaluated sequentially, so the specific order of these commands in the configuration is important. The router sequentially evaluates the *address*/*wildcard-mask* pair for each interface as follows:

1. The *wildcard-mask* is logically ORed with the interface IP address.
2. The *wildcard-mask* is logically ORed with the *address* in the NETWORK command.
3. The router compares the two resulting values.
4. If they match, OSPF is enabled on the associated interface and this interface is attached to the OSPF area specified.

Consider the first NETWORK subcommand. Area ID 10.9.50.0 is configured for the interface on which subnet 131.108.20.0 is located. Assume that a match is determined for interface Ethernet 0. Interface Ethernet 0 is attached to Area 10.9.50.0 only.

The second NETWORK subcommand is evaluated next. For Area 2, the same process is then applied to all interfaces (except interface Ethernet 0). Assume that a match is determined for interface serial 0. OSPF is then enabled for that interface and serial 0 is attached to Area 2.

This process of attaching interfaces to OSPF areas continues for all network subcommands. Note that the last NETWORK subcommand in this example is a special case. With this command all available interfaces (not explicitly attached to another area) are attached to Area 0.

Configuring OSPF Area Parameters

The AREA router subcommand allows control of various area parameters as defined by RFC 1247. This router subcommand also allows you to control certain special capabilities (such as stub areas and virtual links). The discussions that follow outline the various options for the AREA subcommand and summarize their applications.

Note

The specification of the NO AREA AREA-ID router subcommand (with no other keywords or arguments) removes the specified area from the router's configuration. The *area-id* argument is defined in more detail in the following AREA router subcommand descriptions.

Setting Simple OSPF Area Authentication

In general, authentication is configured on a per-area basis. It is enabled for an area using the **authentication** option of the AREA router subcommand. The syntax for this command is as follows:

area *area-id* **authentication**
no area *area-id* **authentication**

The argument *area-id* is the specific area ID of the area for which authentication is to be enabled. The argument *area-id* can be specified as either a decimal value or as an IP address. Specifying authentication for an area sets the authentication to Type 1 (simple password). If this command is not included in the configuration for a router, authentication is of Type 0 (no authentication).

The authentication type must be the same for all routers in an area. The authentication key (password) for all OSPF routers on a network must be the same if they are to communicate with each other via OSPF. Use the IP OSPF AUTHENTICATION-KEY interface subcommand to specify this password.

The **no area** *area-id* authentication option removes the area's authentication specification.

Defining a Stub Area

Define an area as a stub area using two router subcommands: the **stub** and **default-cost** options of the AREA router subcommand. These commands are used only on an area border router attached to a stub. The syntax for the AREA *area-id* STUB router subcommand is as follows:

area *area-id* **stub**
no area *area-id* **stub**

The argument *area-id* is the specific area ID for the stub area. It can be specified as either a decimal value or an IP address.

The **stub** option is used to enable the stub area.

The NO AREA *area-id* STUB option removes the specified area as a stub area.

The syntax for the AREA *area-id* DEFAULT-COST *cost* router subcommand is as follows:

area *area-id* **default-cost** *cost*
no area *area-id* **default-cost** *cost*

The argument *area-id* is the specific area ID for the stub area. It can be specified as either a decimal value or an IP address.

The **default-cost** *cost* keyword/argument pair assigns a specific cost for the default external route used for the stub area. The acceptable value is a 24-bit number. The **no area** *area-id* **default-cost** *cost* removes the assigned default route cost.

Consolidating Advertised Addresses

Routing information is condensed at area boundaries. External to the area, a single route is advertised for each address range. To consolidate or summarize routes, you can use the **range** option of the AREA router subcommand. The result is that a single summary route is advertised to other areas by the area border router. This command is only used with area border routers. The syntax for this router subcommand is as follows:

area *area-id* **range** *address mask*
no area *area-id* **range** *address mask*

The argument *area-id* is the specific area ID for the area about which routes are to be summarized. The argument *area-id* can be specified as either a decimal value or an IP address.

Note

Multiple AREA router subcommands specifying the **range** option can be specified. Thus, OSPF can summarize addresses for many different sets of address ranges.

The *address* argument is a standard IP address. The *mask* argument is a standard IP mask.

Configuring OSPF Interface-Specific Parameters

The interface subcommands described in this section define how you can configure each OSPF router interface parameter. In general, you do not need to configure any of these parameters; however, some interface parameters must be consistent across all routers in an attached network. Be sure that if you do configure any of these parameters, the configurations for all routers on the network have compatible values.

Specifying OSPF Path Cost

To explicitly specify the cost of sending a packet on an interface, use the IP OSPF COST interface subcommand. The syntax for this command is as follows:

ip ospf cost *cost* **no ip ospf cost** *cost*

The argument *cost* is expressed as the link state metric. It is a dimensionless integer value that is always greater than zero. This value is advertised as the link cost in the router's router links advertisement. Type of Service (TOS) is not supported, so you can assign only one cost per interface.

Different physical interfaces have different defaults, as follows:

- 56-kbps serial link—default cost is 1785
- 64-kbps serial link—default cost is 1562
- T1 (1.544-Mbps serial link)—default cost is 65
- E1 (2.048-Mbps serial link)—default cost is 48
- Ethernet—default cost is 10

In general, the path cost is calculated as follows:

$$\frac{10^8}{Bandwidth}$$

The NO IP OSPF COST interface subcommand resets the path cost for an interface to the default value.

Setting the Link State Retransmission Interval

The number of seconds between link state advertisement retransmissions for adjacencies belonging to the interface is specified with the IP OSPF RETRANSMIT-INTERVAL interface subcommand. The syntax for this command is as follows:

ip ospf retransmit-interval *number-of-seconds*
no ip ospf retransmit-interval *number-of-seconds*

The value for the *number-of-seconds* argument is an integer number that should be greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links.

The default value is five seconds.

The NO IP OSPF RETRANSMIT-INTERVAL subcommand resets the link state advertisement retransmission interval to the default value.

Setting the Transmission Time for Link State Updates

To set the estimated number of seconds it takes to transmit a link state update packet on the interface, use the IP OSPF TRANSMIT-DELAY interface subcommand. The syntax for this command is as follows:

ip ospf transmit-delay *number-of-seconds*
no ip ospf transmit-delay *number-of-seconds*

Link state advertisements in the update packet must have their age incremented by this amount before transmission. The value assigned should take into account the transmission and propagation delays for the interface.

The argument *number-of-seconds* is an integer value that must be greater than zero. The default value is one second.

The NO IP OSPF TRANSMIT-DELAY subcommand resets the estimated transmission time for the link state update packet to the default value.

Setting Router Priority

The router priority is used to help determine the designated router for a network. To set the router priority, use the IP OSPF PRIORITY interface subcommand. The syntax for this command is as follows:

ip ospf priority *8-bit-number*
no ip ospf priority *8-bit-number*

The argument *8-bit-number* is an 8-bit unsigned integer.

When two routers attached to a network both attempt to become the designated router, the one with the highest router priority takes precedence. If there is a tie, the router with the highest router ID takes precedence. A router with a router priority set to zero is ineligible to become the designated router or backup designated router. Router priority is only configured for interfaces to multiaccess networks (in other words, not point-to-point networks).

The default router priority is 0.

The NO IP OSPF PRIORITY subcommand resets the router priority to the default value.

Setting the Advertised Hello Interval

Use the IP OSPF HELLO-INTERVAL interface subcommand to specify the length of time, in seconds, between the Hello packets that the router sends on the interface. The syntax for this command is as follows:

ip ospf hello-interval *number-of-seconds*
no ip ospf hello-interval *number-of-seconds* no ip ospf hello-interval number-of-seconds

The argument *number-of-seconds* is an unsigned integer value. This value is advertised in the router's Hello packets. It must be the same for all routers attached to a common network. The smaller the Hello interval, the faster topological changes will be detected, but more routing traffic will ensue.

The default differs for broadcast and nonbroadcast networks. The default for broadcast networks is 10 seconds. The default for nonbroadcast networks (for example, X.25, Frame Relay, and SMDS) is 30 seconds (determined automatically when any of the serial encapsulations are specified).

Note

The **hello-interval** specification must be the same for all nodes on a specific network.

The NO IP OSPF HELLO-INTERVAL subcommand resets the length of time between Hello packet transmissions on an interface to the default value.

Setting the Router Dead Interval

Use the IP OSPF DEAD-INTERVAL interface subcommand to set the number of seconds that a router's Hello packets have not been seen before its neighbors declare the router down. The command syntax follows.

ip ospf dead-interval *number-of-seconds*
no ip ospf dead-interval *number-of-seconds*

The argument *number-of-seconds* is an unsigned integer value. The default is four times the IP OSPF HELLO-INTERVAL value. As with the Hello interval, this value must be the same for all routers attached to a common network.

The NO IP OSPF DEAD-INTERVAL subcommand resets the length of time that a router's Hello packets have not been seen before its neighbors declare the router down to the default value.

The *number-of-seconds* specified must be the same for all nodes on a network.

Specifying the OSPF Authentication Key

Use the IP OSPF AUTHENTICATION-KEY interface subcommand to assign a specific password to be used by neighboring routers on a wire that are using OSPF's simple password authentication. The command syntax follows.

ip ospf authentication-key *8-bytes-of-password*
no ip ospf authentication-key *8-bytes-of-password*

The argument *8-bytes-of-password* is any continuous string of characters that you can enter from the keyboard up to eight bytes in length.

Note

A router will use this key only when authentication is enabled for an area with the AREA *area-id* AUTHENTICATION router subcommand.

This key is inserted directly into the OSPF header when originating routing protocol packets. A separate password can be assigned to each network on a per-interface basis. All neighboring routers on the same network must have the same password to be able to route OSPF traffic.

There is no default value. The NO OSPF AUTHENTICATION-KEY interface subcommand removes any OSPF password that was previously assigned.

Configuring OSPF for Nonbroadcast Networks

OSPF treats a nonbroadcast, multiaccess network (X.25, Frame Relay, or SMDS) much like it treats a broadcast network. Since there may be many routers attached to the network, a designated router is selected for the network. This designated router then originates a networks links advertisement, which lists all routers attached to the nonbroadcast network.

However, due to the lack of broadcast capabilities, it is necessary to use special configuration parameters in the designated router selection. These parameters need only be configured in those routers that are themselves eligible to become the designated router or backup designated router (in other words, routers with a positive, nonzero router priority value).

Use the NEIGHBOR router subcommand to configure routers interconnecting to nonbroadcast networks. The syntax for this command is as follows:

neighbor *address* **interface** *type unit-number* [**priority***8-bit-number*] [**poll-interval** *number-of-seconds*]

no neighbor *address* **interface** *type unit-number* [**priority***8-bit-number*] [**poll-interval** *number-of-seconds*]

One neighbor entry must be included in the router's configuration for each known nonbroadcast network neighbor.

The argument *address* is the specific interface IP address of the neighbor.

The keyword/argument sequence **interface** *type unit-number* identifies the specific router interface for this NEIGHBOR command. The interface must be connected to a nonbroadcast, multiaccess type network. In addition, the neighbor specified must be eligible to be a designated router or backup designated router.

The keyword/argument pair **priority** *8-bit number* is the router priority value of the nonbroadcast neighbor associated with the IP address specified.

If a neighboring router has become inactive (Hello packets have not been seen for router dead interval period), it may still be necessary to send Hello packets to the dead neighbor. These Hello packets will be sent at a reduced rate called the *poll interval*.

Specify the poll interval with the keyword/argument pair **poll-interval** *number-of-seconds*. The argument *number-of-seconds* is an unsigned integer value. RFC 1247 recommends that this value should be much larger than the Hello interval. The default is 2 minutes (120 seconds).

The NO NEIGHBOR router subcommand requires specification of the neighbor's IP address; this command removes the specific neighbor from the list.

Creating Virtual Links

In OSPF, all areas must be connected to a backbone area. If the connection to the backbone is lost, it can be repaired by establishing a *virtual link*. Virtual links are defined using the **virtual link** option of the AREA router subcommand. The general syntax for this subcommand is as follows:

area *area-id* **virtual-link** *router-id* [**hello-interval** *number-of-seconds*]
[**retransmit-interval** *number-of-seconds*]
[**transmit-delay** *number-of-seconds*]
[**dead-interval** *number-of-seconds*]
[**authentication-key** *8-bytes-of-password*]

no area *area-id* **virtual-link** *router-id* [**hello-interval** *number-of-seconds*]
[**retransmit-interval** *number-of-seconds*]
[**transmit-delay** *number-of-seconds*]
[**dead-interval** *number-of-seconds*]
[**authentication-key** *8-bytes-of-password*]

The argument *area-id* is the area ID assigned to the transit area for the virtual link. This can be either a decimal value or a valid IP address.

The argument *router-id* is the router ID associated with the virtual link neighbor. The router id appears in the **show ip ospf** display. It is internally derived by each router from the router's interface IP addresses. This value must be entered in the format of an IP address.

There are no default values for the *area-id* or *router-id* arguments for this command.

Note

Each virtual link neighbor must include the transit area ID and the corresponding virtual link neighbor's router id in order for a virtual link to be properly configured.

Specify the length of time in seconds between the Hello packets that the router sends on the interface with the **hello-interval** option of the AREA *area-id* VIRTUAL-LINK router subcommand. The argument *number-of-seconds* is an unsigned integer value. This value is advertised in the router's Hello packets. It must be the same for all routers attached to a common network. The smaller the Hello interval, the faster topological changes will be detected, but more routing traffic will ensue. The default is ten seconds.

Specify the number of seconds between link state advertisement retransmissions for adjacencies belonging to the interface, with the **retransmit-interval** option of the AREA *area-id* VIRTUAL-LINK router subcommand. The value for the *number-of-seconds* argument is an integer that should be greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links. The default value is five seconds.

Specify the estimated number of seconds it takes to transmit a link state update packet on the interface with the **transmit-delay** option of the AREA *area-id* VIRTUAL-LINK router subcommand. Link state advertisements in the update packet must have their age incremented by this amount before transmission. The value assigned should take into account the transmission and propagation delays for the interface. The argument *number-of-seconds* is an integer value that must be greater than zero. The default value is one second.

Set the number of seconds that a router's Hello packets have not been seen before its neighbors declare the router down with the **dead-interval** option of the AREA *area-id* VIRTUAL-LINK router subcommand. The argument *number-of-seconds* is an unsigned integer value. The default is four times the Hello interval. As with the Hello interval, this value must be the same for all routers attached to a common network.

Assign a specific password to be used by neighboring routers with the **authentication-key** option of the AREA *area-id* VIRTUAL-LINK router subcommand. The argument *8-bytes-of-password* is any continuous string of characters that you can enter from the keyboard up to eight bytes in length. This configured data allows the authentication procedure to generate and/or verify the authentication field in the OSPF header. This key is inserted directly into the OSPF header when originating routing protocol packets. A separate password can be assigned to each network on a per-interface basis. All neighboring routers on the same network must have the same password to be able to route OSPF traffic. There is no default value.

Note

A router will use this authentication key only when authentication is enabled for the backbone with the AREA 0 AUTHENTICATION router subcommand.

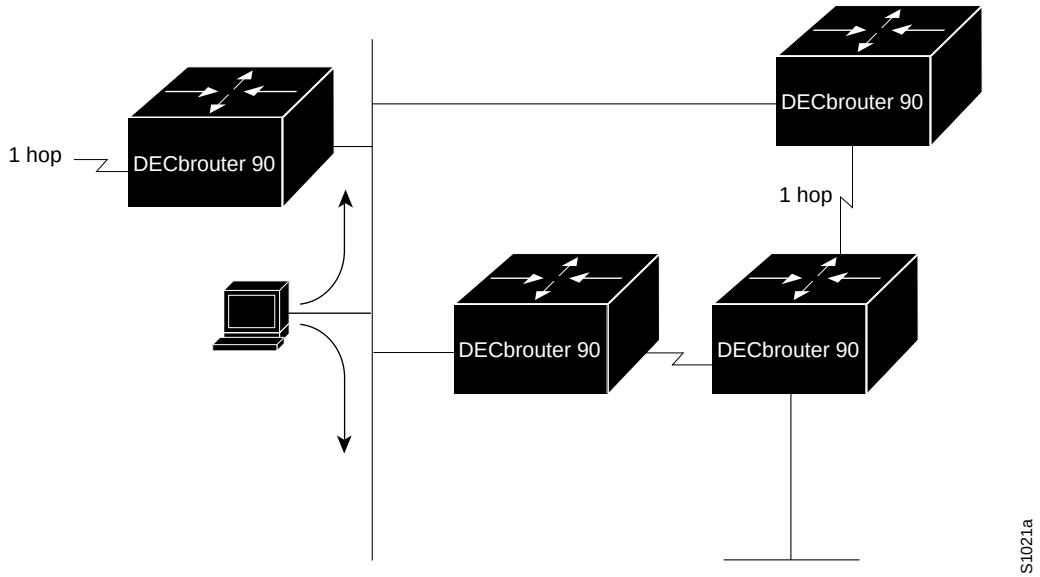
The **no area** *area-id* **virtual-link** *router-id* command removes a virtual link.

Configuring the RIP Protocol

The routing information protocol (RIP) uses broadcast user datagram protocol (UDP) data packets to exchange routing information. Each router sends routing information updates every 30 seconds; this process is termed *advertising*. If a router does not receive an update from another router for 90 seconds or more, it marks the routes served by the nonupdating router as being unusable. If there is still no update after 240 seconds, the router removes all routing table entries for the nonupdating router.

The measure, or metric, that RIP uses to rate the value of different routes is the hop count. The hop count is the number of routers that can be traversed in a route. A directly connected network has a metric of zero (see Figure 6–4; an unreachable network has a metric of 16. This small range of metrics makes RIP unsuitable as a routing protocol for large networks. If the router has a default network path, RIP advertises a route that links the router to the pseudo-network 0.0.0.0. The network 0.0.0.0 does not exist; RIP treats 0.0.0.0 as a network to implement the default routing feature.

Figure 6–4 Hop Count in RIP



Creating the RIP Routing Process

To create a routing process for RIP, use the ROUTER RIP global configuration command:

router rip
no router rip

Use the NO ROUTER RIP command to shut down the routing process.

Specifying the List of Networks

Next, specify the list of networks with the NETWORK router configuration subcommand. The full syntax of this command follows.

network *network-number*
no network *network-number*

The argument *network-number* is a network number in dotted IP notation (of directly connected networks). Note that this number must *not* contain subnet information. You can specify multiple NETWORK subcommands. RIP routing updates will be sent and received only through interfaces on this network.

The NETWORK router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

Example

The following example configuration defines RIP as the routing protocol to be used on all interfaces connected to networks 128.99.0.0 and 192.31.7.0.

```
router rip
network 128.99.0.0
network 192.31.7.0
```

To remove a network from the list, use the **no** network router subcommand followed by the network address.

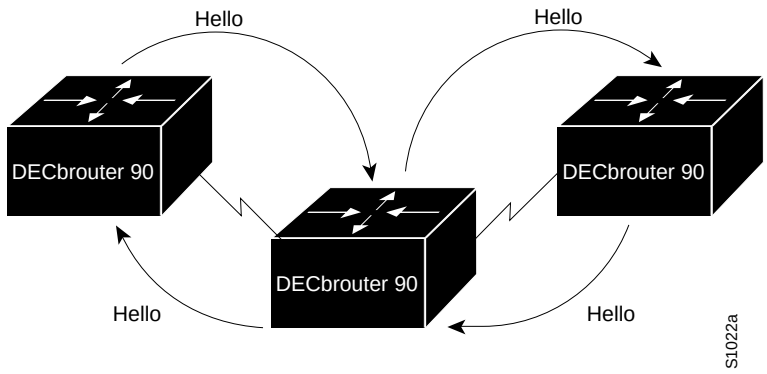
Configuring the Hello Protocol

The Hello protocol, described in RFC 891, was developed for the Fuzzball gateways of the Distributed Computer Network project and was used extensively in the early NSFnet backbone network. Hello is an interior routing protocol.

The DECbrouter 90 implementation of Hello does not implement the extensive time-keeping and delay measurement features. Specifically, the router sets the invalid bit in the Hello date field and clears the time and timestamp fields.

The metric used in Hello is a delay value measured in milliseconds (see Figure 6–5). This metric can range from 0 to 30,000 milliseconds, making Hello a good candidate for routing larger networks. A network with a 30,000-millisecond delay is considered unreachable. The DECbrouter 90 implementation uses a delay of 100 milliseconds for all routes, regardless of their actual delay characteristics.

Figure 6–5 The Hello Protocol



Creating the Hello Routing Process

Create the routing process with the ROUTER global configuration command:

router hello
no router hello

Use the NO ROUTER HELLO command to shut down the routing process.

Specifying the List of Hello Networks

After you have created the routing process, specify the list of networks. This list is specified with the NETWORK router configuration subcommand:

network *network-number*
no network *network-number*

The NETWORK router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

The argument *network-number* is a network number in dotted IP notation (of directly connected networks). Note that this number must not contain subnet information. You can specify multiple NETWORK subcommands.

Example

In the following example, the network 160.1.0.0 is being set up for Hello:

```
router hello
network 160.1.0.0
```

Configuring the BGP Protocol

The border gateway protocol (BGP), as defined in RFC 1267, allows you to set up a distributed routing core that automatically guarantees the loop-free exchange of routing information on an autonomous system (AS) basis.

Creating the BGP Routing Process

To configure BGP, use the ROUTER BGP global configuration command:

router bgp *autonomous-system*
no router bgp *autonomous-system*

The *autonomous-system* number is used to identify the router to other BGP routers and to tag the routing information passed along.

Example

In the following example, a router is assigned to AS 120.

```
router bgp 120
```

Specifying the List of BGP Networks

Use the NETWORK router subcommand to specify networks that are to be advertised as originating within the current AS. These networks can be learned from connected, dynamic routing, and static route sources. The command syntax follows.

network *network-number*
no network *network-number*

The argument *network-number* is the dotted IP address of the network that will be included in the router's BGP updates. The **no** version removes the specified *network-number*.

The NETWORK router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

Note

For exterior protocols, a reference to an IP network from the NETWORK command that is learned by another routing protocol does not require a REDISTRIBUTE command. This is in contrast to interior gateway protocols, such as IGRP, which require the use of the REDISTRIBUTE command.

Example

In the following command, the network 131.108.0.0 is set up to be included in the router's BGP updates.

```
network 131.108.0.0
```

Specifying the List of Neighbors

BGP supports two different kinds of neighbors: internal and external. Internal neighbors reside in the same autonomous system; external neighbors are in other autonomous systems.

BGP routing includes several related router subcommands that specify NEIGHBOR routers and manage your router's relationship with its neighbors. Use the NEIGHBOR router subcommands to:

- Identify BGP peers and their AS numbers
- Assign access lists
- Set up various routing policies

Basic Neighbor Specification

In the simplest case, you want to specify that another router is a neighbor. Use the NEIGHBOR command, as follows:

neighbor *address* **remote-as** *number*
no neighbor *address*

The argument *address* is the neighbor's IP address. The argument *number* is the AS to which the neighbor belongs. Specifying a neighbor with a *number* that matches the AS number specified in the ROUTER BGP command identifies the neighbor as internal to the same AS to which the router belongs.

Using the **no** keyword removes the router as a neighbor. The arguments *address* and *autonomous-system* are the neighbor's address and AS number.

Example 1

This example specifies that the router at the address 131.108.1.2 is a neighbor in AS number 109.

```
neighbor 131.108.1.2 remote-as 109
```

Example 2

In the following example, a BGP router is assigned to AS 109, and two networks are listed as originating in the AS. Then the addresses of three remote routers (and their ASes) are listed. The router being configured will share information about networks 131.108.0.0 and 192.31.7.0 with the neighbor routers. The first router listed is in the same Class B network address space, but in a different AS; the second NEIGHBOR command illustrates specification of an internal neighbor (with the same AS number) at address 131.108.234.2; and the last NEIGHBOR command specifies a neighbor on a different network.

```
router bgp 109
network 131.108.0.0
network 192.31.7.0
neighbor 131.108.200.1  remote-as 167
neighbor 131.108.234.2 remote-as 109

neighbor 150.136.64.19  remote-as  99
```

To control how a BGP process determines which neighbors will be treated as peers, use the NEIGHBOR ANY router subcommand with the ROUTER BGP 0 global subcommand. The syntax for this command is as follows:

neighbor any [*list*]
no neighbor any [*list*]

If the *list* argument is specified, the neighbor must be accepted by the access list number specified to be allowed to peer with the BGP process.

Continuing with the preceding sample configuration, the configuration would look like the following:

```
router bgp 109
network 131.108.0.0
network 192.31.7.0
neighbor 131.108.200.1  remote-as 167
neighbor 131.108.234.2 remote-as 109
neighbor 150.136.64.19  remote-as  99
neighbor any 1
access-list 1 permit 192.31.7.0 0.0.0.255
```

Setting Route Weights

The NEIGHBOR WEIGHT router subcommand specifies a weight to assign to a specific neighbor connection. Its full syntax is as follows:

neighbor *address* **weight** *weight*
neighbor *address* **weight** *weight*

The argument *address* is the address of the neighbor. The argument *weight* is the weight to assign. All routes learned from this neighbor will have this initial weight. The route with the highest weight will be chosen as the preferred route when multiple routes are available to a particular network.

Use the NO NEIGHBOR command with the appropriate arguments and keywords to remove this function.

Example

In this example, the neighbor at address 151.23.12.1 is assigned a weight of 50.

```
neighbor 151.23.12.1 weight 50
```

Filtering BGP Advertisements

You can filter BGP advertisements in two ways: by using access lists, and by using AS-path filters. IP access lists are discussed in Chapter 5 of this manual.

You can apply access lists to BGP updates with the NEIGHBOR DISTRIBUTE-LIST router subcommand. Its full syntax follows.

neighbor *address* **distribute-list** *list* {**in** | **out**}
no neighbor *address* **distribute-list** *list*

The argument *address* is the address of the neighbor. The argument *list* is a predefined access list number. The keywords **in** and **out** specify whether you are applying the access list to incoming or outgoing advertisements to that neighbor. Only standard access lists can be used with this command.

Use the NO NEIGHBOR command with the appropriate arguments and keywords to remove this function.

Example

In the example that follows, list 41 is applied to outgoing advertisements to neighbor 120.23.4.1.

```
neighbor 120.23.4.1 distribute-list 41 out
```

Filtering BGP Routes

You can specify an access list filter on both incoming and outbound BGP routes. In addition, you can assign *weights* based on a set of filters. Each filter is an access list based on regular expressions. Use the IP AS-PATH ACCESS-LIST global configuration command to define an BGP access list, and the NEIGHBOR router subcommand to apply a specific access list.

Defining a BGP Access List

To define a BGP-related access list, use the IP AS-PATH ACCESS-LIST global configuration command. The command syntax is as follows:

ip as-path access-list *list* [**permit** | **deny**] *as-regular-expression*
no ip as-path access-list *list* [**permit** | **deny**] *as-regular-expression*

The *list* argument is an integer from 1 to 99.

Regular expressions are defined in Appendix D, "Pattern Matching." If the regular expression matches the representation of the autonomous system path of the route as an ASCII string, then the **permit** or **deny** condition applies.

Specifying BGP Route Filters

Filters are established with the NEIGHBOR router subcommand, using access lists defined with the IP AS-PATH ACCESS-LIST command. Several variations are allowed. The command syntax follows.

neighbor *address* **filter-list** *list* {**in** | **out** | **weight** *weight*}
no neighbor *address* **filter-list** *list* {**in** | **out** | **weight** *weight*}

The argument *address* is the address of the neighbor.

The argument *list* is a predefined BGP access list number.

One of three alternative keywords also must be used:

- The keyword **in** specifies that you are applying the access list to incoming routes.
- The keyword **out** specifies that you are applying the access list to outgoing routes.
- The keyword/argument pair **weight** *weight* assigns a relative importance to a specific list. The given *weight* is added to the weight of the route if the AS path matches the regular expression. Any number of weight filters are allowed on a per-neighbor basis, but only one in or out filter is allowed. The weight of a route affects BGP's route-selection rules. Acceptable values are 0 to 65535. The default is zero (0).

Example

In the following example, the BGP neighbor with IP address 128.125.1.1 is not sent advertisements about any path through or from the adjacent AS 123.

```
ip as-path access-list 1 deny ^123$
ip as-path access-list 1 deny ^123 .*
! The space in the above expression (^123.*)is required.

router bgp 109
network 131.108.0.0
neighbor 129.140.6.6 remote-as 123
neighbor 128.125.1.1 remote-as 47
neighbor 128.125.1.1 filter-list 1 out
```

Specifying BGP Version Number

BGP now supports Versions 2 and 3 of the protocol and permits dynamic version negotiation with neighbors. Routers can be configured to handle only Version 2 of the protocol using the NEIGHBOR VERSION router subcommand. The command syntax is as follows:

neighbor *ip-address* **version** *value*
no neighbor *ip-address* **version** *value*

The argument *ip-address* is the address of the BGP-speaking neighbor; the version *value* can be set to 2 to force the router to only use Version 2 with the specified neighbor. The default is to use Version 3 of BGP and dynamically

negotiate down to Version 2 if requested. The NO NEIGHBOR *ip-address* VERSION *value* command returns the version to the default state for that neighbor.

Specifying BGP Administrative Distance

BGP allows the use of three possible administrative distances, assigned with the DISTANCE BGP router subcommand. The command syntax follows.

distance bgp *external-distance internal-distance local-distance*
no distance bgp

Use this command if another protocol is known to be able to provide a better route to a node that was actually learned via external BGP or if some local routes should really be preferred by BGP.

Note

Changing the administrative distance of BGP internal routes is considered dangerous and generally is not recommended. One problem that can arise is the accumulation of routing table inconsistencies, which can break routing.

The argument *external-distance* specifies the value for BGP external routes. External routes are routes for which the best path is learned from a neighbor external to the autonomous system. Acceptable values are positive, nonzero integers.

The argument *internal-distance* specifies the value for BGP internal routes. Internal routes are routes that are learned from another BGP entity within the same autonomous system. Acceptable values are positive, nonzero integers.

The argument *local-distance* specifies the value for BGP local routes. Local routes are networks listed with a NETWORK command—possibly as back doors (discussed later in this chapter)—for that router or for networks that are being redistributed from another process.

By default, the administrative distances are as follows:

- *external-distance*—20
- *internal-distance*—200
- *local-distance*—200

The NO DISTANCE BGP command resets these values to the defaults.

Adjusting the BGP Timers

To adjust the default BGP timers, use the TIMERS BGP router subcommand. The full syntax of this command follows.

timers bgp *keepalive holdtime*
no timers bgp

The argument *keepalive* is the frequency in seconds with which the router sends *keepalive* messages to its peer (default 60 seconds), and *holdtime* is the interval

in seconds after not receiving a keepalive message that the router declares a peer dead (default 180 seconds). The NO TIMERS BGP command restores the default.

Example

In this example, the keepalive timer is changed to 70 seconds, and the holdtime is changed to 210 seconds.

```
timers bgp 70 210
```

Clearing BGP Connections

Use the EXEC command CLEAR IP BGP to reset BGP connections. The command syntax is as follows:

clear ip bgp *
clear ip bgp *address*

This command resets the BGP connection with the specified BGP neighbor (identified with the *address* argument). If you specify an asterisk (*), all current BGP sessions are reset.

In general, use this command whenever a policy changes. Changes that might prompt you to use this command are as follows:

- Additions or changes to the BGP-related access lists
- Changes to BGP-related weights
- Changes to BGP-related distribution lists
- Changes in the BGP timer's specifications

BGP and IGP Routing Information

This section discusses the issues of BGP interacting with the various interior gateway protocols (referred to generically as IGPs), such as IGRP, RIP, and Hello. BGP maintains its own routing table separate from the main IP routing table used to make datagram switching decisions. The BGP routing table is organized by network and contains information referred to as *attributes*, such as the list of ASes that a datagram must transit to reach a particular network. Information from the BGP routing table is entered into the main IP routing table (see Figure 6–6). In most cases, the BGP information should override IGP information.

Figure 6–6 BGP and IGP Routing

Networks that originate in the local AS are indicated with the NETWORK router subcommand for the BGP process. Such networks, referred to as local networks, will have a BGP origin attribute of IGP. They appear in the main IP routing table and can have any source; for example, directly connected, static route, learned from an IGP, and so forth. The BGP routing process periodically scans the main IP routing table to detect the presence or absence of local networks, updating the BGP routing table as appropriate.

Backdoor Routes

It is possible to indicate which networks are reachable using a *backdoor* route that the border router should use. A backdoor network is treated as a local network, except that it is not advertised.

Use this variation of the NETWORK router subcommand to specify a backdoor route:

network *address* **backdoor**

The NETWORK router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

The argument *address* is the network that you wish a backdoor route to.

Example

In this example, network 131.108.0.0 is a local network and network 192.31.7.0 is a backdoor network.

```
router bgp 109
network 131.108.0.0
network 192.31.7.0  backdoor
```

Using the REDISTRIBUTE router subcommand, you can inject BGP routing information into the IGP. This creates a situation where BGP is potentially deriving information about local networks from the IGP and then sending such information back into the IGP. This is another reason to suppress nonlocal networks from the IP routing table—you do not want to hear echoes of your routing updates.

It is also possible to inject IP routing table information into the BGP routing table using the REDISTRIBUTE router subcommand. EGP-derived information will have a BGP origin attribute of EGP; all other nonlocal routes will have a BGP origin attribute of incomplete. All IGP and EGP information will now override BGP-derived IP routing table entries. If you are also redistributing information from BGP into an IGP, you must set up appropriate filtering to ensure that routing information does not loop. A configuration such as this is fairly risky, requiring careful attention to filtering. Filtering is described in more detail earlier in this chapter and in subsequent discussions.

BGP Route Selection Rules

The BGP process selects a single AS path to pass along to other BGP-speaking routers. It is important for routing stability that each BGP-speaking router in an AS use the same set of rules so that all BGP-speaking routers arrive at a consistent view of the AS topology. To this end, the BGP implementation has a reasonable set of factory defaults that can be overridden by administrative configuration, as follows:

- An AS path for a network sourced by this BGP-speaking router has the highest preference. The router uses the sourced path with the lowest origin code.
- Administrative weighting is then considered. Larger weight takes precedence.
- Prefer the shorter AS path. All succeeding rules assume equal length paths.
- Prefer external links over internal links.
- Prefer the lowest origin code (IGP <EGP <INCOMPLETE).
- If INTER_AS metric attributes are present, prefer the path with lowest metric.
- Final determinant is the peer with the largest value for the IP address.

BGP Path Attributes

The BGP implementation supports all path attributes defined in RFC 1163 and 1267. This section describes some details of that implementation.

The DEFAULT-METRIC router subcommand can be used to configure the value for the INTER_AS metric attribute. The same metric value will be sent with all BGP updates originating from the router. The default is to not include an INTER_AS metric in BGP updates.

A third-party next hop router address is used in the NEXT_HOP attribute, regardless of the AS of that third-party router.

Transitive, optional, path attributes are passed along to other BGP-speaking routers. The current BGP implementation does not generate such attributes.

Using BGP Without IGP Redistribution

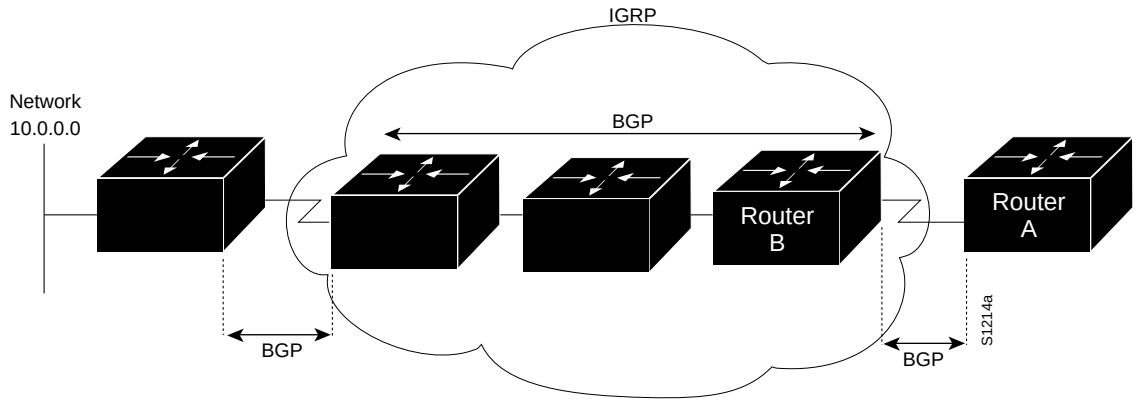
Use the NO SYNCHRONIZATION subcommand of the ROUTER BGP global command when you want to disable the synchronization between BGP and your IGP.

Usually, a BGP speaker does not advertise a route to an external neighbor unless that route is local or exists in the IGP. The NO SYNCHRONIZATION command will allow a router to advertise a network route without waiting for the IGP. This feature allows routers within an AS to have the route before BGP makes it available to other ASes. Use the SYNCHRONIZATION command if there are routers in the AS that do not speak BGP. The default is synchronization.

no synchronization
synchronization

In Figure 6–7, with synchronization on, Router B will not advertise network 10.0.0.0 to Router A until an IGRP route for network 10.0.0.0 exists. If you specify the NO SYNCHRONIZATION command, Router B will advertise network 10.0.0.0 as soon as possible.

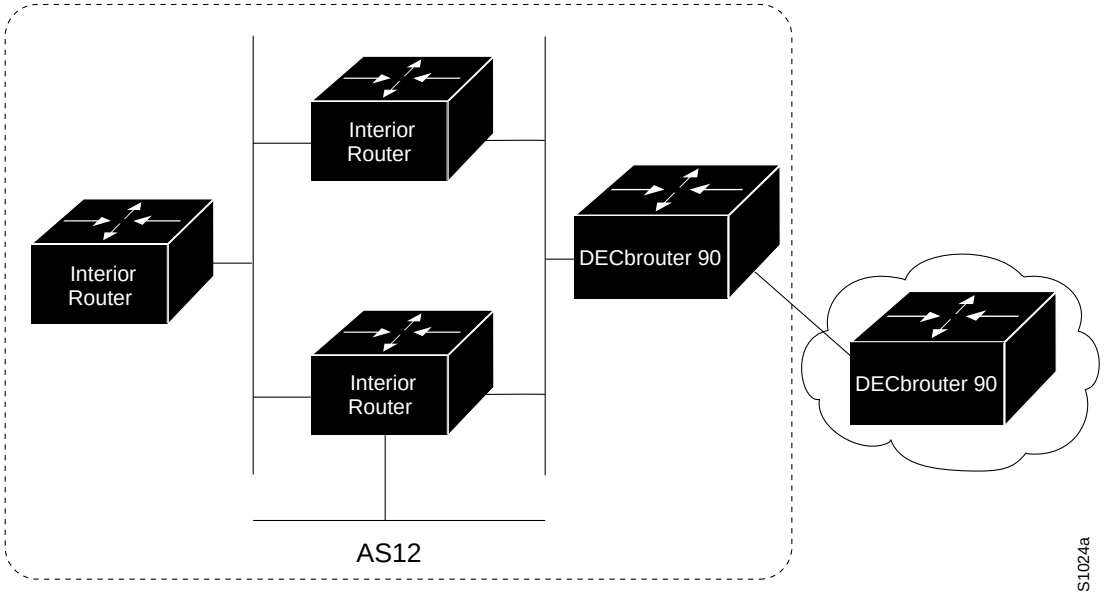
Figure 6–7 Illustration of Synchronization



Configuring the EGP Protocol

The exterior gateway protocol (EGP), specified in RFC 904, is used for communicating with certain routers in the Defense Data Network (DDN) that the U.S. Department of Defense designates as core routers. EGP also is used extensively when attaching to the NSFnet (National Science Foundation Network) and other large backbone networks as shown in Figure 6–8. An exterior router uses EGP to advertise its knowledge of routes to networks within its autonomous system. It sends these advertisements to the core routers, which then readvertise their collected routing information to the exterior router. A neighbor or peer router is any router with which the router communicates using EGP.

Figure 6–8 EGP and Interior and Exterior Routers



Specifying the Autonomous System Number

Before you can set up EGP routing, you must specify an autonomous system number using the AUTONOMOUS-SYSTEM global configuration command. The syntax for this command follows.

autonomous-system *local-AS*
no autonomous-system *local-AS*

The argument *local-AS* is the local autonomous system (AS) number to which the router belongs. The local AS number will be included in EGP messages sent by the router. To remove the AS number, use the NO AUTONOMOUS-SYSTEM global configuration command.

Creating the EGP Routing Process

After the local AS number has been specified, start the EGP routing process with a ROUTER EGP global configuration command:

router egp *remote-AS* **no router egp** *remote-AS*

The argument *remote-AS* is the AS number the router expects its peers to be advertising in their EGP messages. The software does not insist that the actual remote AS number match the configured remote AS numbers. (The output from DEBUG IP-EGP EXEC command will advise of any discrepancies, however. See the section Debugging IP Routing in this chapter for more information.) Turn off your EGP routing process with the NO ROUTER EGP subcommand.

Specifying the List of Neighbors

A router using EGP cannot dynamically determine its neighbor or peer routers. You must provide a list of neighbor routers using the NEIGHBOR router subcommand:

neighbor *ip-address*
no neighbor *ip-address*

The argument *ip-address* is the IP address of a peer router with which routing information will be exchanged. Multiple NEIGHBOR subcommands can be used to specify additional neighbors or peers. The NO neighbor subcommand followed by an IP address removes a peer from the list.

Specifying the Network to Advertise

Use the NETWORK router subcommand to specify the network to be advertised to the EGP peers of an EGP routing process.

network *network-number*
no network *network-number*

The argument *network-number* is the IP address of the network. Such networks are advertised with a distance of zero. There is no restriction on the network number other than that the network must appear in the routing table. The network can be connected, statically configured, or redistributed into EGP from other routing protocols.

Multiple NETWORK subcommands can be used to specify additional networks. The NO NETWORK subcommand followed by the network number removes a network from the list.

The NETWORK router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

Note

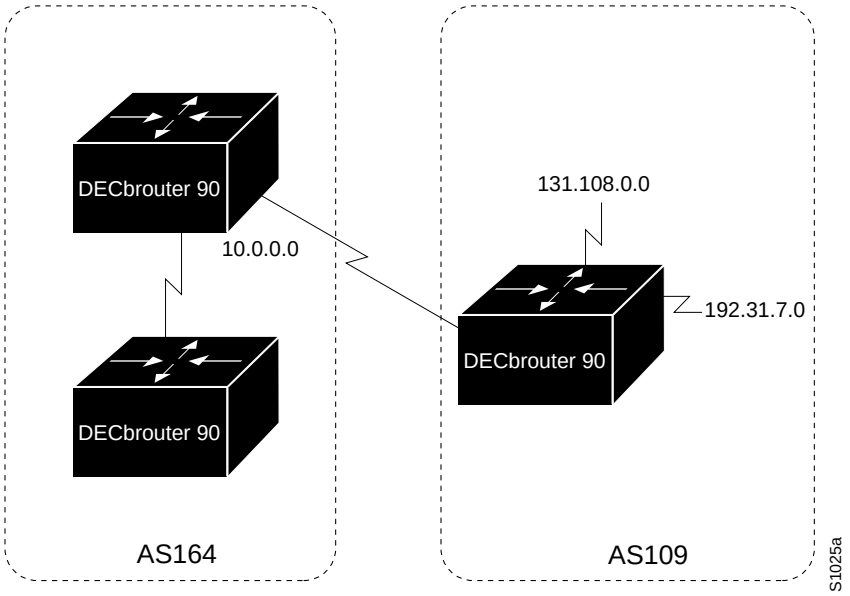
For exterior protocols, a reference to an IP network from the NETWORK command that is learned by another routing protocol does not require a REDISTRIBUTE command. This is in contrast to interior gateway protocols, such as IGRP, which require the use of the REDISTRIBUTE command.

Example

The following is an example configuration for an EGP router process. The router is in autonomous system 109 and is peering with routers in AS 164, as shown in Figure 6–9. It will advertise the networks 131.108.0.0 and 192.31.7.0 to the router in AS 164, 10.2.0.2. The information sent and received from peer routers can be filtered in various ways, including blocking information from certain routers and suppressing the advertisement of specific routes.

```
autonomous-system 109
router egp 164
network 131.108.0.0
network 192.31.7.0
neighbor 10.2.0.2
```

Figure 6–9 Router in AS164 Peers with Router in AS109



Adjusting Timers

The Hello and polltime timers for EGP are adjustable. To adjust the EGP timers, use the following subcommand:

timers egp *hello polltime*
no timers egp

The argument *hello* is the frequency in seconds with which the router sends *Hello* messages to its peer. The default is 60 seconds.

The argument *polltime* is the interval in seconds after not receiving a *Hello* message that the router declares a peer dead. The default is 180 seconds, and the NO TIMERS EGP restores this default.

Example

This command changes the EGP timers to two minutes and five minutes respectively.

```
timers egp 120 300
```

To change the invalid time or flush time for EGP routes, use the TIMERS BASIC command as explained in the section Special Routing Configuration Techniques in this chapter.

Configuring Third-Party EGP Support

EGP supports what is termed a *third-party mechanism*. In this circumstance, EGP tells its peer that another router (the third party) on the shared network is the appropriate router for some set of destinations. If updates mentioning third-party routers are desired, they can be configured using a variation of the NEIGHBOR router subcommand:

neighbor address third-party *third-party-ip-address* [**internal** | **external**]

no neighbor address third-party *third-party-ip-address* [**internal** | **external**]

The argument *third-party-ip-address* is the address of another router (the third party) on the network shared by the DECbrouter 90 and the EGP peer specified by the *address* argument. All networks reachable through that third-party router will be listed in the DECbrouter 90 EGP updates as reachable via that router. Any other networks will be listed as reachable via the DECbrouter 90. The optional keyword **internal** or **external** indicates whether the third-party router should be listed in the internal or external section of the EGP update. Normally, all networks are mentioned in the internal section. You can use the **neighbor** *address* **third-party** router subcommand multiple times to specify additional third-party routers.

Example 1

In the following example, routes learned from router 131.108.6.99 will be advertised to 131.108.6.5 as third-party internal routes.

```
neighbor 131.108.6.5  third-party 131.108.6.99 internal
```

Example 2

In the following example, routes learned from 131.108.6.100 will be advertised to 131.108.6.5 as third-party external routes.

```
neighbor 131.108.6.5  third-party 131.108.6.100 external
```

Configuring a Backup EGP Router

It may be desirable to have a second router belonging to a different AS act as a backup to the EGP router for your AS. To differentiate between the primary and secondary EGP routers, the two routers will advertise network routes with differing EGP distances or metrics. A network with a low metric is generally favored over a network with a high metric.

Networks flagged with the NETWORK router subcommand are always announced with a metric of zero. Networks that are redistributed will be announced with a metric specified by the DEFAULT-METRICS router subcommand. If no metric is specified, redistributed routes will be advertised with a metric of three. All redistributed networks will be advertised with the same metric. The redistributed networks can be learned from static or dynamic routes. See the section Redistributing Routing Information for details about the REDISTRIBUTE router subcommand. A complete configuration example is contained in the section IP Routing Protocol Configuration Examples in this chapter.

Example

The following example configuration illustrates that networks learned by RIP are being advertised with a distance of five. (This is not a complete configuration.)

```
redistribute rip
default-metric 5
```

Generating an EGP Default Route

EGP can now use network 0.0.0.0 as a default route. EGP must be explicitly configured to generate a default route. The router subcommand is as follows:

default-information originate
no default-information originate

If the next hop for the default route can be advertised as a third party, it will be included as a third party.

Defining a Core Gateway EGP Process

In some situations, certain external routing problems can be solved by having a single, central clearinghouse of routing information. The EGP protocol with core gateway support can be used to implement this structure.

Use the ROUTER EGP 0 global configuration command to allow a specific router to have an EGP process that will enable a router to act as a peer with any reachable autonomous system. This defines the router as a *core gateway*. Only one core gateway process can be configured in a router. The command syntax follows.

router egp 0
no router egp 0

The EGP process defined with this command can act as a peer with any autonomous system (AS), and information is interchanged freely between autonomous systems.

Normally, an EGP process expects to communicate with neighbors from a single AS. Because all neighbors are in the same AS, the EGP process assumes that these neighbors all have consistent internal information. Therefore, if the EGP process is informed about a route from one of its neighbors, it will not send it out to other neighbors.

With core EGP, the assumption is that all neighbors are from different ASes, and all have inconsistent information. In this case, the EGP process distributes routes from one neighbor to all others (but not back to the originator). This allows the EGP process to be a central clearinghouse for information.

Note

Split horizon is performed only on a *per-gateway* basis. In other words, if an external router informs a DECbrouter 90 about a specific network, and that router is the *best* path, the DECbrouter 90 will *not* inform the originating external router about that path). The DECbrouter 90 can also perform per-gateway split horizon on third-party updates.

To control how an EGP process determines which neighbors will be treated as peers, use the NEIGHBOR ANY router subcommand with the ROUTER EGP 0 global subcommand. The syntax for this command takes two forms:

neighbor any [*list*]
no neighbor any [*list*]

neighbor any third-party *address* [**internal | external**]
no neighbor any third-party *address* [**internal | external**]

If the *list* argument is specified, the neighbor *must* be accepted by the access-list number specified to be allowed to peer with the EGP process.

The keyword/argument pair **third-party** *address* allows the specified address to be used as the next hop in EGP advertisements.

The optional keywords **internal** or **external** indicate whether the third-party router should be listed in the internal or external section of the EGP update.

Example Core Gateway EGP Configuration

Figure 6–10 illustrates an environment with three routers (designated C1, C2, and C3) attached to a common X.25 network, that are intended to route information using EGP.

With the following configuration (on the router designated Core), C1, C2, and C3 can route traffic directly to each other via the X.25 network:

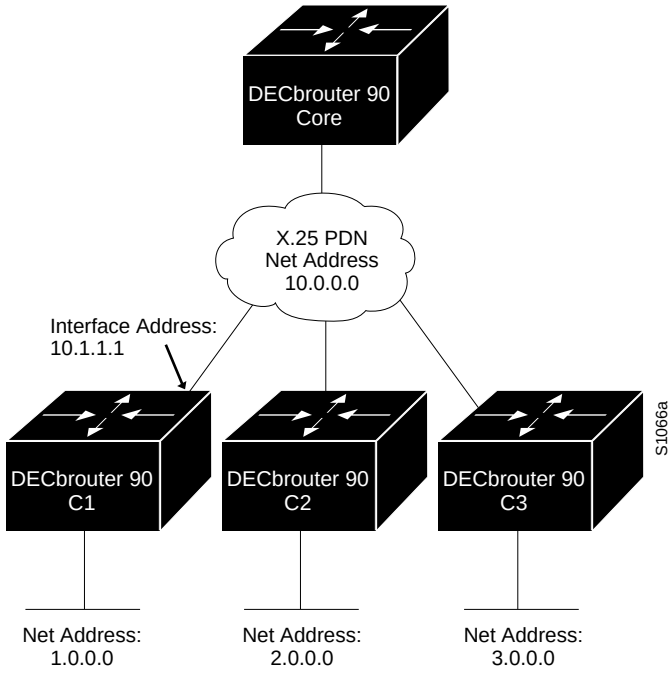
```
access-list 1 permit 10.0.0.0 0.255.255.255
! global access list assignment
router egp 0
neighbor any 1
```

This command specifies that an EGP process on the connected routers (C1, C2, and C3) can act as a peer with any reachable neighbor via the X.25 PDN.

In contrast with this general form, you can specify the particular neighbors that an EGP process will view as peers. The configuration that follows specifies that C1 be advertised directly to C2 and C3, allowing them to bypass Core when routing packets to network 1.0.0.0; however, C2 and C3 route to each other via Core.

```
access-list 2 permit 10.0.0.0 0.255.255.255
! global access list assignment
router egp 0
neighbor any 2
neighbor any third-party 10.1.1.1
```

Figure 6–10 Core EGP Third-Party Update Configuration Example



Configuring IS-IS for TCP/IP

Enabling IP Routing

The first step to configuring your system for the IS-IS routing protocol is to enable IP routing.

Use the following router subcommand to enable IP routing for the router:

ip routing
no ip routing

IP routing is enabled by default.

Example

The following example enables IP routing for the router.

```
ip routing
```

Enabling the IS-IS Routing Protocol

Use the following global configuration command to enable the IS-IS routing protocol on your router and to specify an IS-IS process for IP.

router isis [*tag*]
no router isis [*tag*]

The optional argument *tag* assigns a meaningful name to a routing process; if it is not specified, a null tag is assumed and the process is referenced with a null

tag. The *tag* argument must be unique among all IP router processes for a given router. You can specify only one IS-IS process.

Example

The following example configures the router for IP routing and enables the IS-IS routing protocol.

```
ip routing
router isis
```

Enabling IS-IS for an Interface

Use the following interface subcommand to configure an IS-IS routing process over an interface:

ip router isis *tag*
no ip router isis *tag*

The argument *tag* should be the same routing process name assigned with the ROUTER ISIS command.

Example

The following example specifies IS-IS as an IP routing protocol for a process named Finance, and specifies that the Finance process will be routed on interfaces Ethernet 0 and serial 0.

```
router isis Finance
interface Ethernet 0
ip router isis Finance
interface serial 0
ip router isis Finance
```

Specifying Preferred Routes

Use the following router subcommand to configure the administrative distance for IP routes learned:

distance *value* **ip**
no distance *value* **ip**

The optional argument *value* is the administrative distance, indicating the trustworthiness of a routing information source. This argument has a numerical value between 0 and 255. A higher relative value indicates a lower trustworthiness rating. Preference is given to routes with smaller values. The default value is 110.

Example

In the following example, the distance value for IP routes learned is 90. Preference is given to these IP routes rather than routes with the default administrative distance value of 110.

```
router isis
distance 90 ip
```

Filtering Outgoing Information

This section describes the options you can use to control the redistribution of IP routes into IS-IS.

Advertising Interface Addresses

Use the following router subcommand to instruct IS-IS to advertise the IP addresses for the specified interface without actually running IS-IS on that interface:

passive-interface *interface*
NO passive-interface *interface*

The argument *interface* specifies a particular interface.

The NO PASSIVE-INTERFACE command disables advertising IP addresses for the specified address.

Example

The following configuration enables IS-IS on interfaces serial 1 and serial 0 and advertises the IP addresses of Ethernet 0 in its Link State PDUs.

```
router isis Finance
passive interface Ethernet 0
interface serial 1
ip router isis Finance
interface serial 0
ip router isis Finance
```

Filtering Outbound Updates

Use the following router subcommand to specify the networks to be included in updates:

distribute-list *access-list-number* **out**
no distribute-list *access-list-number* **out**

The argument *access-list-number* is a standard IP access list number as described in the section Configuring IP Access Lists in Chapter 5. The list specifies which network addresses are to be advertised in routing updates. Addresses not specified in the DISTRIBUTE-LIST command will not be advertised in outgoing routing updates.

The keyword **out** applies the access list to outgoing routing updates.

Use the **no distribute-list** with the appropriate access list number and the keyword **out** to disable or change this function.

Example

In the following example, access list 1 is applied to outgoing routing updates and IS-IS is enabled on interface Ethernet 0. Only network 131.131.101.0.0 will be advertised in outgoing IS-IS routing updates.

```
router isis
redistribute ospf 109
distribute-list 1 out
interface Ethernet 0
ip router isis
access-list 1 permit 131.131.101.0 0.0.0.255
```

Exporting IS-IS Routes into Other Protocols

Use the following router subcommand to import IS-IS routes into other protocols.

redistribute isis [*tag*] [**level-1** | **level-1-2** | **level-2**]
no redistribute isis [*tag*] [**level-1** | **level-1-2** | **level-2**]

The optional argument *tag* defines a meaningful name for a routing process. Level-1 and level-2 routes can be redistributed independently. The default setting is **no redistribute isis**.

Example

In the following example, all level-1 routes learned by the IS-IS process ˆfinance˜ will be imported into OSPF 109.

```
router ospf 109
redistribute isis finance level-1
```

Redistributing Static Routes

Use the following router subcommand to redistribute static routes:

redistribute static ip
no redistribute static ip

The default setting is **no redistribute static ip**.

Importing Routes Learned by other IP Routing Protocols

Routes learned by IP routing protocols can be redistributed into IS-IS. Use the following router subcommand to import other routing protocol information into IS-IS:

redistribute *ip-routing-protocol AS-number* **metric** *metric-value*
metric-type [**internal** | **external**] [**level-1** | **level-1-2** | **level-2**]
no redistribute *ip-routing-protocol AS-number* **metric** *metric-value*
metric-type [**internal** | **external**]

The argument *ip-routing-protocol* is one of the following IP routing protocols:

- **igrp**
- **ospf**
- **egp**

- **bgp**
- **rip**
- **hello**

The argument *AS-number* is a unique 16-bit decimal number assigned by the DDN Network Information Center (NIC) to an Autonomous System (AS), which is a collection of networks under a common administration sharing a common routing strategy. For more information about ASes, see the section Autonomous Systems.

The optional keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the route. The *metric-value* argument is a dimensionless link state cost, formed as a 24-bit decimal number. The default value is 0.

Routes learned by IP routing protocols can be redistributed into IS-IS with a configured metric type of **internal** or **external** for different levels. Internal metrics are preferred over external metrics. The default metric type is **internal**.

A router receiving a link state protocol (LSP) with an internal metric will consider the cost of the route from itself to the redistributing router plus the advertised cost to reach the destination. An external metric only considers the advertised metric to reach the destination.

Routes learned from IP routing protocols can be redistributed at **level-1** into an attached area or at **level-2**. The keyword **level-1-2** allows both in a single command. The default setting is **no redistribute**.

Example

In the following example, BGP routes are configured to be redistributed into IS-IS. The link-state cost is specified as 5, and the metric type will be set to external, indicating that it has lower priority than internal metrics.

```
router isis
redistribute bgp 120 metric 5 metric-type external
```

Generating a Default Route

Routers do not by default generate a *default route* into the IS-IS routing domain. Use the following router subcommand to force a router to generate a default route:

default-information originate [**metric** *metric-value*] [**metric-type** *type-value*]
no default-information originate [**metric** *metric-value*] [**metric-type** *type-value*]
[**level-1** | **level-1-2** | **level-2**]

The keyword **originate** causes the router to generate a default external route into an IS-IS domain if the router already has a default route and you want to propagate to other routers.

The optional keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the default route. The *metric-value* argument is a dimensionless link state cost, formed as a 24-bit decimal number. The argument metric-value is a value from 0 to 63. The default value is 0.

The optional keyword/argument pair **metric-type** *type-value* specifies the external link type associated with the default route advertised into the IS-IS routing domain. Valid keywords are **internal** and **external**. If a **metric-type** is not specified, the router adopts an **internal** route.

The optional keywords **level-1**, **level-1-2**, and **level-2** specify if IS-IS advertises network 0.0.0.0 into the level-1 area., the level-2 subdomain, or both. The NO DEFAULT-INFORMATION command disables generation of a default route into the specified IS-IS routing domain. The DEFAULT-INFORMATION subcommand is always used with a REDISTRIBUTE command. If a router configured with this command has a route to 0.0.0.0 in the routing table, IS-IS will originate an advertisement for 0.0.0.0 in its LSPs.

Example

The following configuration forces the router to generate a default external route into an IS-IS domain.

```
router isis
! BGP routes will be distributed into IS-IS
redistribute bgp 120
! access list 2 is applied outgoing routing updates
distribute-list 2 out
! metric of 60 is specified for default router redistributed into IS-IS
! routing domain.
default-information originate metric 60
! access list 2 defined as giving access to network 100.105.0.0
access-list 2 permit 100.105.0.0 0.0.255.255
```

Summarizing Address Ranges

Use the following router subcommand to create aggregate addresses:

summary-address *IP-address IP-mask* [**level-1 | level-2 | level-1-2**]
no summary-address *IP-address IP-mask* [**level-1 | level-2 | level-1-2**]

The argument *IP-address* is a summary address designated for a range of addresses. If level- 2 is specified, routes learned by level-1 routing will be summarized into the level-2 backbone with the configured address/mask value.

If level-1 is specified, only routes redistributed into level-1 are summarized with the configured address/mask value. Multiple groups of addresses can be summarized for a given level. Routes learned from other routing protocols also can be summarized. The metric used to advertise the summary is the smallest metric of all the more specific routes. The default is NO ISIS SUMMARY-ADDRESS. This command helps reduce the size of the routing table.

Example

In the following configuration, summary address 10.1.0.0 includes address 10.1.1, 10.1.2, 10.1.3, 10.1.4, and so forth. Only the address 10.1.0.0 is advertised in an IS-IS level-1 Link State PDU.

```
summary-address 10.1.0.0 255.255.0.0 level-1
```

Configuring Network Entity Titles

Network Entity Titles (NETs) define the area addresses for the IS-IS area, as described in the *DECbrouter 90 Products, Configuration and Reference, Volume 3* publication.

Use the following router subcommand to configure a Network Entity Title (NET) for the routing process:

net *network-entity-title*
no net *network-entity-title*

The argument *network-entity-title* is the NET that specifies the area address and the system ID for an IS-IS routing process. This argument can be either an address or a name. For IS- IS, multiple NETs per router are allowed, with a maximum of three. There is no default value for this command.

Note

Although IS-IS allows you to configure multiple NETs, ISO-IGRP allows only one NET per router.

The NO NET command must be specified with the NET; it removes a specific NET.

Example

The following example illustrates specifying a single NET:

```
router isis Pieinthesky
net 47.0004.004d.0001.0000.0c11.1111.00
```

The following example illustrates the use of a name for a NET:

```
clns host NAME 39.0001.0000.0c00.1111.00
!
router isis
net NAME
!
```

Specifying Router Level Support

Use the following router subcommand to configure the level at which the router will operate.

is-type [**level-1** | **level-1-2** | **level-2-only**]
no is-type [**level-1** | **level-1-2** | **level-2-only**]

If **level-1** is specified, the router acts as a station router. If **level-1-2** is specified, the router acts as both a station router and an area router. If **level-2-only** is specified, the router acts as an area router only. The default value is **level-1-2**.

The command NO IS-TYPE resets the parameter to the default.

Example

The following example specifies an area router:

```
is-type level-2-only
```

Configuring IS-IS Link State Metrics

The interface subcommand ISIS METRIC configures the metric (or cost) for the specified interface. The syntax for this command is as follows:

isis [**metric** *default-metric*] [**level-1** | **level-2**]
no isis metric [**level-1** | **level-2**]

The optional keyword/argument pair **metric** *default-metric* specifies the link state cost to be assigned to the interface. The *default-metric* argument is a dimensionless link state cost, formed as a 24-bit decimal number. The default value for the *default-metric* argument is 10. You can configure this metric for Level 1 and/or Level 2 routing.

The NO ISIS METRIC command resets the *default-metric* value to ten. Specification of the **level-1** or **level-2** optional keywords resets the metric only for Level 1 or Level 2 routing, respectively.

Example

In the following example, interface serial 0 is configured for a default link-state metric cost of 15 for level 1.

```
interface serial 0
isis metric 15 level-1
```

Setting the Advertised hello Interval

Use this interface subcommand to specify the length of time, in seconds, between hello packets that the router sends on the interface.

isis hello-interval *seconds* [**level-1** | **level-2**]
no isis hello-interval *seconds* [**level-1** | **level-2**]

The argument *seconds* is an unsigned integer value. A value three times the hello interval *seconds* is advertised as the *holdtime* in the hello packets transmitted. It must be the same for all routers attached to a common network. With smaller hello intervals, topological changes are detected faster, but there is more routing traffic.

The hello interval can be configured independently for level-1 and level-2, except on serial point-to-point interfaces. (Because there is only a single type of hello packet sent on serial links, it is independent of level-1 or level-2.) The **level-1** and **level-2** keywords are used on X.25, SMDS, and Frame Relay multiaccess networks. The default value is 10 seconds.

Example

In the following example, interface serial 0 is configured to advertise hello packets every five seconds. The router is configured to act as a station router. This will cause more traffic than configuring a longer interval, but topological changes will be detected faster.

```
interface serial 0
isis hello-interval 5 level-1
```

Setting the Advertised CSNP Interval

Complete Sequence Number PDUs (CSNP)s are sent by the designated router to maintain database synchronization. Use the following interface command to configure the IS-IS CSNP interval for the interface:

isis csnp-interval *seconds* [**level-1 | level-2**]
no isis csnp-interval *seconds* [**level-1 | level-2**]

This *seconds* argument is the interval of time between transmission of CSNPs on multiaccess networks. This interval only applies for the designated router. The interval can be configured independently for level-1 and level-2. This command does not apply to serial point-to-point interfaces. This command does apply to WAN connections if the WAN is viewed as a multiaccess meshed network. The default *seconds* is 10 seconds.

Example

In the following example, interface serial 0 is configured for transmitting CSN PDUs every five seconds. The router is configured to act as a station router.

```
interface serial 0
isis csnp-interval 5 level-1
```

Setting the Retransmission Interval

Use the following interface command to configure the number of seconds between retransmission of IS-IS link state PDU (LSP) retransmission for point-to-point links.

isis retransmit-interval *seconds*
no isis retransmit-interval *seconds*

The value for the *number-of-seconds* argument is an integer that should be greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links. The default value is five seconds.

Example

The following example configures interface serial 0 for retransmission of IS-IS LSP every 10 seconds for a large serial line.

```
serial interface 0
isis retransmit-interval 10
```

Specifying Designated Router Election

Use the following interface subcommand to configure the priority of designated routers:

isis priority *value* [**level-1 | level-2**]
NO isis priority [**level-1 | level-2**]

The argument *value* sets the priority of a router and is a number from 0 through 127. The default *value* is 64. Priorities can be configured for Level 1 and Level 2 individually.

The NO ISIS PRIORITY command resets priority to 64. Specification of the **level-1** or **level-2** optional keywords resets priority only for Level 1 or Level 2 routing, respectively.

Example

The following example shows level 1 routing given priority by setting the priority level to 50 (default is 64).

```
interface serial 0
isis priority 50 level-1
```

Specifying Interface Circuit Type

Use the following interface subcommand to configure the type of *adjacency* desired for this interface:

isis circuit-type [**level-1 | level-1-2 | level-2**]
no isis circuit-type

If **level-1** is specified, a Level 1 adjacency may be established if there is at least one area address in common between this system and its neighbors.

If **level-1-2** is specified, a Level 1 and 2 adjacency is established if the neighbor is also configured as **level-1-2** and there is at least one area in common. If there is no area in common, a Level 2 adjacency is established.

If **level-2-only** is specified, a Level 2 adjacency is established if and only if the neighbor is configured exclusively to be a Level 2 router.

The default value for this command is **level-1-2**. The NO ISIS CIRCUIT-TYPE command resets the circuit type to Level l and Level 2.

Example

In the following example, a router is configured to require level 1 adjacency if there is at least one area address in common between this system and its neighbors.

```
clns router isis
interface serial 0
isis circuit type level-1
```

Configuring IS-IS Authentication Passwords

Using the authentication password commands in this section, you can assign passwords to interfaces, areas, and domains.

Assigning a Password for an Interface

Use the following interface subcommand to configure the authentication password for an interface:

isis password *password* [**level-1** | **level-2**]
no isis password [**level-1** | **level-2**]

Different passwords can be assigned for different routing levels using the optional **level-1** and **level-2** keyword arguments. By default authentication is disabled. The NO ISIS PASSWORD command disables authentication for IS-IS. Specifying **level-1** or **level-2** optional keywords disables the password only for Level 1 or Level 2 routing, respectively. If no keyword is specified, the default is **level-1**.

Example

The following example configures a password for interface serial 0 at level 1.

```
interface serial 0
isis password frank level-1
```

Assigning a Password for an Area

Use the following router subcommand to configure the area authentication password. This password is inserted in level-1 (station router level) LSPs, CSNPs, and Partial Sequence Number PDUs (PSNP).

area-password [*password*] **no area-password** [*password*]

The default is **no area-password** [*password*]

Example

The following example assigns an area authentication password.

```
router isis
area-password angel
```

Assigning a Password for a Domain

Use the following router subcommand to configure the routing domain authentication password. This password is inserted in level-2 (the area router level) LSP, CSNP, and PSNP PDUs.

domain-password [*password*]
no domain-password [*password*]

The default is **no domain-password** [*password*].

Example

The following example assigns an authentication password to the routing domain.

```
router isis
domain-password flower
```

Filtering Routing Information

The information sent and received on the networks can be filtered in various ways, including blocking information from certain routers, not sending updates onto a particular subnet, and suppressing the advertisement of specific routes. This section reviews the options for filtering incoming and outgoing information.

Filtering Outgoing Information

This section describes the options you can use to control outgoing information.

Suppressing Updates on an Interface

The PASSIVE-INTERFACE router subcommand disables sending routing updates on an interface.

passive-interface *interface* **no passive-interface** *interface*

The argument *interface* specifies a particular interface. The particular subnet will continue to be advertised to other interfaces. Updates from other routers on that interface continue to be received and processed.

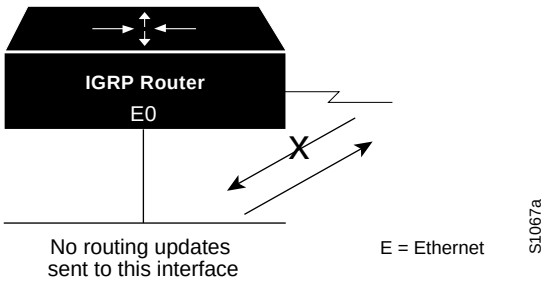
The NO PASSIVE-INTERFACE command re-enables sending routing updates on the specified interface.

Example 1: IGRP Implementation

In the following example, IGRP updates are sent to all interfaces on network 131.108.0.0 except interface Ethernet 0. Figure 6–11 shows this configuration.

```
router igrp 109
network 131.108.0.0
passive-interface ethernet 0
```

Figure 6–11 Filtering IGRP Updates



Example 2: With Neighbor Specification

As with Example 1, in the following example, IGRP updates are sent to all interfaces on network 131.108.0.0 except interface Ethernet 0. However, in this case a NEIGHBOR router subcommand is included. This command permits the sending of routing updates to specific neighbors. One copy of the routing update is generated per neighbor.

```
router igrp 109
network 131.108.0.0
passive-interface ethernet 0
neighbor 131.108.20.4
```

Example 3: OSPF Implementation

In OSPF, the PASSIVE-INTERFACE command disables OSPF Hello protocol neighbor discovery.

This results in the Hello protocol on that interface being disabled, hence the router will not be able to discover any neighbors, and no OSPF neighbor will be able to see the router on that network. In effect, this interface will appear as stub network to OSPF domain. This is useful if you want to import routes associated with a connected network into OSPF domain without any OSPF activity on that interface.

This command typically is used when the wildcard specification on the NETWORK router subcommand configures more interfaces than is desirable. The configuration that follows causes OSPF to run on all subnets of 131.108.0.0.

```
interface Ethernet 0
ip address 131.108.1.1 255.255.255.0
interface Serial 0
ip address 131.108.2.1 255.255.255.0
interface Serial 1
ip address 131.108.3.1 255.255.255.0
router ospf 109
network 131.108.0.0 0.0.255.255 area 0
```

If you do not want OSPF to run on 131.108.3.0 (as an example), then enter the following command:

```
router ospf 109
network 131.108.0.0 0.0.255.255 area 0
passive-interface Serial 1
```

Filtering Outbound Updates

To suppress networks from being sent in updates, use the DISTRIBUTE-LIST router subcommand. Full syntax for this command follows.

distribute-list *access-list-number* **out** [*interface-name* | *routing-process*]
no distribute-list *access-list-number* **out** [*interface-name* | *routing-process*]

The argument *access-list-number* is a standard IP access list number as described in the section Configuring IP Access Lists in Chapter 5. The list explicitly specifies which networks are to be sent and which are to be suppressed.

Use the keyword **out** to apply the access list to outgoing routing updates.

When redistributing networks, a routing process name can be specified as an optional trailing argument to the DISTRIBUTE-LIST subcommand. This causes the access list to be applied to only those routes derived from the specified routing process. After the process-specific access list is applied, any access list specified by a DISTRIBUTE-LIST subcommand without a process name argument will then be applied.

Use the NO DISTRIBUTE-LIST command with the appropriate access list number and keyword to disable or change this function.

Note

To filter networks received in updates, use the DISTRIBUTE-LIST command with the **in** keyword, as explained in the section Filtering Received Updates in this chapter.

Example 1

The following example set of configuration subcommands would cause only one network to be advertised by a RIP routing process: network 131.108.0.0.

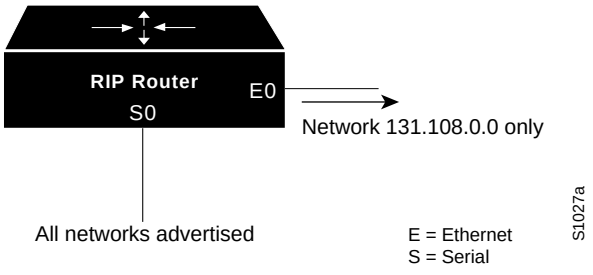
```
access-list 1 permit 131.108.0.0
access-list 1 deny 0.0.0.0 255.255.255.255
router rip
network 131.108.0.0
distribute-list 1 out
```

Example 2

To filter a routing update sent on a specific interface, you can optionally specify the interface. If the last line of the previous example were written as follows, the access list would be applied to updates sent on Ethernet 0. Figure 6–12 illustrates the effects of this command.

```
distribute-list 1 out ethernet 0
```

Figure 6–12 Filtering RIP Updates



Example 3

In the following example, access list 3 is applied to networks derived from process IGRP 109 that are being redistributed by process EGP 164. Networks suppressed by that access list will not be advertised by EGP 164.

```
router egp 164
network 131.108.0.0
redistribute igrp 109
distribute-list 3 out igrp 109
```

Point-to-Point Updates

The NEIGHBOR router subcommand defines a neighboring router with which to exchange routing information, as discussed under the specific protocols:

neighbor *address* **no neighbor** *address*

The argument *address* is the neighboring router address.

For exterior routing protocols such as EGP and BGP, this command specifies routing peers. For normally broadcast protocols such as IGRP or RIP, this subcommand permits the point-to-point (nonbroadcast) exchange of routing information. When used in combination with the PASSIVE-INTERFACE subcommand, routing information can be exchanged between a subset of routers on a LAN. The **no** version removes the neighbor.

Adjusting Metrics

The OFFSET-LIST router subcommand can be used to add a positive offset to incoming and outgoing metrics for networks matching an access list. Full syntax for this command follows.

offset-list *list* {**in** | **out**} *offset*
no offset-list *list* {**in** | **out**}

If the argument *list* is zero, the argument supplied to *offset* is applied to all metrics. If *offset* is zero, no action is taken. For IGRP, the offset is added to the delay component only. This subcommand is implemented for the RIP and Hello routing protocols as well.

The NO OFFSET-LIST command with the appropriate keyword removes the offset list.

Example

In the following example, a router using IGRP applies an offset of ten to its delay component for all outgoing metrics.

```
offset-list 0 out 10
```

In the next example, the router applies the same offset only to access list 121.

```
offset-list 121 out 10
```

Filtering Incoming Information

This section describes the options you can use to control incoming information.

Filtering Received Updates

Use the router subcommand DISTRIBUTE-LIST to filter networks received in updates.

distribute-list *access-list-number* **in** [*interface-name*]
no distribute-list *access-list-number* **in** [*interface-name*]

The argument *access-list-number* is a standard IP access list number as described in the section Configuring IP Access Lists in Chapter 5. The list explicitly specifies which networks are to be received and which are to be suppressed.

Use the keyword **in** to suppress incoming routing updates.

The optional argument *interface-name* specifies the interface (for example, Ethernet 0) on which the access list should be applied to incoming updates. If no interface is specified, the access list will be applied to all incoming updates.

Use the NO DISTRIBUTE-LIST command with the appropriate access list number and keyword to disable or change this function.

Example

The following set of example configuration subcommands would cause only two networks to be accepted by an RIP routing process, network 0.0.0.0 (the RIP default) and network 131.108.0.0.

```
access-list 1 permit 0.0.0.0
access-list 1 permit 131.108.0.0
access-list 1 deny 0.0.0.0 255.255.255.255
router rip
network 131.108.0.0
distribute-list 1 in
```

Filtering Sources of Routing Information

In a large network, some routing protocols and some routers can be more reliable than others as sources of routing information. By specifying administrative distance values, you enable the router to intelligently discriminate between sources of routing information.

An administrative distance is a rating of the trustworthiness of a routing information source, such as an individual router or a group of routers. Numerically, an administrative distance is an integer between 0 and 255. In general, the higher the value, the lower the trust rating. An administrative distance of 255 means the routing information source cannot be trusted at all and should be ignored.

The router always uses the best routing source available: this is the routing source with the lowest administrative distance. For example, consider a router using IGRP and RIP. Suppose you trust the IGRP-derived routing information more than the RIP-derived routing information. If you set the administrative distances accordingly, the router uses the IGRP-derived information and ignores the RIP-derived information. However, if you lose the source of the IGRP-derived information (to a power shutdown in another building, for example), the router uses the RIP-derived information until the IGRP-derived information reappears.

You also can use administrative distance to rate the routing information from routers running the same routing protocol. This application is generally discouraged, however, since it can result in inconsistent routing information including forwarding loops. Example 1 that follows shows how to do this safely.

To define an administrative distance, use the DISTANCE router subcommand.

distance *weight* [*address mask*] [*access-list-number*]
no distance *weight* [*address mask*] [*access-list-number*]

The argument *weight* is an integer from 10 to 255 that specifies the administrative distance. (Values 0 through 9 are reserved for internal use.) Used alone, the argument *weight* specifies a default administrative distance that the router uses when no other specification exists for a routing information source. Weight values are subjective; there is no quantitative method for choosing them.

The optional argument pair *address* and *mask* specifies a particular router or group of routers to which the weight value applies. The argument *address* is an Internet address that specifies a router, network, or subnet. The argument *mask* (in dotted-decimal format) specifies which bits, if any, to ignore in the address value; a set bit in the *mask* argument instructs the router to ignore the corresponding bit in the address value. The optional argument *access-list-number* is the number of a standard IP access list. When used, the access list will be applied to incoming routing updates. Routes which are allowed by the access list will be applied to the routing table at the distance given in the command. Routes which are denied by the access list will be applied to the routing table at the default distance. This allows filtering of networks according to the IP address of the router supplying the routing information. This could be used, as an example, to filter out possibly incorrect routing information from routers not under your administrative control.

To remove an administrative distance value, use the NO DISTANCE subcommand with the appropriate arguments and keywords.

Example 1

In this example, the ROUTER igrp global configuration command sets up IGRP routing in AS number 109. The network subcommands specify routing on networks 192.31.7.0 and 128.88.0.0. The first DISTANCE router subcommand sets the default administrative distance to 255, which instructs the router to ignore all routing updates from routers for which an explicit distance has not been set. The second DISTANCE subcommand sets the administrative distance for all routers on the Class C network 192.31.7.0 to 90. The third DISTANCE subcommand sets the administrative distance for the router with the address 128.88.1.3 to 120.

```
router igrp 109
network 192.31.7.0
network 128.88.0.0
distance 255
distance 90  192.31.7.0  0.0.0.255
distance 120  128.88.1.3  0.0.0.0
```

Example 2

The order in which you enter DISTANCE router subcommands can affect the assigned administrative distances in unexpected ways. For example, the following subcommands assign the router with the address 192.31.7.18 an administrative distance of 100, and all other routers on subnet 192.31.7.0 an administrative distance of 200.

```
distance 100 192.31.7.18  0.0.0.0
distance 200 192.31.7.0  0.0.0.255
```

Example 3

If you reverse the order of these subcommands, all routers on subnet 192.31.7.0 are assigned an administrative distance of 200, including the router at address 192.31.7.18.

```
distance 200 192.31.7.0  0.0.0.255
distance 100 192.31.7.18  0.0.0.0
```

Assigning administrative distances is a problem unique to each network and is done in response to the greatest perceived threats to the connected network. Even when general guidelines exist, the network manager must ultimately determine a reasonable matrix of administrative distances for the network as a whole. Table 6–1 shows the default administrative distance for various sources of routing information.

Table 6–1 Default Administrative Distances

Route Source	Default Distance
Connected interface	0
Static route	1
External BGP	20
IGRP	100
OSPF	110
RIP	120
Hello	130
EGP	140
Internal BGP	200
Unknown	255

Directly Connected Routes

Directly connected routes are routes to the networks specified by the interface addresses of the router. An interface can have multiple IP addresses.

Treatment of Directly Connected Routes

The router automatically enters a directly connected route in the routing table if the interface is usable. A usable interface is one through which the router can send and receive packets. If the router determines that an interface is not usable, it removes the directly connected routing entry from the routing table. Removing the entry allows the router to use dynamic routing protocols to determine backup routes to the network (if any).

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.

Multiple Interface Addresses

The software supports multiple IP addresses per interface. In addition to the primary address specified by the IP ADDRESS interface subcommand, an unlimited number of secondary addresses can be specified by adding the optional keyword **secondary**, as shown:

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

The **no** version of this command removes the specified secondary address association.

Example

In the following example, 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
```

Secondary addresses are treated like primary addresses, except that the system never generates datagrams other than routing updates with secondary source addresses. IP broadcasts and ARP requests are handled properly, as are interface routes in the IP routing table.

Secondary IP addresses can be used in a variety of situations. The following are the most common applications:

- There may not be enough host addresses for a particular network segment. For example, your subnetting allows up to 254 hosts per logical subnet, but on one physical subnet you need to have 300 host addresses. Using secondary IP addresses on the routers allows you to have two logical subnets using one physical subnet.
- Many older networks were built using Level 2 bridges. The judicious use of secondary addresses can aid in the transition to a subnetted, router-based network. Routers on an older, bridged segment can be easily made aware that there are many subnets on that segment.

- Two subnets of a single network might otherwise be separated by another network. This situation is not permitted when subnets are in use. In these instances, the first network is extended, or layered on top of the second network using secondary addresses.

  Note

  If any router on a network segment uses a secondary address, all other routers on that same segment must also use a secondary address from the same network or subnet. An inconsistent use of secondary addresses on a network segment can very quickly lead to routing loops.

Overriding Static Routes with Dynamic Protocols

A static routing entry remains in effect until you remove it. This section describes how a static route can be overridden by dynamic routing information from one of the IP routing protocols. The full syntax of the IP ROUTE global configuration command follows.

ip route *network mask router* [*distance*]

The argument *network* is the Internet address of the target network or subnet. The argument *mask* is a network mask that lets you mask network and subnetwork bits. The argument *router* is the Internet address of a router that can reach that network. The *distance* argument specifies an administrative distance.

If you specify an administrative distance, you are flagging a static route that can be overridden by dynamic information. For example, IGRP-derived routes have a default administrative distance of 100. To have a static route that would be overridden by an IGRP dynamic route, specify an administrative distance greater than 100.

Example

In the following example, an administrative distance of 110 was chosen. In this case, packets for network 10.0.0.0 will be routed via 131.108.3.4 if dynamic information with administrative distance less than 110 is not available.

```
ip route 10.0.0.0 255.0.0.0 131.108.3.4 110
```

Default Routes

A router may not be able to determine the routes to all other networks. To provide complete routing capability, the common practice is to use some routers as "smart routers" and give the remaining routers default routes to the smart router. These default routes can be passed along dynamically or can be configured into the individual routers.

Generating Default Routes

Most dynamic interior routing protocols include a mechanism for causing a *smart router* to generate dynamic default information that is then passed along to other routers.

On your router, use this global configuration command:

ip default-network *network-number*
no ip default-network *network-number*

The argument *network-number* is a network number.

If the router has a directly connected interface onto the specified network, the dynamic routing protocols running on that router will generate or source a default route. In the case of RIP and Hello, this is the mention of the pseudonetwork 0.0.0.0. In the case of IGRP, it is the network itself, flagged as an exterior route.

A router that is generating the default for a network may also need a default of its own. This can be done by specifying a static route to the network 0.0.0.0 via the appropriate router. The **no** version of this command removes the specified default network.

Picking a Default Route

When default information is being passed along through the dynamic routing protocol, no further configuration is required. The system will periodically scan its routing table to choose the optimal default network as its default route. In the case of RIP and Hello, there will be only one choice, network 0.0.0.0. In the case of IGRP, there may be several networks that can be candidates for the system default. The router uses both administrative distance and metric information to determine the default route. The selected default route appears in the gateway of last resort display of the EXEC command SHOW IP ROUTE.

If dynamic default information is not being passed to the router, candidates for the default route can be specified with the IP DEFAULT-NETWORK global command. In this usage, IP DEFAULT-NETWORK takes a nonconnected network as an argument. If this network appears in the routing table from any source (dynamic or static), it is flagged as a candidate default route and is subject to being chosen as the default route for the router. Multiple IP DEFAULT-NETWORK commands can be given. All candidate default routes, both static (that is, flagged by IP DEFAULT-NETWORK) and dynamic, appear in the routing table preceded by an asterisk.

Example

In the following example, a static route to network 10.0.0.0 is defined as the static default route.

```
ip route 10.0.0.0  131.108.3.4
ip default-network 10.0.0.0
```

If the following global configuration command was issued on a router not connected to network 129.140.0.0, then the router might choose the path to that network as a default route when the network appeared in the routing table.

```
ip default-network 129.140.0.0
```

Redistributing Routing Information

In addition to running multiple routing protocols simultaneously, the router can redistribute information from one routing protocol to another. For example, you can instruct the router to readvertise IGRP-derived routes using the RIP protocol, or to readvertise static routes using the IGRP protocol.

The metrics of one routing protocol do not necessarily translate into the metrics of another. For example, the RIP metric is a hop count, the Hello metric is a delay, and the IGRP metric is a combination of five quantities. In such situations, an artificial metric is assigned to the redistributed route. Because of this unavoidable tampering with dynamic information, carelessly exchanging routing information between different routing protocols can create routing loops, which can seriously degrade network operation.

Supported Metric Translations

This section describes the few supported automatic metric translations between the routing protocols. These descriptions assume that you have not defined a default redistribution metric that replaces metric conversions (see the section Setting Default Metrics in this chapter). The following overview describes the router's automatic metric translations.

- RIP can automatically redistribute static routes and Hello-derived routes. RIP assigns static routes a metric of 1 (directly connected) and converts Hello metrics in accordance with Table 6–2. Values are derived from the mapping function defined by Dave Mills and other UNIX gated program developers at the Cornell University Theory Center.
- EGP can automatically redistribute static routes and all dynamically derived routes. EGP assigns the metric three to all static and derived routes.
- BGP does not normally send metrics in its routing updates.
- Hello can automatically redistribute static routes and RIP- and IGRP-derived routes. Hello assigns static routes a metric of 100 (directly connected) and converts RIP metrics in accordance with Table 6–2. Hello advertises IGRP-derived routes with a metric equal to the delay portion of the IGRP metric or 100, whichever is larger. Ethernets have a delay of 1 millisecond and serial links have a delay of 20 milliseconds; Hello assigns a metric of 100 to routes using these media.

Table 6–2 RIP and Hello Metric Transformations

From Hello	To RIP	From RIP	To Hello
0	0	0	0
1–100	1	1	100
101–148	2	2	200
149–219	3	3	300
220–325	4	4	325
326–481	5	5	481
482–713	6	6	713
714–1057	7	7	1057

(continued on next page)

Table 6–2 (Cont.) RIP and Hello Metric Transformations

From Hello	To RIP	From RIP	To Hello
1058–1567	8	8	1567
1568–2322	9	9	2322
2323–3440	10	10	3440
3441–5097	11	11	5097
5098–7552	12	12	7552
7553–11190	13	13	11190
11191–16579	14	14	16579
16580–24564	15	15	24564
24565–30000	16	16	30000

- IGRP can automatically redistribute static routes and information from other IGRP-routed autonomous systems. IGRP assigns static routes a metric that identifies them as directly connected. IGRP does not change the metrics of routes derived from IGRP updates from other autonomous systems.
- Note that any protocol can redistribute other routing protocols if a default metric is in effect (see the section Setting Default Metrics in this chapter).

Passing Routing Information Among Protocols

By default, the router does not exchange information among different routing protocols. If you want to pass routing information among routing protocols, use the REDISTRIBUTE router subcommand. Full syntax for this command follows.

redistribute *process-name* [*AS-number*]
no redistribute *process-name* [*AS-number*]

The argument *process-name* specifies a routing information source using one of the following keywords:

- **static**
- **rip**
- **bgp**
- **egp**
- **hello**
- **ospf**
- **igrp**

When you specify the **bgp** , **igrp**, or **egp** keyword, use the optional argument *AS-number* to specify the autonomous system number.

Use the NO REDISTRIBUTE command with the appropriate arguments to remove this function.

Example

To redistribute RIP-derived information using the Hello protocol, enter these commands:

```
router hello
redistribute rip
```

To end redistribution of information from a routing protocol, use the NO REDISTRIBUTE router subcommand, supplying the appropriate arguments.

Redistributed routing information should always be filtered by the DISTRIBUTE-LIST OUT router subcommand described in the section Filtering Outgoing Information in this chapter. This ensures that only those routes intended by the administrator are passed along to the receiving routing protocol.

When redistributing information between IGRP processes, use the DEFAULT-INFORMATION router subcommand. The full syntax of this command follows.

default-information allowed {**in** | **out**}
no default-information allowed {**in** | **out**}

This subcommand controls the handling of default information between multiple processes.

The NO DEFAULT-INFORMATION ALLOWED IN subcommand causes IGRP exterior or default routes to be suppressed when received by an IGRP process. Normally exterior routes are always accepted. The NO DEFAULT-INFORMATION ALLOWED OUT subcommand causes IGRP exterior routes to be suppressed in updates. Default information is normally passed between IGRP processes when doing redistribution.

The default network of 0.0.0.0 used by RIP and Hello cannot be redistributed by IGRP.

Setting Default Metrics

The DEFAULT-METRIC router subcommand, used in conjunction with the REDISTRIBUTE router subcommand, causes the current routing protocol to use the same metric value for all redistributed routes. (Redistributed routes are those routes established by other routing protocols.) A default metric helps solve the problem of redistributing routes with incompatible metrics; whenever metrics do not convert, using a default metric provides a reasonable substitute and enables the redistribution to proceed.

The DEFAULT-METRIC router subcommand has two forms, depending on the routing protocol specified in the REDISTRIBUTE subcommand.

For RIP, EGP, BGP, and Hello, which use scalar, single-valued metrics, the subcommand has this syntax:

default-metric number

The argument *number* is the default metric value (an unsigned integer) appropriate for the specified routing protocol.

For IGRP, the DEFAULT-METRIC router subcommand has this syntax:

default-metric *bandwidth delay reliability loading mtu*
no default-metric *bandwidth delay reliability loading mtu*

- The argument *bandwidth* is the minimum bandwidth of the route in kilobits per second.
- The argument *delay* is the route delay in tens of microseconds.
- The argument *reliability* is the likelihood of successful packet transmission expressed as a number between 0 and 255 (255 is 100 percent reliability).
- The argument *loading* is the effective bandwidth of the route in kilobits per second.
- The argument *mtu* is the minimum maximum transmission unit (MTU) of the route.

Use the NO DEFAULT METRIC command to return the routing protocol to using the built-in, automatic metric translations.

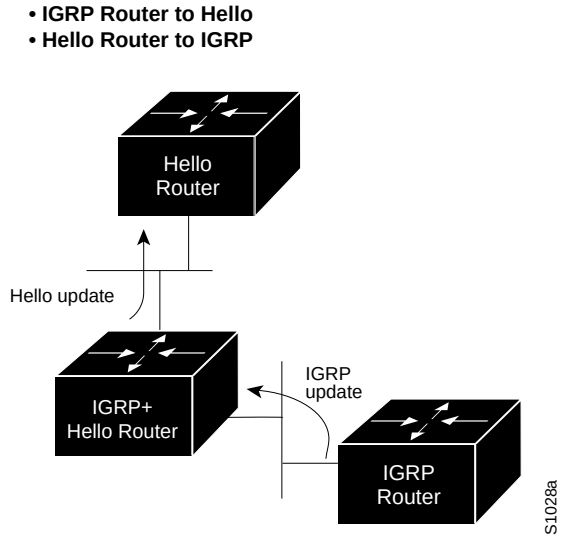
Example

For example, consider a router in autonomous system 109 using both the Hello and IGRP routing protocols. To advertise IGRP-derived routes using the Hello protocol and to assign the IGRP-derived routes a Hello metric of 10,000, the configuration subcommands are as follows:

```
router hello
default-metric 10000
redistribute igrp 109
```

Figure 6–13 shows this type of redistribution.

Figure 6–13 Assigning Metrics for Redistribution



Redistributing Routes into OSPF

Routes from other OSPF routing domains and non-OSPF routing domains can be redistributed into a specific OSPF routing domain. This is accomplished with the REDISTRIBUTE router subcommand. The syntax for this command is as follows:

redistribute *protocol* [*source-id*]
[**metric** *metric-value*]
[**metric-type** *type-value*]
[**tag** *tag-value*]
[**subnets**]

no redistribute *protocol* [*source-id*]
[**metric** *metric-value*]
[**metric-type** *type-value*]
[**tag** *tag-value*]
[**subnets**]

The argument *protocol* is the source protocol from which routes are being redistributed. It can be one of the following keywords:

- **bgp**
- **egp**
- **hello**
- **igrp**
- **ospf**
- **rip**
- **static**

The optional argument *source-id* is either an AS (IGRP) or an appropriate OSPF process id from which routes are to be redistributed. This value takes the form of either a positive integer. If the keywords **hello** or **rip** are used, then no *source-id* value is specified.

The optional keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the redistributed route. The metric-value argument is a dimensionless link state cost, formed as a 24-bit decimal number. If a value is not specified for this option, and no value is specified using the DEFAULT-METRIC router subcommand, the default metric value is 20.

Note

The **metric** value specified in the REDISTRIBUTE router subcommand supersedes the **metric** value specified using the DEFAULT-METRIC router subcommand.

The keyword/argument pair **metric-type** *type-value* specifies the external link type associated with the default route advertised into the OSPF routing domain. The *type-value* argument can assume one of two values.

- 1—Type 1 external route
- 2—Type 2 external route

If a **metric-type** is not specified, the router adopts a Type 2 external route.

The optional keyword/argument pair **tag** *tag-value* specifies a 32-bit decimal value attached to each external route. This is not used by the OSPF protocol itself. It may be used to communicate information between AS boundary routers. If none is specified, then the remote AS number is used for routes from BGP and EGP; for other protocols, zero (0) is used.

The optional keyword **subnets** specifies the scope of redistribution for the specified protocol. If **subnets** is not specified, only major networks are redistributed. If subnets is specified, major networks and subnets are redistributed. For the purposes of this discussion, a major network is any administratively assigned Class A, B, or C IP network.

Note

You can redistribute subnets from any IP routing protocol into OSPF and between different OSPF processes. In addition, if you do not specify the **subnets** keyword, routing information is effectively summarized at redistribution time. For example, assume subnet routes 10.1.0.0 and 10.2.0.0 are learned via IGRP. By omitting the **subnets** keyword, OSPF can be configured to redistribute those routes as a single net (10.0.0.0) route into the OSPF domain.

The NO REDISTRIBUTE command disables redistribution for the protocol or OSPF process specified; it requires that all arguments be specified and disables redistribution only for specific protocols as routing processes. If the keywords **subnet** or **tag** are used, then the effects are isolated to processes associated with these arguments (for example, only subnet redistribution is disabled when the subnet keyword is included, while route redistribution into major nets continues).

Example

The following command example causes the specified IGRP process routes to be redistributed in to an OSPF domain. The IGRP-derived metric will be remapped to 100 and RIP routes to 200.

```
router ospf 109
redistribute igrp 108 metric 100 subnets
redistribute rip metric 200 subnets
```

Generating Default AS Boundary Router Routes for OSPF

Whenever you use the REDISTRIBUTE subcommand to redistribute routes into an OSPF routing domain, the router automatically becomes an AS boundary router. However, an AS boundary router does not by default generate a *default route* into the OSPF routing domain. The DEFAULT-INFORMATION router subcommand allows you to force the AS boundary router do this. The DEFAULT-INFORMATION subcommand is always used with a REDISTRIBUTE command. The syntax of this router subcommand follows.

default-information originate metric *metric-value* **metric-type** *type-value*
no default-information originate metric *metric-value* **metric-type** *type-value*

The keyword **originate** causes the router to generate a default external route into an OSPF domain if the router already has a default route.

The keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the default route. The metric-value argument is a dimensionless link state cost, formed as a 24-bit decimal number.

The keyword/argument pair **metric-type** *type-value* specifies the external link type associated with the default route advertised into the OSPF routing domain. The *type-value* argument can assume one of two values:

- 1—Type 1 external route
- 2—Type 2 external route

If a **metric-type** is not specified, the router adopts a Type 2 external route.

The NO DEFAULT-INFORMATION command disables generation of a default route into the specified OSPF routing domain.

Example

The following example specifies a metric of 100 for the default route redistributed into the OSPF routing domain and an external metric type of Type 1.

```
router ospf 109
redistribute igrp 108 metric 100 subnets
default-information originate metric 100 metric-type 1
```

Redistributing OSPF Routes into Other Domains

This implementation of OSPF includes an extension to the redistribute subcommand that is specific to redistribution of routes from OSPF to other routing domains. The syntax for redistributing OSPF is as follows:

redistribute ospf *ospf-process-id*
[**metric** *metric-value*]
[**match internal | external** *type-value* **external** *type-value*]

no redistribute ospf *ospf-process-id*
[**metric** *metric-value*]
[**match internal | external** *type-value* **external** *type-value*]

This subcommand only applies when redistributing OSPF routes to other routing protocols. You can select any combination of **internal** and/or **external** (Type 1 or Type 2) routes to redistribute. By default, if routes are redistributed into EGP or BGP, only **internal** routes are redistributed. Otherwise all routes are redistributed by default.

The argument *ospf-process-id* is the OSPF process id from which routes are to be redistributed. This value takes the form of a decimal number.

The optional keyword/argument pair **metric** *metric-value* maps OSPF cost assigned to the redistributed route into the destination routing domain metric type. Use a value consistent with the destination protocol.

The optional keyword **match** specifies the criteria by which OSPF routes are redistributed into other routing domains. The keywords used are **internal** and **external**. The keyword **internal** refers to routes that are internal to a specific AS; the keyword **external** refers to routes that are external to the AS, but are to be imported to OSPF as external routes.

The argument *type-value* specifies the external route type to be redistributed into other routing domains. The *type-value* argument can assume one of two values:

- 1—Type 1 external route
- 2—Type 2 external route

There is no default value.

Note

Any match variable is not exclusive; all can be specified. The example that follows illustrates the use of multiple matching criteria.

The NO REDISTRIBUTE command disables redistribution for the OSPF process specified; you must specify the *ospf-process-id* and can only disable individual redistributions.

Example

The following example illustrates the use of this version of the REDISTRIBUTE router subcommand, with the **match** keyword and its options enabled:

```
redistribute ospf 109 match internal external 1 external 2
```

Special Routing Configuration Techniques

This section describes configuration techniques for special situations and requirements.

Configuring Static Routes

The IP ROUTE global configuration command is used to establish static routes. A static route is appropriate if the router cannot dynamically build a route to the destination. The command syntax is as follows:

ip route *network mask* {*address* | *interface*} [*distance*]

The argument *network* is the Internet address of the target network or subnet. The argument *mask* is a network mask that lets you mask network and subnetwork bits. The argument *address* is the Internet address of a router that can reach that network. The argument *interface* is the name of the interface to use for that network. The optional *distance* argument specifies an administrative distance.

If you specify an administrative distance, you are flagging a static route that may be overridden by dynamic information.

Example

In the following example, packets for network 131.108.0.0 will be routed to the router at 131.108.6.6:

```
ip route 131.108.0.0 255.255.0.0 131.108.6.6
```

Enabling and Disabling Split Horizon for IP Networks

Normally, routers that are connected to broadcast-type IP networks and that use distance vector routing protocols employ the *split horizon* mechanism to prevent routing loops. Split horizon blocks information about routes from being advertised by a router out any interface from which that information originated. This behavior usually optimizes communications among multiple routers, particularly when links are broken. However, with nonbroadcast networks, such as frame relay and SMDS, situations can arise for which this behavior is less than ideal.

Use the NO IP SPLIT-HORIZON interface subcommand to disable the split horizon mechanism:

ip split-horizon
no ip split-horizon

For all interfaces except those for which either frame relay or SMDS encapsulation is enabled, the default condition for this command is **ip split-horizon**; in other words, the split horizon feature is active. If the interface configuration includes either the ENCAPSULATION FRAME-RELAY or ENCAPSULATION SMDS commands, the default is for split horizon to be disabled. Split horizon is not disabled by default for interfaces using any of the X.25 encapsulations.

Note

For networks that include links over X.25 PSNs, the neighbor interface subcommand can be used to defeat the split horizon feature. You can as an alternative explicitly specify the no ip split-horizon command in your configuration. However, if you do so you *must* similarly disable split horizon for all routers in any relevant multicast groups on that network.

If split horizon has been disabled on an interface and you wish to enable it, use the IP SPLIT-HORIZON interface subcommand to restore the split horizon mechanism.

Note

In general, changing the state of the default for this interface subcommand is not recommended unless you are certain that your application requires making a change in order to advertise routes properly. Remember: If split horizon is disabled on a serial interface (and that interface is attached to a packet-switched network), you *must* disable split horizon for all routers in any relevant multicast groups on that network.

Example

The following illustrates a simple example of disabling split horizon on a serial link. In this example, the serial link is connected to an X.25 network.

```
interface serial 0
encapsulation x25
no ip split-horizon
```

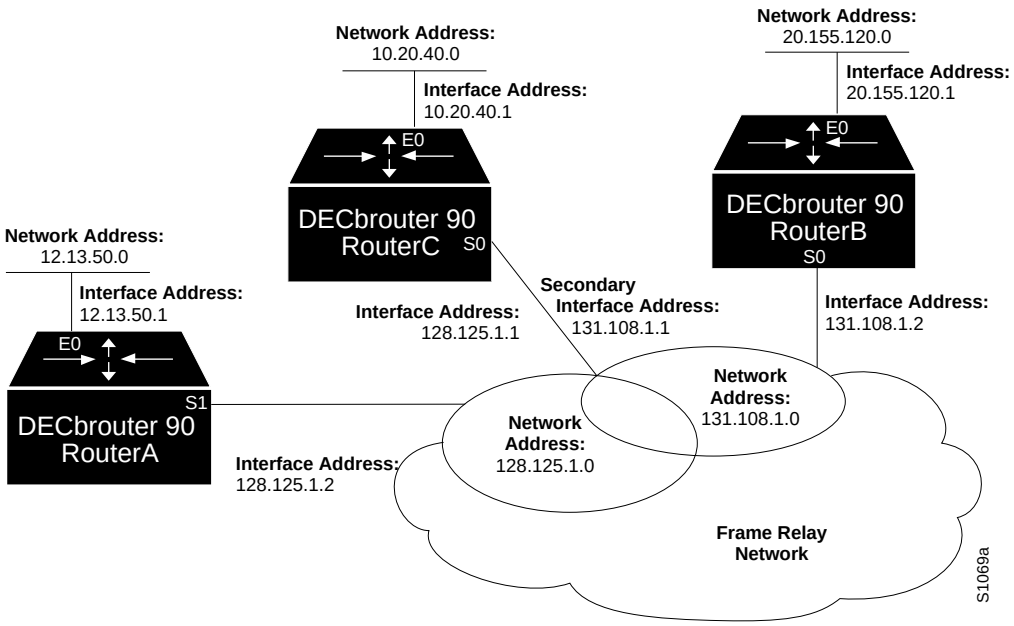
Example of Implicit Split Horizon Conditions

A typical situation in which the NO IP SPLIT-HORIZON command would be useful is illustrated in Figure 6–14. This figure depicts two IP subnets that are both accessible via a serial interface on RouterC (connected to frame relay network). In this example, the serial interface on RouterC accommodates one of the subnets via the assignment of a secondary IP address.

The Ethernet interfaces for RouterA, RouterB, and RouterC (connected to IP networks 12.13.50.0, 20.155.120.0, and 10.20.40.0, respectively) all have split horizon *enabled* by default, while the serial interfaces connected to networks 128.125.1.0 and 131.108.1.0 all have split horizon *disabled* by default. The partial interface configuration specifications for each router that follow Figure 6–14 illustrate that the IP SPLIT-HORIZON interface subcommand is *not* explicitly configured under normal conditions for any of the interfaces.

In this example, split horizon must be disabled in order for network 128.125.1.0 to be advertised into network 131.108.1.0, and vice versa. These subnets overlap at RouterC, interface S0. If split horizon were enabled on serial interface S0, it would not advertise a route back into the frame relay network for either of these subnets.

Figure 6–14 Disabled Split Horizon Example for Frame Relay Network



Example interface configuration for RouterA:

```
interface ethernet 0
ip address 12.13.50.1
!
interface serial 1
ip address 128.125.1.2
encapsulation frame-relay
```

Example interface configuration for RouterB:

```
interface ethernet 0
ip address 20.155.120.1
!
interface serial 0
ip address 131.108.1.2
encapsulation frame-relay
```

Example interface configuration for RouterC:

```
interface ethernet 0
ip address 10.20.40.1
!
interface serial 0
ip address 128.125.1.1
ip address 131.108.1.1 secondary
encapsulation frame-relay
```

IGRP Metric Adjustments

The following METRIC router subcommands alter the default behavior of IGRP routing and metric computation and allow the tuning of the IGRP metric calculation for a particular type of service (TOS):

metric weights TOS K1 K2 K3 K4 K5
no metric weights

The TOS parameter currently must always be zero. Parameters K1 through K5 are constants in the equation that converts an IGRP metric vector into a scalar quantity.

If K5 equals 0, the composite IGRP metric is computed according to the following formula:

metric = [K1*bandwidth + (K2*bandwidth)/(256 - load) + K3*delay]

If K5 does not equal 0, an additional operation is done:

metric = metric * [K5/(reliability + K4)]

The default version of IGRP has K1 == K3 == 1, K2 == K4 == K5 == 0.

Delay is in units of 10 microseconds. This gives a range of 10 microseconds to 168 seconds. A delay of all ones indicates that the network is unreachable.

Bandwidth is inverse bandwidth of the path in bits per second scaled by a factor of 1e10. The range is from a 1200-bps line to 10 Gbps.

Because of the somewhat unusual units used for bandwidth and delay, some examples seem in order. Table 6–3 lists the default values used for several common media.

Table 6–3 Default Bandwidth Values by Media Type

Media Type	Delay	Bandwidth
Satellite	200,000 (2 sec)	20 (500 Mbit)
Ethernet	100 (1 ms)	1,000
1.544 Mbps	2000 (20 ms)	6,476
64 kbps	2000	156,250
56 kbps	2000	178,571
10 kbps	2000	1,000,000
1 kbps	2000	10,000,000

Reliability is given as a fraction of 255. That is, 255 is 100 percent reliability or a perfectly stable link.

Load is given as a fraction of 255. A load of 255 indicates a completely saturated link.

Use the NO METRIC WEIGHTS command to return these five constants to their default values.

The IGRP-only router subcommand METRIC HOLDDOWN can be used to disable holddown. The syntax is as follows:

metric holddown
no metric holddown

The IGRP-only router subcommand METRIC MAXIMUM-HOPS sets a maximum hop count:

metric maximum-hops *hops*
no metric maximum-hops *hops*

This command causes the IP routing software to advertise as unreachable routes with a hop count greater than the value assigned to the *hops* argument. This is a safety mechanism that breaks any potential *count-to-infinity* problems. The default value is 100 hops; the maximum value is 255.

Example

In the following example, a router in AS 71 attached to network 15.0.0.0 wants a maximum hop count of 200, doubling the default. The network administrators decided to do this because they have a complex WAN that can generate a large hop count under normal (nonlooping) operations. (Other commands are needed between the NETWORK command and the METRIC command in a real-world situation; this is not a complete configuration example.)

```
router igrp 71
network 15.0.0.0
metric maximum-hops 200
```

Keepalive Timers

It is possible to tune the IP routing support in the software to enable faster convergence of the various IP routing algorithms and hence, quicker fallback to redundant routers. The total effect is to minimize disruptions to end users of the network in situations where quick recovery is essential.

The network administrator can configure the keepalive interval, the frequency at which the router sends messages to itself (Ethernet and Token Ring) or to the other end (serial), to ensure a network interface is alive. The interval in previous software versions was ten seconds; it is now adjustable in one-second increments down to one second. An interface is declared down after three update intervals have passed without receiving a keepalive packet.

The syntax for the KEEPALIVE interface subcommand is as follows:

keepalive [*seconds*]
no keepalive

If the optional argument *seconds* is not specified, a default of ten seconds is assumed.

Example

In the following example, the keepalive interval is set to three seconds:

```
interface ethernet 0
keepalive 3
```

Setting the keepalive timer to a low value is very useful for rapidly detecting Ethernet interface failures (transceiver cable disconnecting and cable unterminated).

A typical serial line failure involves losing carrier detect (CD). Since this sort of failure is typically noticed within a few milliseconds, adjusting the keepalive timer for quicker routing recovery is generally not useful.

Note

When adjusting the keepalive timer for a very low bandwidth serial interface, large datagrams can delay the smaller keepalive packets long enough to cause the line protocol to go down. You may need to experiment to determine the best value.

Adjustable Routing Timers

The basic timing parameters for IGRP, EGP, RIP, and Hello are adjustable. Since these routing protocols are executing a distributed, asynchronous routing algorithm, it is important that these timers be the same for all routers in the network.

To adjust timers, use the TIMERS BASIC router subcommand. Full syntax for this command follows.

timers basic *update invalid holddown flush sleeptime*
no timers basic

- The argument *update* is the rate (time in seconds between updates) at which updates are sent. This is the fundamental timing parameter of the routing protocol.
- The argument *invalid* is an interval of time (in seconds) after which a route is declared invalid; it should be three times the value of *update*. A route becomes invalid when there is an absence of updates that refresh the route. The route is marked inaccessible and advertised as unreachable. However, the route still is used for forwarding packets.
- The argument *holddown* is the interval (in seconds) during which routing information regarding better paths is suppressed. It should be at least three times the value of update. A route enters into a holddown state when an update packet is received that indicates the route is unreachable. The route is marked inaccessible and advertised as unreachable. However, the route still is used for forwarding packets. When the holddown interval expires, routes advertised by other sources are accepted and the route is no longer inaccessible.
- The argument *flush* is the amount of time (in seconds) that must pass before the route is removed from the routing table; the interval specified for *flush* must be at least the sum of *invalid* and *holddown*. If it is less than this sum, the proper holddown interval cannot elapse, which results in a new route being accepted before the holddown interval expires.
- The argument *sleeptime* is used to postpone routing updates for the specified number of milliseconds. Note that other timing values are specified in seconds. The *sleeptime* value should be less than the *update* time. If the *sleeptime* is greater than the *update* time, routing tables will become unsynchronized.

Use the NO TIMERS BASIC command to reset the defaults.

Note

The current and default timer values can be seen by inspecting the output of the EXEC command SHOW IP PROTOCOLS. The relationships of the various timers should be preserved as described previously.

Example 1: IGRP

In the following example, updates are broadcast every five seconds. If a router is not heard from in 15 seconds, the route is declared unusable. Further information is suppressed for an additional 15 seconds. At the end of the suppression period, the route is flushed from the routing table.

```
router igrp 109
timers basic 5  15  15  30
```

Note that by setting a short update period, you run the risk of congesting slow-speed serial lines; however, this is not a big concern on faster-speed Ethernets and T1-rate serial lines. Also, if you have many routes in your updates, you can cause the routers to spend an excessive amount of time processing updates.

Example 2: EGP

When timers basic is used with EGP, the update time and holddown time are ignored. For example, the commands that follow will set the invalid time for EGP to 100 seconds and the flush time to 200 seconds.

```
router egp 47
timers basic 0 100 0 200
```

Gateway Discovery Protocol (GDP)

The gateway discovery protocol (GDP) allows hosts to dynamically detect the arrival of new routers, as well as determine when a router goes down. Host software is needed to take advantage of this protocol.

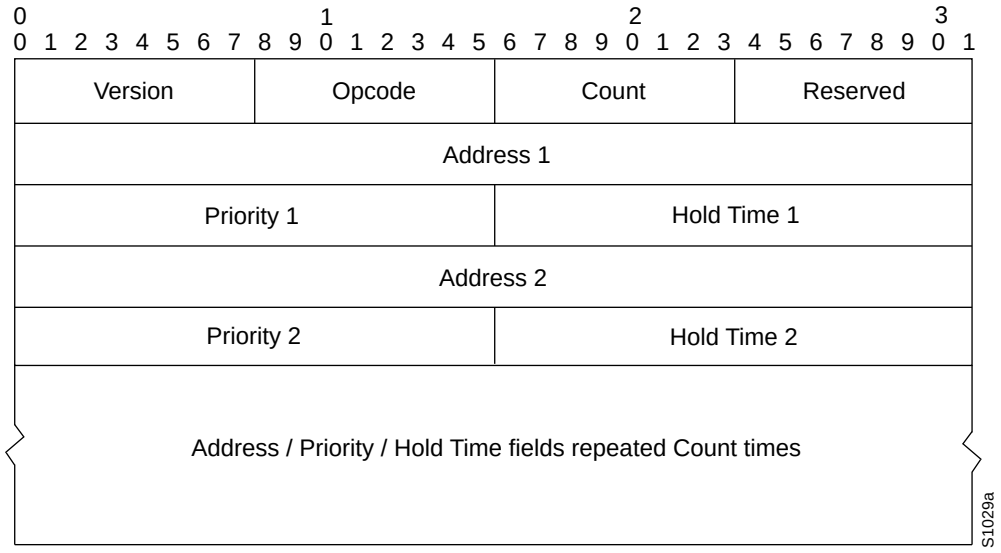
GDP is not a standard; it is a protocol designed by Cisco Systems. The DECbrouter 90 fully supports GDP. Work is in progress to establish GDP or a similar protocol as a standard means of discovering routers. The current GDP implementation is described next in more detail.

For ease of implementation on a variety of host software, GDP is based on the user datagram protocol (UDP). The UDP source and destination ports of GDP datagrams are both set to 1997 (decimal).

There are two types of GDP messages: *Report* and *Query*. On broadcast media, Report message packets are periodically sent to the IP broadcast address announcing that the router is present and functioning. By listening for these Report packets, a host can detect a vanishing or appearing router. If a host issues a Query packet to the broadcast address, the routers each respond with a Report sent to the host's IP address. On nonbroadcast media, routers send Report message packets only in response to Query message packets. The protocol provides a mechanism for limiting the rate at which Query messages are sent on nonbroadcast media.

Figure 6–15 shows the format of the GDP Report message packet format. A GDP Query message packet has a similar format, except that the Count field is always zero and no address information is present.

Figure 6–15 GDP Report Message Packet Format



The fields in the Report and Query messages are as follows:

- **Version**—8-bit field containing the protocol version number. The current GDP version number is 1. If an unrecognized version number is found, the GDP message must be ignored.
- **Opcode**—8-bit field that describes the GDP message type. Unrecognized opcodes must be ignored. Opcode 1 is a Report message and opcode 2 is a Query message.
- **Count**—8-bit field that contains the number of address, priority, and hold time tuples in this message. A Query message has a Count field value of zero. A Report message has a Count field value of 1 or greater.
- **Reserved**—8-bit reserved field; it must be set to zero.
- **Address**—32-bit fields containing the IP address of a router on the local network segment. There are no other restrictions on this address. If a host encounters an address that it believes is not on its local network segment, then the host should quietly ignore that address.
- **Priority**—16-bit fields that indicate the relative quality of the associated address. The numerically larger the value in the priority field, the better the address should be considered.
- **Hold Time**—16-bit fields. On broadcast media, the number of seconds the associated address should be used as a router without hearing further Report messages regarding that address. On nonbroadcast media, such as X.25, this is the number of seconds the requester should wait before sending another Query message.

Numerous actions can be taken by the host software listening to GDP packets. One possibility is to flush the host's ARP cache whenever a router appears or disappears. A more complex possibility is to update a host routing table based on the coming and going of routers. The particular course of action taken depends on the host software and the customer's requirements.

Using GDP Commands

The IP GDP interface subcommand enables GDP processing on an interface. Full syntax for this command follows.

ip gdp
no ip gdp

If you use this form of the IP GDP subcommand, you use only the default parameters.

The default parameters are as follows:

- A reporting interval of five seconds for broadcast media such as Ethernets, and zero seconds (never) for nonbroadcast media such as X.25
- A priority of 100
- A hold time of 15 seconds

If you want to alter some of these parameters, use one of the following interface subcommands:

ip gdp priority *number*
ip gdp reporttime *seconds*
ip gdp holdtime *seconds*

The **priority** keyword takes a number parameter and alters the priority from its default of 100. A larger number signifies a higher priority.

The **reporttime** keyword takes a time parameter in seconds.

The **holdtime** keyword also takes a time parameter in seconds.

When enabled on an interface, GDP updates report the primary and secondary IP addresses of that interface.

Example

In the following example, GDP is enabled on interface Ethernet 0 with a report time of ten seconds, and priority and hold time set to their defaults (because none are specified).

```
interface ethernet 0
ip gdp reporttime 10
```

ICMP Router Discovery Protocol

The ICMP router discover protocol (IRDP) implemented in the DECbrouter 90 products fully conforms to the router discovery protocol outlined in RFC 1256. When operating as a client, router discovery packets are generated, and when operating as a host, router discovery packets are received.

Note

A router can proxy-advertise other machines that use IRDP; however, this is not recommended because it is possible to advertise nonexistent machines or machines that are down.

Using IRDP Commands

The IP IRDP interface subcommand enables IRDP processing on an interface. Full syntax for this command follows:

ip irdp
no ip irdp

The default is for IRDP processing not to be enabled. If you enable IRDP processing, you will use the default parameters. The parameters are as follows:

- The **preference** default value is 100. A lower value increases this router's preference level. The allowed range is from 0 to 255. You can modify a particular router's preference value so that it will only be selected if other routers are down or so that it will be the preferred router to home to.
- The maximum advertised interval **maxadvertinterval** default value is 600 seconds. This value sets the maximum interval between advertisements.
- The minimum advertised interval **minadvertinterval** default value is 400 seconds. If you change **maxadvertinterval**, it defaults to two-thirds of the new value. This value sets the minimum interval between advertisements.
- The **holdtime** value determines how long the advertisements are valid.

Additionally, you can specify an address to proxy-advertise and its preference value.

Use the following interface subcommands to change IRDP parameters:

ip irdp preference *number*
ip irdp maxadvertinterval *seconds*
ip irdp minadvertinterval *seconds*
ip irdp holdtime *seconds*
ip irdp address *address* [*number*]

Example

```
ip irdp                          ! enable irdp
ip irdp preference 50            ! increase router preference from 100 to 50
ip irdp maxadvertinterval 400    ! set maximum time between advertisements
                                 ! to 400 secs
ip irdp minadvertinterval 100    ! set minimum time between advertisements to
                                 ! 100 secs
ip irdp holdtime 6000            ! advertisements are good for 6000 seconds
ip irdp address 131.108.14.5     ! proxy-advertise 131.108.14.5 with default
                                 ! router preference
ip irdp address 131.108.14.6 50 ! proxy-advertise 131.108.14.6 with
                                 ! preference of 50
```

Displaying IRDP Values

To display IRDP values, use the SHOW IP IRDP EXEC command:

```
router> show ip irdp

Ethernet 0 has router discovery enabled

Advertisements will occur between every 450 and 600 seconds.
Advertisements are valid for 1800 seconds.
Default preference will be 100.
 --More--
Serial 0 has router discovery disabled
 --More--
Ethernet 1 has router discovery disabled
```

IP Routing Protocol Configuration Examples

This section contains complete configuration examples of the IP routing protocols.

Static Routing Redistribution

In the example that follows, three static routes are specified, two of which wish to have the IGRP process advertised. Do this by specifying the REDISTRIBUTE STATIC subcommand, then specifying an access list that allows only those two networks to be passed to the IGRP process. Any redistributed static routes should be sourced by a single router to minimize the likelihood of creating a routing loop.

Example

```
ip route 192.1.2.0  192.31.7.65
ip route 193.62.5.0  192.31.7.65
ip route 131.108.0.0  192.31.7.65
access-list 3 permit 192.1.2.0
access-list 3 permit 193.62.5.0
router igrp 109
network 192.31.7.0
redistribute static
distribute-list 3 out static
```

RIP and Hello Redistribution

Consider a wide area network at a university that uses RIP as an interior routing protocol. Assume that the university wants to connect its wide area network to a regional network, 128.1.1.0, which uses Hello as the routing protocol. The goal in this case is to advertise the networks in the university network to the routers on the regional network. The commands for the interconnecting router are listed in the example that follows.

Example

```
router hello
network 128.1.1.0
redistribute rip
default-metric 10000
distribute-list 10 out rip
```

In this example, the ROUTER command starts a Hello routing process. The NETWORK subcommand specifies that network 128.1.1.0 (the regional network) is to receive Hello routing information. The REDISTRIBUTE subcommand specifies that RIP-derived routing information be advertised in the Hello routing updates. The DEFAULT-METRIC subcommand assigns a Hello delay of 10,000 to all RIP-derived routes.

The DISTRIBUTE-LIST router subcommand instructs the router to use access list 10 (not defined in this example) to limit the entries in each outgoing Hello update. The access list prevents unauthorized advertising of university routes to the regional network.

This example could have specified automatic conversion between the RIP and Hello metrics. However, in the interest of routing table stability, it is not desirable to do so. Instead, this example limits the routing information exchanged to availability information only.

IGRP Redistribution

Each IGRP routing process can provide routing information to only one autonomous system; the router must run a separate IGRP process and maintain a separate routing database for each autonomous system it services. However, you can transfer routing information between these routing databases.

Examples

Suppose the router has one IGRP routing process for network 15.0.0.0 in autonomous system 71 and another for network 192.31.7.0 in autonomous system 109, as the following commands specify:

```
router igrp 71
network 15.0.0.0
router igrp 109
network 192.31.7.0
```

To transfer a route to 192.31.7.0 and the database of the first routing process (without passing any other information about autonomous system 109), use the command in the following example:

```
router igrp 71
redistribute igrp 109
distribute-list 3 out igrp 109
access-list 3 permit 192.31.7.0
```

Third-Party EGP Support

In this example configuration, the router is in AS 110 communicating with an EGP neighbor in AS 109 with address 131.108.6.5. Network 131.108.0.0 is advertised as originating within AS 110. The configuration specifies that two routers, 131.108.6.99 and 131.108.6.100, should be advertised as third-party sources of routing information for those networks that are accessible through those routers. The global configuration commands also specify that those networks should be flagged as internal to AS 110.

Example

```
autonomous-system 110
router egp 109
network 131.108.0.0
neighbor 131.108.6.5
neighbor 131.108.6.5  third-party 131.108.6.99 internal
neighbor 131.108.6.5  third-party 131.108.6.100 internal
```

Backup EGP Router

The following example configuration illustrates a router that is in AS 110 communicating with an EGP neighbor in AS 109 with address 131.108.6.5. Network 131.108.0.0 is advertised with a distance of zero, and networks learned by RIP are being advertised with a distance of five. Access list 3 filters which RIP-derived networks are allowed in outgoing EGP updates.

Example

```
autonomous-system 110
router egp 109
network 131.108.0.0
neighbor 131.108.6.5
redistribute rip
default-metric 5
distribute-list 3 out rip
```

BGP Route Advertisement and Redistribution

The following examples illustrate configurations for advertising and redistributing BGP routes. The first example details the configuration for two neighboring routers that run IGRP within their respective ASs and that are configured to advertise their respective BGP routes between each other. The second example illustrates route redistribution of BGP into IGRP and IGRP into BGP.

Example 1: Simple BGP Route Advertisement

This example provides the required configuration for two routers (R1 and R2) that are intended to advertise BGP routes to each other and to redistribute BGP into IGRP.

```
! Router R1 Configuration:
! Assumes AS 1 has network number 131.108.0.0
router bgp 1
network 131.108.0.0
neighbor 131.108.1.1 remote-as 2
!
router igrp 1
network 131.108.0.0
redistribute bgp 1
! Note that IGRP is not redistributed into BGP
!
! Router R2 Configuration:
! Assumes AS 2 has network number 150.136.0.0:
router bgp 2
network 150.136.0.0
neighbor 131.108.1.2 remote-as 1
!
router igrp 2
network 150.136.0.0
redistribute bgp 2
```

Example 2: Mutual Route Redistribution

The most complex redistribution case is one in which *mutual* redistribution is required between an IGP (in this case IGRP) and BGP.

Suppose that EGP is running on a router somewhere else in AS 1, and that the EGP routes are injected into IGRP routing process 1. You must filter to ensure that the proper routes are advertised. The example configuration for router R1 illustrates use of access filters and a distribution list to filter routes advertised to BGP neighbors. This example also illustrates configuration commands for redistribution between BGP and IGRP.

```
! Configuration for router R1:
router bgp 1
network 131.108.0.0
neighbor 131.108.1.1 remote-as 2
neighbor 131.108.2.1 remote-as 1
! 131.108.2.1 an internal peer
neighbor 131.108.3.1 remote-as 3
! 131.108.3.1 is an external peer
redistribute igrp 1
distribute-list 1 out igrp 1
!
! All networks that should be
! advertised from R1 are
! controlled with access lists:
!
access-list 1 permit 131.108.0.0
access-list 1 permit 150.136.0.0
access-list 1 permit 128.125.0.0
!
router igrp 1
network 131.108.0.0
redistribute bgp 1
```

OSPF Routing and Route Redistribution

OSPF typically requires coordination among many internal routers, area border routers, and AS boundary routers. At a minimum, OSPF-based routers can be configured with all default parameter values, with no authentication, and with interfaces assigned to areas.

If you intend to customize your environment, you must ensure coordinated configurations of all routers. Activities that require careful planning include:

- Establishing stub areas
- Enabling and defining OSPF authentication
- Attaching routers to nonbroadcast networks
- Modifying specific OSPF interface options
- Creating virtual links

Three examples follow:

- The first is a simple configuration illustrating basic OSPF commands.
- The second example illustrates a configuration for internal, area border, and AS boundary routers within a single, arbitrarily assigned, OSPF AS.
- The third example illustrates a more complex configuration and the application of various tools available for controlling OSPF-based routing environments.

Example 1: Basic OSPF Configuration

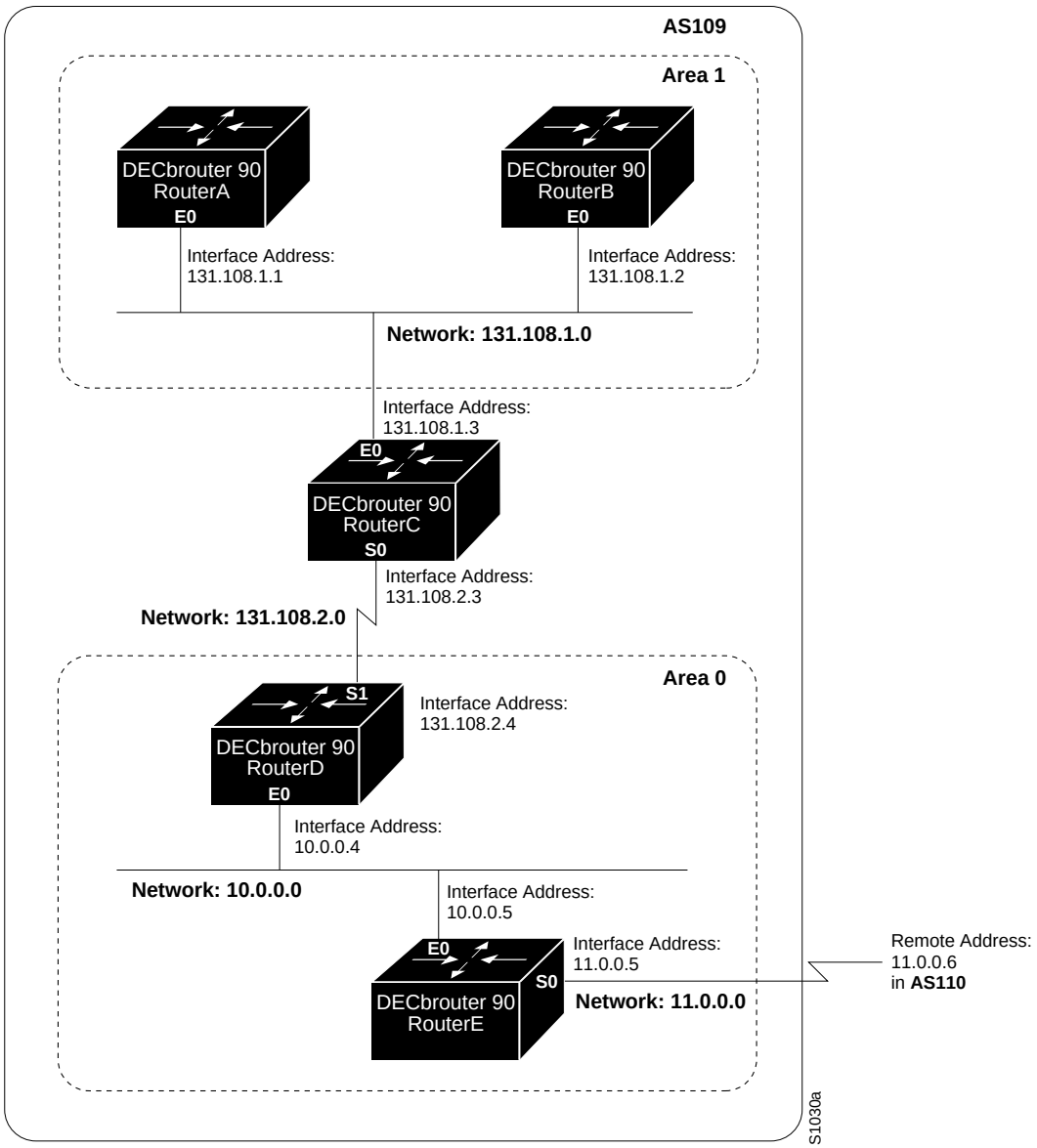
The following example illustrates a simple OSPF configuration that enables OSPF routing process 9000, attaches serial 0 and serial 1 to area 0.0.0.0, and redistributes RIP into OSPF.

```
interface serial0
ip address 130.93.1.1 255.255.255.0
ip ospf cost 1
!
interface serial 1
ip address 130.94.1.1 255.255.255.0
!
router ospf 9000
network 130.93.0.0 0.0.255.255 area 0.0.0.0
redistribute rip metric 1 subnets
!
router rip
network 130.94.0.0
redistribute ospf 9000
default-metric 1
```

Example 2: Internal, Area Border, and AS Boundary Routers

The following example outlines a configuration for several routers within a single OSPF autonomous system. Figure 6–16 provides a general network map that illustrates this example configuration.

Figure 6–16 Sample OSPF Autonomous System Network Map



In this configuration, five routers are configured in OSPF autonomous system AS 109:

- RouterA and RouterB are both internal routers within Area 1.
- RouterC is an OSPF area border router; note that for RouterC, Area 1 is assigned to subnet 1 and Area 0 is assigned to subnet 2.

- RouterD is an internal router in Area 0 (backbone area); in this case, both NETWORK router subcommands specify the same area (Area 0, or the backbone area).
- RouterE is an OSPF AS boundary router; note that BGP routes are redistributed into OSPF and that these route are advertised by OSPF.

Note

It is not necessary to include definitions of all areas in an OSPF AS in the configuration of all routers in the AS. You need only define the *directly* connected areas. In the example that follows, routes in Area 0 are learned by the routers in Area 1 (RouterA and RouterB) when the area border router (RouterC) injects summary link state advertisements (LSAs) into Area 1.

AS 109 is connected to the outside world via the BGP link to the external peer at IP address 11.0.0.6.

```
! RouterA - internal router
interface Ethernet 0
ip address 131.108.1.1 255.255.255.0

router ospf 109
network 131.108.0.0 0.0.255.255 area 1

! RouterB - internal router
interface Ethernet 0
ip address 131.108.1.2 255.255.255.0

router ospf 109
network 131.108.0.0 0.0.255.255 area 1
!
! RouterC - area border router
interface Ethernet 0
ip address 131.108.1.3 255.255.255.0

interface Serial 0
ip address 131.108.2.3 255.255.255.0

router ospf 109
network 131.108.1.0 0.0.0.255 area 1
network 131.108.2.0 0.0.0.255 area 0
!
! RouterD - internal router
interface Ethernet 0
ip address 10.0.0.4 255.0.0.0

interface Serial 1
ip address 131.108.2.4 255.255.255.0

router ospf 109
network 131.108.2.0 0.0.0.255 area 0
network 10.0.0.0 0.255.255.255 area 0
!
! RouterE - AS boundary router
interface Ethernet 0
ip address 10.0.0.5 255.0.0.0

interface Serial 0
ip address 11.0.0.5 255.0.0.0

router ospf 109
network 10.0.0.0 0.255.255.255 area 0
redistribute bgp 109 metric 1 metric-type 1
```

```
router bgp 109
network 131.108.0.0
network 10.0.0.0
neighbor 11.0.0.6 remote-as 110
```

Maintaining IP Routing Operations

The IP routing protocols have one command to help you maintain the IP routing operations.

Use the privileged EXEC command CLEAR IP ROUTE to remove a dynamic route from the routing table. Enter this command at the EXEC prompt:

clear ip route {*network* [*mask*] | *}

The argument *network* is the network address portion of the routing table entry to be removed. You can optionally supply a *mask* to remove a specific subnet mask. If you specify an asterisk (*), all routes are removed. Note that routing entries you remove with the CLEAR IP ROUTE command may reappear because of dynamic routing or because they are floating static routes.

To remove a static route from the routing table, use the NO IP ROUTE global configuration command with the appropriate arguments for the type of static route.

Monitoring IP Routing Operations

This section describes the EXEC commands you can use to monitor IP routing operations configured on your router.

Displaying the BGP Routing Table

Use the SHOW IP BGP EXEC command to display a particular network in the BGP routing table. Enter this command at the EXEC prompt:

show ip bgp [*network*]

The optional argument *network* is a network number and is entered to display a particular network in the BGP routing table.

Following is sample output of the command without specifying a network number:

```
puck> show ip bgp
BGP table version is 22855, local router ID is 192.54.222.5
Status codes: * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network       Next Hop        Metric Weight Path
*> 128.128.0.0   131.192.115.3                0 ?
*> 192.132.70.0  192.54.222.9               20 702 701 ?
*> 152.155.0.0   131.192.77.1                0 ?
*> 192.74.137.0  192.54.222.9               20 702 701 i
*> 192.52.247.0  131.192.2.1                 0 ?
*> 146.150.0.0   131.192.77.1                0 ?
*> 192.139.79.0  192.54.222.9               20 702 701 ?
*> 192.75.142.0  192.54.222.9               20 702 701 ?
*> 192.75.141.0  192.54.222.9               20 702 701 ?
*> 142.136.0.0   192.54.222.9               20 702 701 ?
*> 192.139.77.0  192.54.222.9               20 702 701 ?
*> 129.135.0.0   192.54.222.9               20 702 701 ?
*> 192.160.103.0 192.54.222.9               20 702 701 ?
*> 7.0.0.0       192.54.222.9               20 702 701 35 e
*> 139.140.0.0   131.192.34.2                0 ?
*> 8.0.0.0       131.192.2.1                 0 ?
*> 192.58.242.0  131.192.2.1                 0 ?
```

In the display:

- The `Table Version` is the internal version number for the table. This is incremented any time the table changes.
- The first three characters (`Status codes`) indicate the status. The asterisk (*) indicates that the table entry is valid. The > character indicates that the table entry is the best entry to use for that network. The lowercase i indicates that the table entry was learned via an internal BGP session.
- The `Next Hop` entry is the IP address of the next system to use when forwarding a packet to the destination network. An entry of 0.0.0.0 indicates that the local router has some non-BGP route to this network.
- The `Metric` field, if any, is the value of the interautonomous system metric. This is frequently not used.
- The `Weight` field is set through the use of AS filters.
- The `Path` field is the autonomous system path to the destination network. At the end of the path is the origin code for the path. The lowercase `i` indicates that the entry was originated with the local IGP and advertised with a NETWORK subcommand. A lowercase `e` indicates that the route originated with EGP. A question mark (?) indicates that the origin of the path is not clear. Usually this a path that is redistributed into BGP from an IGP.

Following is sample output of the command when used with a network number:

```
puck> show ip bgp 7.0.0.0
BGP routing table entry for 7.0.0.0, version 20760
Paths: (1 available, best #0)
  702 701 35
    192.54.222.9 from 192.54.222.9, metric 0, weight 20
      Origin EGP, valid, external, best
```

In the display:

- The first line indicates the network that the route is for and the version number of the table the last time the route changed.
- `Paths` indicates the number of paths, and the index to the best path.
- The third line is the AS path associated with the route.
- `192.54.222.9 from 192.54.222.9` indicates the next hop for the route and the router that it is learned from.
- The `metric` is the metric assigned to the route.
- The `weight` is the administrative weight assigned to the route.
- `Origin EGP` is the origin code for the route.
- `Valid` indicates whether or not the route is valid (usable).
- `External` indicates whether or not the route was learned externally or internally.
- `Best` indicates if the route is the best route or not.

Displaying BGP Neighbors

Use the SHOW IP BGP NEIGHBORS EXEC command to display detailed information on the TCP and BGP connections to individual neighbors. Enter this command at the EXEC prompt:

show ip bgp neighbors

Following is sample output:

```
BGP neighbor is 131.108.6.68, remote AS 10, external link
BGP version 3, remote router ID 131.108.6.68
BGP state = Established, table version = 22, up for 0:00:13
Last read 0:00:12, hold time is 180, keepalive interval is 60 seconds
Received 24 messages, 0 notifications
Sent 28 messages, 4 notifications
Connections established 1; dropped 0
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 131.108.6.69, 12288   Foreign host: 131.108.6.68, 179

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

Event Timers (current time is 835828):
Timer:       Retrans   TimeWait    AckHold    SendWnd  KeepAlive
Starts:           20          0         18          0          0
Wakeups:           1          0          2          0          0
Next:              0          0          0          0          0

iss:      60876  snduna:      62649  sndnxt:      62649     sndwnd:   1872
irs:   95187024  rcvnxt:   95188733  rcvwnd:       1969  delrcvwnd:    271

SRTT: 364 ms, RTTO: 1691 ms, RTV: 481 ms, KRTT: 0 ms
minRTT: 4 ms, maxRTT: 340 ms, ACK hold: 300 ms
Flags: higher precedence

Datagrams (max data segment is 1450 bytes):
Rcvd: 36 (out of order: 0), with data: 18, total data bytes: 1708
Sent: 40 (retransmit: 1), with data: 36, total data bytes: 1817
```

In the display:

- The first line lists the IP address of the BGP neighbor and its AS number. If the neighbor is in the same AS as the local router, then the link between them is internal. Otherwise, it is considered external.
- The next line specifies that the BGP version being used to communicate with the remote router is BGP version 3; the neighbor's router id (IP address) is also specified.
- The BGP state indicates the internal state of this BGP connection. The table version indicates that the neighbor has been updated with this version of the primary BGP routing table. The up time indicates the amount of time that the underlying TCP connection has been in existence.
- The last read time is the time that BGP last read a message from this neighbor. The hold time is the maximum amount of time that can elapse between messages from the peer. The keepalive interval is the time period between sending keepalive packets, which help ensure that the TCP connection is up.
- The number of Received messages indicates the number of total BGP messages received from this peer, including keepalives. The number of notifications is the number of error messages received from the peer.
- The number of Sent messages indicates the total number of BGP messages that have been sent to this peer, including keepalives. The number of notifications is the number of error messages that we have sent to this peer.
- The number of Connections established is a count of the number of times that we have established a TCP connection and the two peers have agreed to speak BGP with each other. The number of dropped connections is the number of times that a good connection has failed or been taken down.
- The remainder of the display describes the status of the underlying TCP connection. See Chapter 5 for more information.

Displaying Routes Learned from a Neighbor

Use the SHOW IP BGP NEIGHBORS network routes command to show the routes learned from that particular neighbor. Enter this command at the EXEC prompt:

show ip bgp neighbors *network* **routes**

The optional argument *network* is the network number for the neighbor whose routes you have learned from.

The display is the same as the display for the SHOW IP BGP.

Displaying BGP Paths

Use the SHOW IP BGP PATHS command to display all the BGP paths in the database. Enter this command at the EXEC prompt:

show ip bgp paths

Following is sample output:

```
Address    Hash Refcount Metric Path
0x297A9C      0        2      0 i
0x30BF84      1        0      0 702 701 ?
0x2F7BC8      2      235      0 ?
0x2FA1D8      3        0      0 702 701 i
```

In the display:

- Address—Internal address where the path is stored
- Hash—Hash bucket where path is stored
- Refcount—Number of routes using that path
- Metric—The INTER_AS metric for the path
- Path—The AS_PATH for that route, followed by the origin code for that route

Displaying BGP Summaries

Use the SHOW IP BGP SUMMARY command to display the status of all BGP connections. Enter this command at the EXEC prompt:

show ip bgp summary

Following is sample output:

```
BGP table version is 3937, main routing table version 3937

Neighbor        V    AS MsgRcvd MsgSent TblVer InQ OutQ Uptime/State
192.54.222.6    2   690    7655     268   3937   0    0 2:39:51
192.54.222.9    3   702     682     364   3937   0    0 2:39:54
```

In the display:

- BGP table version—Internal version number of BGP database
- Routing table version—Indicates last version of BGP database that was injected into main routing table
- Neighbor—IP address of a neighbor
- V—Indicates BGP version number spoken to that neighbor
- AS—Indicates the autonomous system number of that neighbor
- MsgRcvd—BGP messages received from that neighbor
- MsgSent—BGP messages sent to that neighbor
- TblVer—Last version of the BGP database that was sent to that neighbor
- InQ—Number of messages from that neighbor waiting to be processed
- OutQ—Number of messages waiting to be sent to that neighbor
- Update/State—The length of time that the BGP session has been in state Established, or the current state if it is not Established

Displaying EGP Statistics

Use the SHOW IP EGP command to display detailed statistics on EGP connections. Enter this command at the EXEC prompt:

show ip egp

The command output includes detailed information about neighbors. Sample output follows. Table 6–4 describes the fields seen.

```
Local autonomous system is 109
  EGP Neighbor FAS/LAS State   SndSeq RcvSeq Hello Poll j/k Flags
  10.3.0.27      1/109 IDLE       625  61323    60  180   0 Perm, Act
* 10.2.0.37      1/109 UP 12:29   250  14992    60  180   3 Perm, Act
* 10.7.0.63      1/109 UP  1d19   876  10188    60  180   4 Perm, Pass
```

Table 6–4 Show IP EGP Field Descriptions

Field	Description
EGP Neighbor	Address of the EGP neighbor
FAS	Foreign autonomous system number
LAS	Local autonomous system number
State	State of the connection between peers
SndSeq	Send sequence number
RcvSeq	Receive sequence number
Hello	Interval between Hello/I-Heard-You packets
Poll	Interval between Poll/Update packets
j/k	Measure of reachability; 4 is perfect
Flags	• Perm—Permanent • Temp—Temporary (neighbor will be removed) • Act—Active, controlling the connection • Pass—Passive, neighbor controls the connection

Displaying Routing Protocol Parameters and Status

Use the EXEC command SHOW IP PROTOCOLS to display the parameters and current state of the active routing protocol process. Enter this command at the EXEC prompt:

show ip protocols

Following is sample output:

```
Routing Protocol is "igrp 109"
  Sending updates every 90 seconds, next due in 88 seconds
  Invalid after 270 seconds, hold down for 280, flushed after 630
  Outgoing update filter list for all routes is not set
  Incoming update filter list for all routes is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  IGRP maximum hopcount 100
  Redis tributing: igrp 109
  Routing for Networks:
    131.108.0.0
    192.31.7.0
  Routing Information Sources:
    Gateway         Distance  Last Update
    131.108.2.201        100      0:00:08
    131.108.200.2        100      5d15
    131.108.2.200        100      0:00:09
    131.108.2.203        100      0:00:11
```

The information displayed by SHOW IP PROTOCOLS is useful in debugging routing operations. Information in the Routing Information Sources field of the SHOW IP PROTOCOLS output can help you identify a router suspected of delivering bad routing information.

In the display:

- The Routing Protocol field specifies the routing protocol used.
- The Sending updates field specifies the time between sending updates, as well as precisely when the next is due to be sent.
- The Invalid field specifies the value of the invalid parameter.
- The hold down field specifies the current value of the hold-down parameter.
- The flushed field specifies the time in seconds after which the individual routing information will be thrown (flushed) out.
- The outgoing update field specifies whether the outgoing filtering list has been set.
- The incoming update field specifies whether the incoming filtering list has been set.
- The Default networks field specifies how these networks will be handled in both incoming and outgoing updates.
- The IGRP metric field specifies the value of the K0-K5 metrics as well as the maximum hopcount.
- The Redistributing field lists the protocol that is being redistributed.
- The Routing field specifies the networks that the routing process is currently injecting routes for.
- The Routing Information Sources field lists all the routing sources the router is using to build its routing table. For each source, you will see displayed:
  - — The IP address
  - — The administrative distance

— The time the last update was received from this source

Displaying OSPF Routing Processes

To display general information about OSPF routing processes in a particular router, use the SHOW IP OSPF EXEC command. The command syntax is as follows:

show ip ospf [*ospf-process-id*]

The optional argument *ospf-process-id* is a specific process id. If this argument is included, only information for the specified routing process is included.

The following is a partial sample **show ip ospf** output when entered without a specific OSPF process id:

```
gwtest> show ip ospf
 Routing Process "ospf 201" with ID 192.4   2.110.200
 Supports only single TOS(TOS0) routes
 It is an area border and autonomous system boundary router
 Summary Link update interval is 0:30:00 and the update due in 0:16:26
 External Link update interval is 0:30:00 and the update due in 0:16:27
 Redistributing External Routes from,
    igrp 200 with metric mapped to 2, includes subnets in redistribution
    rip with metric mapped to 2
    igrp 2 with metric mapped to 100
    igrp 32 with metric mapped to 1
 Number of areas in this router is 3
    Area 192.42.110.0
        Number of interfaces in this area is 1
        Area has simple password authentication
        SPF algorithm executed 6 times
        Area ranges are
           192.42.110.0  255.255.255.0 Active(1)
        Link State Update Interval is 0:30:00 and due in 0:16:25
        Link State Age Interval is 0:20:00 and due in 0:16:25
```

This display provides the following information:

- Sequential list of information for each routing process in the configuration
- Router ID number
- Number of types of service (TOS) supported (Type 0 only)
- Type of OSPF router
- Link Update Interval
- Route redistribution specified in configuration
- Number of areas in the router and the type of areas
- Number of interfaces in the area
- Form of authentication
- Number of times the SPF algorithm has run since the software was initialized
- Area ranges
- Link State Update Interval and Link State Age Interval, and their expected time due

Displaying the OSPF Database

To display lists of information related to the OSPF database for a specific router, use the database option of the SHOW IP OSPF EXEC command. There are several versions of this command, each delivering information about different OSPF link states advertisements. The syntax for the various versions of this command is as follows:

show ip ospf [*ospf-process-id area-id*] **database**
show ip ospf [*ospf-process-id area-id*] **database** [**router**] [*link-state-id*]
show ip ospf [*ospf-process-id area-id*] **database** [**network**] [*link-state-id*]
show ip ospf [*ospf-process-id area-id*] **database** [**summary**] [*link-state-id*]
show ip ospf [*ospf-process-id area-id*] **database** [**asb-summary**] [*link-state-id*]
show ip ospf [*ospf-process-id*] **database** [**external**] [*link-state-id*]

When entered with the optional keywords *router*, *network*, *summary*, and *asb-summary*, different information is displayed. Samples and brief descriptions of each version follow.

Three optional arguments are valid for each of these command variations: *ospf-process-id*, *area-id*, and *link-state-id*.

- *ospf-process-id*—Internally used identification parameter. It is locally assigned and can be any positive integer number. The number used here is the number assigned administratively when enabling the OSPF routing process.
- *area-id*—Area number associated with the OSPF address range defined in the NETWORK command used to define the particular area.
- *link-state-id*—Identifies the portion of the Internet environment that is being described by the advertisement. The value entered depends on the advertisement's LS type. It must be entered in the form of an IP address.

When the link state advertisement is describing a network, the *link-state-id* can be one of two forms:

- The network's IP address (as in type 3 summary link advertisements and in AS external link advertisements)
- A derived address obtained from the link state ID. (Note that masking a network links advertisement's link state ID with the network's subnet mask yields the network's IP address.)

When the link state advertisement is describing a router, the link state ID is always the described router's OSPF router ID.

When an AS external advertisement (LS Type = 5) is describing a default route, its link state ID is set to Default Destination (0.0.0.0).

The following is a sample output from the SHOW IP OSPF DATABASE general display with no optional arguments or keywords:

```
gwtest> show ip ospf database

       OSPF Router with id(190.20.239.66) (Autonomous system 300)

                Displaying Router Link States(Area 0.0.0.0)

  Link ID       ADV Router     Age      Seq#       Checksum  Link count
155.187.21.6   155.187.21.6    1731    0x80002CFB    0x69BC       8
155.187.21.5   155.187.21.5    1112    0x800009D2    0xA2B8       5
155.187.1.2    155.187.1.2     1662    0x80000A98    0x4CB6       9
155.187.1.1    155.187.1.1     1115    0x800009B6    0x5F2C       1
155.187.1.5    155.187.1.5     1691    0x80002BC     0x2A1A       5
155.187.65.6   155.187.65.6    1395    0x80001947    0xEEE1       4
155.187.241.5  155.187.241.5   1161    0x8000007C    0x7C70       1
155.187.27.6   155.187.27.6    1723    0x80000548    0x8641       4
155.187.70.6   155.187.70.6    1485    0x80000B97    0xEB84       6

                Displaying Net Link States(Area 0.0.0.0)

  Link ID       ADV Router      Age        Seq#        Checksum
155.187.1.3  192.20.239.66     1245    0x800000EC      0x82E

                Displaying Summary Net Link States(Area 0.0.0.0)

  Link ID       ADV Router       Age        Seq#        Checksum
155.187.240.0   155.187.241.5    1152      0x80000077      0x7A05
155.187.241.0   155.187.241.5    1152      0x80000070      0xAEB7
155.187.244.0   155.187.241.5    1152      0x80000071      0x95CB
```

Fields in this display provide the following information:

- Link ID
- Advertising router's router ID
- Link state age
- Link state sequence number (detects old or duplicate link state advertisements)
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Link count (number of interfaces detected for router)

The following is a partial sample output from the SHOW IP OSPF DATABASE router display with no optional arguments:

```
gwtest> show ip ospf database router

       OSPF Router with id(190.20.239.66) (Autonomous system 300)
Displaying Router Link States(Area 0.0.0.0)

  LS age: 1176
  Options: (No TOS-capability)
  LS Type: Router Links
  Link State ID: 155.187.21.6
  Advertising Router: 155.187.21.6
  LS Seq Number: 80002CF6
  Checksum: 0x73B7
  Length: 120
  AS Boundary Router
155   Number of Links: 8
```

```
    Link connected to: another Router (point-to-point)
     (link ID) Neighboring Router ID: 155.187.21.5
     (Link Data) Router Interface address: 155.187.21.6
      Number of TOS metrics: 0
       TOS 0 Metrics: 2
```

Fields in this display provide the following information:

- Router ID number
- OSPF autonomous system number (OSPF process id)
- Link state age
- Type of service (TOS) options (Type 0 only)
- Link state type
- Link state ID
- Advertising router's router ID
- Link state sequence number (detects old or duplicate link state advertisements)
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Length (length in bytes of the link state advertisement)
- Definition of router type
- Number of active links
- Link type
- Link ID
- Router interface address
- Type of Service metric (Type 0 only)

The following is a sample output from the SHOW IP OSPF DATABASE NETWORK display with no optional arguments:

```
gwtest> show  ip  ospf  database network

       OSPF Router with id(190.20.239.66) (Autonomous system 300)

                Displaying Net Link States(Area 0.0.0.0)
  LS age: 1367
  Options: (No TOS-capability)
  LS Type: Network Links
  Link State ID: 155.187.1.3 (address of Designated Router)
  Advertising Router: 190.20.239.66
  LS Seq Number: 800000E7
  Checksum: 0x1229
  Length: 52
  Network Mask: 255.255.255.0
        Attached Router: 190.20.239.66
        Attached Router: 155.187.241.5
        Attached Router: 155.187.1.1
        Attached Router: 155.187.54.5
        Attached Router: 155.187.1.5
        Attached Router: 155.187.1.2
        Attached Router: 155.187.21.5
```

Fields in this display provide the following information:

- Router ID number
- OSPF autonomous system number (OSPF process ID)
- Link state age
- Type of Service options (Type 0 only)
- Link state type
- Link state id of designated router
- Advertising router's router ID
- Link state sequence number (detects old or duplicate link state advertisements)
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Length (length in bytes of the link state advertisement)
- Network mask implemented
- List of routers attached to the network (by IP address)

The following is a sample output from the SHOW IP OSPF DATABASE SUMMARY display with no optional arguments:

```
gwtest> show ip ospf database summary

       OSPF Router with id(190.20.239.66) (Autonomous system 300)

                Displaying Summary Net Link States(Area 0.0.0.0)

  LS age: 1401
  Options: (No TOS-capability)
  LS Type: Summary Links(Network)
  Link State ID: 155.187.240.0 (summary Network Number)
  Advertising Router: 155.187.241.5
  LS Seq Number: 80000072
  Checksum: 0x84FF
  Length: 28
  Network Mask: 255.255.255.0   TOS: 0  Metric: 1
```

Fields in this display provide the following information for each network:

- Router ID number
- OSPF autonomous system number (OSPF process ID)
- Link state age
- Type of Service options (Type 0 only)
- Link state type
- Link state ID (summary network number)
- Advertising router's router ID
- Link state sequence number (detects old or duplicate link state advertisements)
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Length (length in bytes of the link state advertisement)

- Network mask implemented, Type of Service, and link state metric

The following is a sample output from the SHOW IP OSPF DATABASE ASB-SUMMARY display with no optional arguments:

```
gwtest> show ip ospf database asb-summary

       OSPF Router with id(190.20.239.66) (Autonomous system 300)

                Displaying Summary ASB Link States(Area 0.0.0.0)

  LS age: 1463
  Options: (No TOS-capability)
  LS Type: Summary Links(AS boundary router)
  Link State ID: 155.187.245.1 (AS Boundary Router address)
  Advertising Router: 155.187.241.5
  LS Seq Number: 80000072
  Checksum: 0x3548
  Length: 28
  Network Mask: 0.0.0.0 TOS: 0  Metric: 1
```

Fields in this display provide the following AS boundary router information:

- Router ID number
- OSPF autonomous system number (OSPF process ID)
- Link state age
- Type of Service options (Type 0 only)
- Link state type
- Link state ID (AS Boundary Router)
- Advertising router's router ID
- Link state sequence number (detects old or duplicate link state advertisements)
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Length (length in bytes of the link state advertisement)
- Network mask implemented, Type of Service, and link state metric

The following is a sample output from the SHOW IP OSPF DATABASE EXTERNAL display with no optional arguments:

```
gwtest> show ip ospf database external

       OSPF Router with id(190.20.239.66) (Autonomous system 300)

                Displaying AS External Link States
```

```
LS age: 280
Options: (No TOS-capability)
LS Type: AS External Link
Link State ID: 143.105.0.0 (external network number)
Advertising Router: 155.187.70.6
LS Seq Number: 80000AFD
Checksum: 0xC3A
Length: 36
Network Mask: 255.255.0.0
      Metric Type: 2 (Larger than any link state path)
      TOS: 0
      Metric: 1
      Forward Address: 0.0.0.0
      External Route Tag: 0
```

Fields in this display provide the following external links information:

- Router ID number
- OSPF autonomous system number (OSPF process ID)
- Link state age
- Type of Service options (Type 0 only)
- Link state type
- Link state ID (External Network Number)
- Advertising router's router ID
- Link state sequence number (detects old or duplicate link state advertisements)
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Length (length in bytes of the link state advertisement)
- Network mask implemented
- External type
- Type of Service
- Link state metric
- Forwarding address (data traffic for the advertised destination will be forwarded to this address. If the forwarding address is set to 0.0.0.0, data traffic will be forwarded instead to the advertisement's originator
- External route tag (a 32-bit field attached to each external route; this is not used by the OSPF protocol itself)

Displaying OSPF Interface Parameters

To display OSPF-related interface information, use the SHOW IP OSPF INTERFACE EXEC command. The command syntax is as follows.

show ip ospf interface [*interface-name*]

The argument *interface-name* can be formed either as the one-word interface description (for example, serial0 or Ethernet0) or can be formed as the two-word *interface-type unit-number* specification (for example, serial 0, Ethernet 0, or e 1).

The following is a sample output from the SHOW IP OSPF INTERFACE display with Ethernet interface 0 specifically stated:

```
gwtest> show ip ospf interface ethernet 0

Ethernet 0 is up, line protocol is up
  Internet Address 131.119.254.202, Mask 255.255.255.0, Area 0.0.0.0
  AS 201, Router ID 192.77.99.1, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State OTHER, Priority 1
  Designated Router id 131.119.254.10, Interface address 131.119.254.10
  Backup Designated router id 131.119.254.28, Interface address 131.119.254.28
  Timer intervals configured, Hello 10, Dead 60, Wait 40, Retransmit 5
    Hello due in 0:00:05
  Neighbor Count is 8, Adjacent neighbor count is 2
     Adjacent with neighbor 131.119.254.28  (Backup Designated Router)
     Adjacent with neighbor 131.119.254.10  (Designated Router)
```

Fields in this display provide the following interface-related information:

- Status of physical link and operational status of protocol
- Interface IP address, subnet mask, and area address
- AS number (OSPF process ID), router ID, network type, link state cost
- Transmit delay, interface state, and router priority
- Designated router ID and respective interface IP address
- Backup designated router ID and respective interface IP address
- Configuration of timer intervals
- Anticipated arrival of next Hello packet
- Count of network neighbors and list of adjacent neighbors

Displaying OSPF Neighbor Information

To display OSPF neighbor information on a per-interface basis, use the SHOW IP OSPF NEIGHBOR EXEC command. The command syntax follows:

show ip ospf neighbor [*interface-name*]

The argument *interface-name* can be formed either as the one-word interface description (for example, serial0 or Ethernet0) or as the two-word *interface-type unit-number* specification (for example, serial 0, Ethernet 0, or e 1).

The following is a partial sample output from the SHOW IP OSPF NEIGHBOR display with no interface argument:

```
gwtest> show ip ospf neighbor serial0

 Neighbor 155.187.241.5, interface address 155.187.1.240
    In the area 0.0.0.0 via interface Ethernet0
    Neighbor priority is 1, State is FULL
    Options 2
    Dead timer due in 0:00:59
 --More--
 Neighbor 155.187.1.1, interface address 155.187.1.1
    In the area 0.0.0.0 via interface Ethernet0
    Neighbor priority is 1, State is FULL
    Options 2
    Dead timer due in 0:00:55
```

Fields in this display provide the following interface-related information:

- Neighbor router ID and interface address
- Area and interface through which OSPF neighbor is known
- Router priority of neighbor, neighbor state
- Hello packet options field contents (E-bit only; possible values are 0 and 2; 2 indicates area is not a stub; 0 indicates area is a stub)
- Expected time before router will declare neighbor dead

Displaying IS-IS Protocol-Specific Information

The SHOW IP PROTOCOLS command lists the protocol-specific information for each ISO-IGRP routing process in this router. Enter this command at the EXEC prompt:

show ip protocols

Following is sample output:

```
Chrysostom>  show ip protocols
Routing Protocol is "isis"
  Sending updates every 0 seconds
  Invalid after 0 seconds, hold down 0, flushed after 0
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: isis
  Address Summarization:
    None
  Routing for Networks:
    Serial0
  Routing Information Sources:
  Distance: (default is 115)
```

- Distance: (default is 100) The first line provides the domain address and area number for Level 1 routing processes. For Level 2 routing processes, this command lists the domain address.
- The next set of fields indicate some of the protocol timers. The field labeled ˆSending updates˜ displays when the next routing updates will be sent.
- The Invalid field indicates how long routing updates are to be believed.
- The Hold Down field indicates how long a route will be held down before new information is to be believed.
- The Sending Router Hellos field indicates how often the routers will send Hello packets to each other and when the next is due.
- The field labeled Invalid indicates how long a neighbor entry will be remembered.
- The ISO-IGRP metric weight displays lists the weights applied to the various components of the metric. These fields are followed by the list of interfaces that are in this area.

In the IS-IS portion of the display:

- The first portion specifies the relevant IS-IS area tag.
- The second line lists the System ID and IS Type.

- The next two information fields display area addresses.
- Area addresses are followed by a list of interfaces on the router supporting IS-IS.
- Several information fields are provided that indicate the next expected IS-IS update and any configuration of route redistribution.

Displaying the IS-IS Database

The SHOW ISIS DATABASE EXEC command displays the IS-IS link state database. If **detail** is specified, the contents of each LSP is displayed. Otherwise, a summary display is provided. The command syntax is as follows:

show isis database [**level-1** | **level-2** | **l1** | **l2** | **detail**]

Note

The notations **l1** and **l2** are abbreviations for the options **level-1** and **level-2**, respectively. Each of the options shown in brackets for this command can be entered in a arbitrary string within the same command entry. For example, the following are both valid command specifications and provided the same display: SHOW ISIS DATA DETAIL 12 and SHOW ISIS DATA 12 DETAIL.

Following is sample output that shows detailed level 2 information in the IS-IS link state database, including IP destinations.

```
gray# show isis database detail 12
IS-IS Level-2 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
1600.8906.4017.00-00* 0x00000010   0xBE1F        1129          1/0/0
  Area Address: 47.0005.80ff.ef00.0000.0001.5940
  NLPID:       0x81 0xCC
 IP Address:  160.89.64.17
  Metric: 10   IS 1600.8906.4017.08
  Metric: 10   IP 160.89.64.0 255.255.255.0
1600.8906.4017.08-00* 0x0000000C   0xCAD1        902           0/0/0
  Metric: 0    IS 1600.8906.4017.00
  Metric: 0    IS VIOLET.00
VIOLET.00-00          0x00000015   0x4437        1083          1/0/0
  Area Address: 39.0001
  NLPID:       0x81 0xCC
  IP Address:  160.89.66.18
  Metric: 10   IS VIOLET.0A
  Metric: 10   IS VIOLET.09
  Metric: 10   IS VIOLET.05
  Metric: 10   IS 1600.8906.4017.08
  Metric: 10   IP 160.89.67.0 255.255.255.0
  Metric: 10   IP 160.89.66.0 255.255.255.0
```

The following fields are provided in this display:

- LSPID —The link state PDU ID. The first six octets form the system ID. A name is displayed in this field if one has been assigned with the clns host command. The next octet is the pseudo ID. When this value is zero, the LSP describes links from the system. When it is nonzero, the LSP is a pseudonode LSP. The designated router for an interface is the only system that originates pseudonode LSPs. The last octet is the LSP number. If there is more data than can fit in a single LSP, additional LSPs are sent with increasing LSP

numbers. An asterisk (*) indicates that the LSP was originated by the local system.

- LSP Seq Num—The sequence number for the LSP that allows other systems to determine whether they have received the latest information from the source.
- LSP Checksum—The checksum of the entire LSP packet.
- LSP Holdtime—The amount of time the LSP remains valid, in seconds.
- ATT—The attach bit. This indicates that the router is also a Level 2 router, and it can reach other areas.
- P—The P bit. Detects whether the IS is area partition repair capable.
- OL—The Overload bit. Determines whether the IS is congested.

A sample output of the SHOW ISIS DATABASE DETAIL command follows.

```
IS-IS Level-1 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
SNUG.00-00          * 0x00000006   0xF6FA        288           0/0/0
  Area Address: 47.0004.004D.0001
  Metric: 10   IS DEMETER.00
  Metric: 0    ES SNUG
 --More--
DEMETER.00-00         0x0000000D   0x9106        330           0/0/0
  Area Address: 47.0004.004D.0001
  Metric: 10   IS DEC.01
  Metric: 10   IS SNUG.00
  Metric: 0    ES DEMETER
 --More--
DEC.00-00             0x00000F98   0xC041        720           1/0/0
  Area Address: 47.0004.004D.0001
  Metric: 10   IS DEC.01
  Metric: 0    ES DEC
  Metric: 0    ES AA00.0400.2D05
 --More--
DEC.01-00             0x00000FB3   0x14BB        334           1/0/0
  Metric: 0    IS DEC.00
  Metric: 0    IS DEMETER.00
  Metric: 0    ES AA00.0400.9105
  Metric: 0    ES 0800.2B14.0528
  Metric: 0    ES AA00.0400.9205
  Metric: 0    ES 0800.2B14.060E

IS-IS Level-2 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
0000.0C00.0C35.03-00  0x00000005   0x04C8        317           0/0/0
  Metric: 0    IS 0000.0C00.0C35.00
 --More--
0000.0C00.3E51.00-00  0x00000009   0xAB98        1182          0/0/0
  Area Address: 39.0004
  Metric: 10   IS 0000.0C00.40AF.00
  Metric: 10   IS 0000.0C00.3E51.05
 --More--
0000.0C00.40AF.00-00* 0x0000000A   0x3AA9        599           0/0/0
  Area Address: 47.0004.004D.0001
```

```
Area Address: 39.0001
Metric: 10   IS 0800.2B16.24EA.01
Metric: 10   IS 0000.0C00.3E51.00
```

In addition to the information displayed in SHOW ISIS DATABASE, this SHOW command displays the contents of each LSP.

Displaying the Routing Table

Use the EXEC command SHOW IP ROUTE to display the current state of the routing table. Enter this command at the EXEC prompt:

show ip route [*address* [*mask*]]

When entered with the optional *address* argument, the command displays detailed routing information for the specified network or subnet. The optional *mask* argument allows you to specify a mask, in addition to the address, which is useful for identifying specific subnet routes. If no *mask* is supplied, the longest matching subnet associated with the address is shown.

Following is sample output from this command when entered without an address:

```
Codes: I - IGRP derived, R - RIP derived, H - Hello derived, O - OSPF derived
       C - connected, S - static, E - EGP derived, B - BGP derived
       * - candidate default route, IA - OSPF inter area route
       E1 - OSPF external type 1 route, E2 - OSPF external type 2 route

Gateway of last resort is 131.119.254.240 to network 129.140.0.0

O E2 150.150.0.0 [160/5] via 131.119.254.6, 0:01:00, Ethernet0
E    192.67.131.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet0
O E2 192.68.132.0 [160/5] via 131.119.254.6, 0:00:59, Ethernet0
O E2 130.130.0.0 [160/5] via 131.119.254.6, 0:00:59, Ethernet0
E    128.128.0.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet0
E    129.129.0.0 [200/129] via 131.119.254.240, 0:02:22, Ethernet0
E    192.65.129.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet0
E    131.131.0.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet0
E    192.75.139.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet0
E    192.16.208.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet0
E    192.84.148.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet0
E    192.31.223.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet0
E    192.44.236.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet0
E    140.141.0.0 [200/129] via 131.119.254.240, 0:02:22, Ethernet0
E    141.140.0.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet0
```

In the displays, the asterisk (*) indicates a candidate default route; NET indicates a network rather than subnetwork entry. Other fields contain this information:

- The first column lists the protocol that derived the route.
- The second column may list certain protocol-specific information as defined in the display header.
- The third column lists the address of the remote network. The first number in the brackets is the administrative distance of the information source; the second number is the metric for the route.
- The fourth column specifies the address of the router that can build a route to the specified remote network.
- The fifth column specifies the last time the route was updated in hours: minutes:seconds.

- The final column specifies the interface through which the specified network can be reached.

Directly connected routers can include subnet information where appropriate.

Following is sample output from this command when entered with both an address and a mask:

```
gwtest> show ip route 131.108.13.111 255.255.255.255

Routing entry for 131.108.13.111 (mask 255.255.255.255)
  Known via "static", distance 1, metric 0
  Tag 0
  Redistributing via ospf 47
  Last update from 131.108.6.7, 8 seconds ago
  Routing Descriptor Blocks:
  * 131.108.6.7
      Route metric is 0, traffic share count is 1
```

This display provides the following information:

- The "Known via" entry describes the protocol from which the route was learned.
- The "Tag 0" entry describes a tag value for the route. This value will be zero if the protocol does not support tags.
- "Routing Descriptor Blocks" provides information about the next hop and which router it was learned from. Each route has up to four descriptor blocks.

Following is sample output from this command when entered with the address 10.0.0.0:

```
Routing entry for 10.0.0.0 (mask 255.0.0.0)
  Known via "igrp 109", distance 100, metric 28476, candidate default path
  Redistributing via igrp 109
  Last update from 192.31.7.130 on Serial1, 29 seconds ago
  Routing Descriptor Blocks:
  * 192.31.7.130, from 192.31.7.130, 29 seconds ago, 15 uses, via Serial2
      Route metric is 28476, traffic share count is 1
      Total delay is 220000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 128/255, Hops 0
```

When you specify that you want information about a specific network displayed, more detailed statistics are shown, as follows:

- The protocol that provided the information
- The administrative distance
- The metric as provided by the protocol (IGRP, in this case)
- The redistribution protocol
- The address of the source of this routing information, along with the following:
  - — Time of the last incoming update for the route
  - — The interface that the information arrived on
- Much of this information is repeated in the Routing Descriptor Block, along with:
  - — Number of times this route has been used since it was added to the table

— Total round-trip delay in seconds

— Minimum bandwidth on the route (the smallest pipe you will encounter along the way to the remote network)

— `Reliability`, which is the likelihood of successful packet transmission expressed as a number between 0 and 255 (255 is 100% reliability)

— Minimum MTU

— `Loading` (which is the effective bandwidth of the route in kilobits per second)

— Hop count

Displaying Network Masks

The EXEC command SHOW IP MASKS displays the masks used for network addresses and the number of subnets using each mask. The full command syntax is as follows:

show ip masks [*address*]

Following is sample output from this command when entered with an address:

```
gwtest> show ip masks 131.108.0.0
Mask            Reference count
255.255.255.255 2
255.255.255.0   3
255.255.0.0     1
```

Debugging IP Routing

The EXEC command DEBUG and the global configuration command LOGGING enable you to record useful routing information. Using the privileged EXEC command DEBUG, you can instruct the router to log any combination of RIP, EGP, Hello, BGP, OSPF, and IGRP routing events as well as routing table events to the console terminal. In general, these commands are entered during troubleshooting sessions with Digital engineers. For each DEBUG command, there is a corresponding UNDEBUG command that disables the command output.

debug ip-bgp

The DEBUG IP-BGP command provides debug messages about BGP events, including inbound updates.

debug ip-bgp-events

The DEBUG IP-BGP-EVENTS command provides debug messages about BGP events, including BGP state machine changes and outbound updates.

debug ip-bgp-updates

The DEBUG IP-BGP-UPDATES command generates per-update messages.

debug ip-egp [*IP-address*]
debug ip-egp-events [*IP-address*]

These EXEC commands allow for the presentation of debugging information relating to a particular neighbor. The argument *IP-address* is optional. If omitted, debugging information for all neighbors are displayed. If both DEBUG IP-EGP and DEBUG IP-EGP-EVENTS are enabled, the mention of individual networks in updates is suppressed. This reduction in the logging output permits easier debugging of EGP update problems.

debug ip-hello

The DEBUG IP-HELLO command enables logging of Hello routing transactions.

debug ip-igrp

The DEBUG IP-IGRP command enables logging of IGRP routing transactions.

debug ip-ospf-adj

The DEBUG IP-OSPF-ADJ command provides information concerning database synchronization and the election of designated routers.

debug ip-ospf-events

The DEBUG IP-OSPF-EVENTS command provides information concerning OSPF related events, such as adjacencies, flooding information, how designated routers are selected, and how SPF is calculated.

debug ip-ospf-flood

The DEBUG IP-OSPF-FLOOD command provides information about link state advertisement flooding.

debug ip-ospf-packet

The DEBUG IP-OSPF-PACKET command displays debugging information on OSPF packet traffic. This display confirms the receipt of each OSPF-related packet (Hello, link state request, and so on).

debug ip-ospf-spf

The DEBUG IP-OSPF-SPF command provides information about how the SPF tree is built and how routes are derived.

debug ip-rip

The DEBUG IP-RIP command enables logging of RIP routing transactions.

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 *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-based transactions.

debug isis-adj-packets

Logs information for adjacency-related functions such as Hello packets sent and received and IS-IS adjacencies going up and down.

debug isis-spf-events

Logs events associated with the Dijkstra algorithm.

debug isis-update-packets

Logs incoming and outgoing sequence number packets and link state packets. It also logs flooding-related functions.

Global Configuration Command Summary

This section provides an alphabetical list of the global commands used with the IP routing protocols.

[**no**] **autonomous-system** *local-AS*

Specifies an autonomous system (AS) number. The argument *local-AS* is the AS number to which the router belongs. To remove the AS number, use the NO AUTONOMOUS-SYSTEM configuration command.

[**no**] **ip as-path access-list** *list* {**permit** | **deny**} *as-regular-expression*

Defines a BGP-related access list. The list argument is a number between 1 and 99. The **permit** and **deny** keywords specify the type of access allowed. The argument *AS-regular-expression* allows use of special characters in specifying AS in access list.

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

Provides a mechanism for causing a smart router to generate dynamic default information that is then passed along to other routers. The argument *network-number* is a network number.

ip route *network* {*address* | *interface*} [*distance*]

Establishes static routes. The *argument* network is the Internet address of the target network or subnet, and the argument *address* is the Internet address of a router that can reach that network. The optional *distance* argument specifies an administrative distance.

[**no**] **router ospf** *ospf-process-id*

Enables OSPF for the router. The argument *ospf-process-id* is an internally used identification parameter. It is locally assigned and can be any positive integer number. A unique value is assigned for each OSPF routing process. You can specify multiple OSPF routing processes in each router.

[**no**] **router** *protocol* [*autonomous-system*]

Creates an IP routing process. The argument *protocol* is one of these protocol-type keywords—**rip**, **egp**, **hello**, **bgp**, or **igrp**. The IGRP, BGP, and EGP protocols use the optional argument *autonomous-system* to supply the number of an autonomous system.

[**no**] **router egp 0**

Allows a specific router to have an EGP process that will enable a router to act as a peer with any reachable autonomous system. This defines the router as a *core gateway*. Only one core gateway process can be configured in a router.

[**no**] **router isis** [*tag*]

Specifies an IS-IS process for IP. The argument tag assigns a meaningful name to a routing process.

Router Subcommand Summary

This section provides an alphabetical list of the router subcommands used with the IP routing protocols.

[**no**] **area** *area-id* **authentication**

Enables authentication for an area. The argument *area-id* is the specific area id of the area for which authentication is to be enabled. The authentication *type* (AuType 0 or AuType 1) must be the same for all routers in an area. The authentication *key* (password) for all OSPF routers on a network must be the same if they are to communicate with each other via OSPF. Use the IP OSPF AUTHENTICATION-KEY interface subcommand to specify this password.

[**no**] **area** *area-id* **range** *address mask*

Consolidates or summarizes routes. The result is that a single summary route is advertised to other areas by the area border router. This command is only used with area border routers.

The argument *area-id* is the specific area id for the area about which routes are to be summarized. The argument *area-id* can be specified as either a decimal value or as an IP address.

The *address* argument is a standard IP address.

The *mask* argument is a standard IP mask.

[**no**] **area** *area-id* **stub**
[**no**] **area** *area-id* **default-cost** *cost*

Defines an area as a stub area. These commands are used only on an area border router attached to a stub.

The argument *area-id* is the specific area id for the stub area. The argument *area-id* can be specified as either a decimal value or as an IP address.

The **stub** option is used to enable the stub area.

The **default-cost** *cost* keyword/argument pair assigns a specific cost for the default external route used for the stub area. The acceptable value is a 24-bit number.

[**no**] **area** *area-id* **virtual-link** *router-id* [**hello-interval** *number-of-seconds*]
[**retransmit-interval** *number-of-seconds*]
[**transmit-delay** *number-of-seconds*]
[**dead-interval** *number-of-seconds*]
[**authentication-key** *8-bytes-of-password*]

Defines virtual links.

The argument *area-id* is the area id assigned to the transit area for the virtual link.

The argument *router-id* is the router id associated with the virtual link neighbor.

There are no default value for the *area-id* or *router-id* arguments for this command.

Specify the length of time, in seconds, between the Hello packets that the router sends on the interface with the **hello-interval** option of the AREA *area-id* VIRTUAL-LINK router subcommand. The argument *number of-seconds* is an unsigned integer value. This value is advertised in the router's Hello packets. It must be the same for all routers attached to a common network. The smaller the Hello interval, the faster topological changes will be detected, but more routing traffic will ensue. The default is ten seconds.

The number of seconds between link state advertisement retransmissions for adjacencies belonging to the interface is specified with the **retransmit-interval** option of the AREA *area-id* VIRTUAL-LINK router subcommand. The value for the *number-of-seconds* argument is a integer number, that should be greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links. The default value five seconds.

The estimated number of seconds it takes to transmit a link state update packet on the interface is specified with the **transmit-delay** option of the AREA *area-id* VIRTUAL-LINK router subcommand. Link state advertisements in the update packet must have their age incremented by this amount before transmission. The value assigned should take into account the transmission and propagation delays for the interface. The argument *number-of-seconds* is a integer value that must be greater than zero. The default value is one second.

Set the number of seconds that a router's Hello packets have not been seen before its neighbors declare the router down with the **dead-interval** option of the AREA *area-id* VIRTUAL-LINK router subcommand. The argument *number-of-seconds* is an unsigned integer value. The default is four times the HelloInterval. As with the HelloInterval, this value must be the same for all routers attached to a common network.

Assign a specific password to be used by neighboring routers with the **authentication-key** option of the AREA *area-id* VIRTUAL-LINK router subcommand. The argument *8-bytes-of-password* is any continuous string of characters that you can enter from the keyboard up to eight bytes in length. This configured data allows the authentication procedure to generate and/or verify the authentication field in the OSPF header. There is no default value.

[**no**] **area-password** [*password*]

Configures the IS-IS authentication password for an area.

[**no**] **default-information originate** [**metric** *metric-value*] [**metric-type** *type-value*] [**level-1 | level-1-2 | level-2**]

Forces the IS=IS AS boundary router to generate a default route into the routing domain. The keyword **originate** causes the router to generate a default external route into a domain if the router already has a default route. The DEFAULT-INFORMATION subcommand is always used with a REDISTRIBUTE command.

The keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the default route. The *metric-value* argument is a dimensionless link state cost, formed as a 24-bit decimal number.

The keyword/argument pair **metric-type** *type-value* specifies the external link type associated with the default route advertised into the OSPF routing domain. The *type-value* argument can assume one of two values: **internal** or **external**.

The optional keywords **level-1**, **level-1-2**, and **level-2** specify if IS-IS advertises network 0.0.0.0 into the level-1 area, the level-2 subdomain, or both.

[**no**] **distribute-list** *access-list-number* **out**

Specifies networks to be included in updates. The argument *access-list-number* is a standard IP access list number as described in the section Configuring IP Access Lists in Chapter 5. The list explicitly specifies which networks are to be sent and which are to be suppressed. The keyword out applies the access list to outgoing routing updates.

[**no**] **distance** *value* **ip**

Configures the IS-IS administrative distance for IP routes learned. The argument value is the administrative distance, which indicates the trustworthiness of a routing information source.

[**no**] **domain-password** [*password*]

Configures the IS-IS routing domain authentication password.

[**no**] **is-type** [**level-1 | level-1-2 | level-2-only**]

Configures the IS-IS level at which the router should operate for IS-IS. If **level-1** is specified, the router acts as an station router. If **level-1-2** is specified, the router acts as both a station router and a router that routes between areas. If **level-2-only** is specified, the router acts as an interarea router only. The default value is **level-1-2**. The NO IS-TYPE command resets routing level to Level 1 and 2.

[**no**] **net** *network-entity-title*

Configures a Network Entity Title (NET) for the IS-IS routing process. The argument *network-entity-title* configures the domain, area, and system-id for

an ISO-IGRP routing process or the area address and system-id for an IS-IS routing process.

[**no**] **redistribute** *ip-routing-protocol AS-number* **metric** *metric-value* **metric-type** [**internal** | **external**]

Imports routes learned by other IP routing protocols in IS-IS. The argument *ip-routing-protocol AS-number* is one of the following IP routing protocols:

- igrp
- ospf
- egp
- bgp
- rip
- igp

The argument *AS-number* is a unique 16-bit decimal number assigned by the DDN Network Information Center (NIC) to an Autonomous System (AS), which is a collection of networks under a common administration sharing a common routing strategy.

The *metric-value* is a dimensionless link state cost formed as a 24-bit decimal number The optional keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the default route. The *metric-value* argument is a dimensionless link state cost, formed as a 24-bit decimal number. The default value is 0.

Routes learned by OSI routing protocols can be redistributed into IS-IS with a configured metric type of **internal** or **external** for different levels. Internal metrics are preferred to external metrics. The default metric type is **internal**.

[**no**] **redistribute isis** [*tag*] [**level-1** | **level-1-2** | **level-2**]

Imports IS-IS routes into other protocols. The optional argument tag defines a meaningful name for a routing process. Level-1 and level-2 routes can be redistributed independently. The default setting is NO REDISTRIBUTE ISIS.

[**no**] **redistribute static ip**

Causes the specified process to advertise IS-IS routes that have been manually entered into the routing table. Default is NO REDISTRIBUTE STATIC IP.

[**no**] **summary-address** *IP-address IP-mask* [**level-1** | **level-2** | **level-1-2**]

Creates IS-IS aggregate addresses. The argument *IP-address* is the summary address, and the argument *IP-mask* is the related subnet mask. If level-2 is specified, routes learn via level-1 routing will be summarized into the

level-2 backbone with the configured address/mask value. The default is NO SUMMARY-ADDRESS.

[no] default-information allowed {in | out}

Controls the handling of default information between multiple IGRP processes. The NO DEFAULT-INFORMATION ALLOWED IN subcommand causes IGRP exterior or default routes to be suppressed when received by an IGRP process. Normally, exterior routes are always accepted. The NO DEFAULT-INFORMATION ALLOWED OUT subcommand causes IGRP exterior routes to be suppressed in updates. Default information is normally passed between IGRP processes.

[no] default-information originate

Explicitly configures EGP to generate a default route. If the next hop for the default route can be advertised as a third party, it will be included as a third party.

[no] default-information originate metric *metric-value* **metric-type** *type-value*

Allows you to force the AS boundary router redistribute OSPF routes. Always use this command with a REDISTRIBUTE command.

The keyword **originate** causes the router to generate a default external route into an OSPF domain.

The keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the default route. The *metric-value* argument is a dimensionless link state cost, formed as a 24-bit decimal number.

The keyword/argument pair **metric-type** *type-value* specifies the external link type associated with the default route advertised into the OSPF routing domain. The *type-value* argument can assume one of two values:

- 1—Type 1 external route
- 2—Type 2 external route

If a **metric-type** is not specified, the router adopts Type 2 external route.

The NO DEFAULT-INFORMATION command disables generation of a default route into the specified OSPF routing domain.

[no] default-metric *bandwidth delay reliability loading mtu*

Sets metrics for IGRP only. The argument *bandwidth* is the minimum bandwidth of the route in kilobits per second. The argument *delay* is

the route delay in tens of microseconds. The argument *reliability* is the likelihood of successful packet transmission expressed as a number between 0 and 255 (255 is 100% reliability). The argument *loading* is the effective bandwidth of the route in kilobits per second. The argument *mtu* is the minimum maximum transmission NO DEFAULT-METRIC command causes the current routing protocol to return to using the built-in, automatic metric translations.

[**no**] **default-metric** *number*

Sets metrics for RIP, EGP, BGP, and Hello, which use scalar, single-valued metrics. The argument *number* is the default metric value (an unsigned integer) appropriate for the specified routing protocol. The NO DEFAULT-METRIC command causes the current routing protocol to return to using the built-in, automatic metric translations.

[**no**] **distance bgp** *external-distance internal-distance local-distance*

Specifies administrative distance.

The argument *external-distance* specifies the value for BGP external routes. External routes are those routes for which the best path is learned from a neighbor external to the autonomous system. Acceptable values are positive, nonzero integers.

The argument *internal-distance* specifies the value for BGP internal routes. Internal routes are routes that are learned from another BGP entity within the same autonomous system. Acceptable values are positive, nonzero integers.

The argument *local-distance* specifies the value for BGP local routes. Local routes are those networks listed with a NETWORK command (possibly as backdoors) for that router or networks that are being redistributed from another process.

By default, the administrative distances are as follows

- *External-distance*—20
- *Internal-distance*—200
- *Local-distance*—200

[**no**] **distance** *weight* [*address mask*] [*access-list-number*]

Defines or deletes an administrative distance. The argument *weight* is an integer from 10 to 255 that specifies the administrative distance. Used alone, the argument *weight* specifies a default administrative distance that the router uses when no other specification exists for a routing information source. The optional argument pair *address* and *mask* specifies a particular router or group of routers to which the weight applies. The argument *address*

is an Internet address that specifies a router, network, or subnet. The argument *mask* (in dotted-decimal format) specifies which bits, if any, to ignore in the address value; a set bit in the *mask* argument instructs the router to ignore the corresponding bit in the address value. The optional argument *access-list-number* is the number of a standard IP access list.

[**no**] **distribute-list** *access-list-number* **in** [*interface-name*]

Filters networks received in updates. The argument *access-list-number* is a standard IP access list number. Use the keyword **in** to suppress incoming routing updates. The optional argument *interface-name* specifies the interface on which the access list should be applied to incoming updates. If no interface is specified, the access list will be applied to all incoming updates.

[**no**] **distribute-list** *access-list-number* **out** [*interface-name* | *routing-process*]

Suppresses networks from being sent in updates. The argument *access-list-number* is a standard IP access list number. Use the keyword **out** to apply the access list to outgoing routing updates. To filter a routing update sent on a specific interface, you can specify the interface. When redistributing networks, you can specify a routing process name.

[**no**] **metric holddown**

Disables or re-enables holddown (IGRP only).

[**no**] **metric maximum-hops** *hops*

Causes the IP routing software to advertise as unreachable routes with a hop count greater than the value assigned to the *hops* argument (IGRP only). The default value is 100 hops; the maximum value is 255.

[**no**] **metric weights** TOS K1 K2 K3 K4 K5

Allows the tuning of the IGRP metric calculation for a particular TOS. The TOS parameter currently must always be zero. Parameters K1 through K5 are constants in the equation that converts an IGRP metric vector into a scalar quantity.

[**no**] **neighbor** *address*

Creates a list of neighbor routers. The argument *address* is the IP address of a peer router with which routing information will be exchanged. The **no** form of the command removes an entry.

[**no**] **neighbor any** [*list*]

Controls how a BGP process determines which neighbors will be treated as peers.

If the *list* argument is specified, the neighbor *must* be accepted by the access list to be allowed to peer with the BGP process.

[**no**] **neighbor any** [*list*]
[**no**] **neighbor any third-party** *address* [**internal** | **external**]

Controls how an EGP process determines which neighbors will be treated as peers; used with the ROUTER EGP 0 router subcommand.

If the *list* argument is specified, the neighbor must be accepted by the access list to be allowed to peer with the EGP process.

The keyword/argument pair **third-party** *address* allows the specified address to be used as the next hop in EGP advertisements.

The optional keywords **internal** or **external** indicate whether the third-party router should be listed in the internal or external section of the EGP update.

[**no**] **neighbor** *address* **distribute-list** *list* {**in** | **out**}

Distributes neighbor information as specified in an access list list for BGP. You specify the access list to be applied to incoming or outgoing updates with the **in** and **out** keywords.

[**no**] **neighbor** *address* **filter-list** *list* **in**
[**no**] **neighbor** *address* **filter-list** *list* **out**
[**no**] **neighbor** *address* **filter-list** *list* **weight** *weight*

Establishes filters using access lists defined with the IP AS-PATH ACCESS-LIST command.

The argument *address* is the address of the neighbor.

The argument *list* is a predefined BGP access list number.

The keywords **in** and **out** specify whether you are applying the access list to incoming or outgoing routes.

The keyword/argument pair **weight** *weight* assigns a relative importance to a specific list. Any number weight filters are allowed on a per-neighbor basis, but only one in or out filter is allowed

[**no**] **neighbor** *address* **interface** *type unit-number* [**priority** *8-bit-number*] [*poll-interval number-of-seconds*]

Configures OSPF-based routers interconnecting to nonbroadcast networks.

The argument *address* is the specific interface IP address of the neighbor.

The keyword/argument sequence **interface** *type unit-number* identifies the specific router interface for this NEIGHBOR command. The interface must be connected to a nonbroadcast, multiaccess type network. In addition, the neighbor specified must be eligible to be a designated router or backup designated router.

The keyword/argument pair **priority** *8-bit* number is the router priority value of the nonbroadcast neighbor associated with the IP address specified.

If a neighboring router has become inactive (Hello packets have not been seen for Router Dead Interval period), it may still be necessary to send Hello packets to the dead neighbor. These Hello packets will be sent at a reduced rate called the *Poll Interval* (Poll Interval).

Specify this interval with the keyword/argument pair **poll-interval** *number-of-seconds*. The argument *number-of-seconds* is an unsigned integer value. RFC 1247 recommends that this value should be much larger than the HelloInterval. The default is 2 minutes (120 seconds).

neighbor *address* **remote-as** *number*
no neighbor *address*

Adds a neighbor entry to the routing table for BGP. The keywords *address* and *number* specify the IP address and AS of the neighbor router.

[**no**] **neighbor** *address* **third-party** *third-party-ip-address* [**internal** | **external**]

Adds third-party information to routing updates. The argument *third-party-ip-address* is the address of the third party on the network shared by the DECbrouter 90 and the EGP peer specified by the address argument. The optional keyword **internal** or **external** indicates whether the third party router should be listed in the internal or external section of the EGP update. Normally, all networks are mentioned in the internal section. You can enter this command multiple times.

[**no**] **neighbor** *address* **weight** *weight*

Specifies a weight to assign to a specific neighbor connection indicated by the argument *address*. The route with the highest weight is chosen as the preferred route when multiple routes are available to a particular network.

[**no**] **neighbor** *ip-address*

Adds to a list of neighbor routers. The argument *ip-address* is the IP address of a peer router with which routing information will be exchanged. Multiple NEIGHBOR subcommands may be used to specify additional neighbors or peers. The NO NEIGHBOR subcommand followed by an IP address removes a peer from the list.

[**no**] **neighbor** *ip-address* **version** *value*

Configures router to handle only a specific version of BGP.

The argument *ip-address* is the address of the BGP-speaking neighbor; the version value can be set to 2 to force the router to only use Version 2 with the specified neighbor. The default is to speak Version 3 of BGP and dynamically negotiate down to Version 2 if requested. The NO NEIGHBOR *ip-address* VERSION VALUE command returns the version to the default state for that neighbor.

[**no**] **network** *network-number*

Specifies the list of networks with the NETWORK router configuration subcommand. The argument *network-number* is a network number in dotted IP notation. Note that this number must not contain subnet information. You may specify multiple NETWORK subcommands. Use this version of the command for BGP, EGP, RIP, Hello, and IGRP protocols. The **no** version of this command removes the specified network number.

[**no**] **network** *address* **backdoor**

Specifies a backdoor route to a BGP border router that will provide better information about the network.

[**no**] **network** *address wildcard-mask* **area** *area-id*

Specifies a range of IP addresses for any area in which OSPF is to be used as a routing protocol. The *address* and *wildcard-mask* pair together define a range of IP addresses to be associated with a specific OSPF area. The argument *address* is formed as an IP address. The argument *wildcard-mask* is an IP-address-type mask that includes "don't care" bits.

The keyword/variable argument pair **area** *area-id* specifies an area to be associated with the OSPF address range as defined in the same NETWORK command. The argument *area-id* can be specified as either a decimal value or as an IP address. If you intend to associate areas with IP subnets, you can specify a subnet address as the *area-id*.

[**no**] **offset-list** *list* {**in** | **out**} *offset*

Adds or removes a positive offset to incoming and outgoing metrics (as indicated by the in and out keywords) for networks matching an access list specified with the list argument (for IGRP, RIP and Hello only). If the argument *list* is zero, the argument supplied to *offset* is applied to all metrics. If *offset* is zero, no action is taken. For IGRP, the offset is added to the delay component only.

[**no**] **passive-interface** *interface*

Disables or enables sending routing updates on an interface. The argument *interface* specifies a particular interface.

[**no**] **redistribute** *process-name* [*AS-number*]

Passes routing information among routing protocols. The argument *process-name* specifies a routing information source using one of the keywords: **static**, **rip**, **bgp**, **egp**, **hello**, **igrp**. Use the optional argument *AS-number* when you specify the **bgp**, **igrp** or **egp** keyword to specify the autonomous system number.

[**no**] **redistribute** *protocol* [*source-id*]
[**metric** *metric-value*]
[**metric-type** *type-value*]
[**tag** *tag-value*]
[**subnets**]

Redistributes routes from other OSPF routing domains and non-OSPF routing domains into a specific OSPF routing domain. The argument *protocol* is the source protocol from which routes are being redistributed. It can be one of the keywords that follow.

- **bgp**
- **egp**
- **hello**
- **igrp**
- **ospf**
- **rip**
- **static**

The optional argument *source-id* is either an autonomous system (IGRP) or an appropriate OSPF process id from which routes are to be redistributed. This value takes the form of a positive integer. If the keywords **hello** or **rip** are used, then no *source-id* value is specified.

The optional keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the redistributed route. The *metric-value* argument is a dimensionless link state cost formed as a 24-bit decimal number. If a value is not specified for this option, and no value is specified using the DEFAULT-METRIC router subcommand, the default **metric** value is 20.

The keyword/argument pair **metric-type** *type-value* specifies the external link type associated with the default route advertised in the OSPF routing domain. The *type-value* argument can assume one of two values:

- 1—Type 1 external route
- 2—Type 2 external route

If a **metric-type** is not specified, the router adopts Type 2 external route.

The optional keyword/argument pair **tag** *tag-value* specifies a 32-bit decimal value attached to each external route. The optional keyword **subnets** specifies the scope of redistribution for the specified protocol.

[**no**] **redistribute ospf** *ospf-process-id*
[**metric** *metric-value*]
[**match internal** | **external** *type-value* | **external** *type-value*]

Redistributes routes from OSPF to other routing domains. You can select any combination of **internal** and/or **external** (Type 1 or Type 2) routes to redistribute. By default, if routes are redistributed into EGP or BGP, only **internal** routes are redistributed. Otherwise all routes are redistributed by default.

The argument *ospf-process-id* is the OSPF process id from which routes are to be redistributed. This value takes the form of either a decimal number or an IP address.

The optional keyword/argument pair **metric** *metric-value* maps OSPF cost assigned to the redistributed route into the destination routing domain metric type. Use a value consistent with the destination protocol.

The optional keyword **match** specifies the criteria by which OSPF routes are redistributed into other routing domains. The keywords used are **internal** and **external**. The keyword **internal** refers to routes that are internal to a specific AS; the keyword **external** refers to routes that are external to the AS, but are to imported to OSPF as external routes.

The argument *type-value* specifies the external route type to be redistributed into other routing domains. The *type-value* argument can assume one of two values:

- 1—Type 1 external route

- 2—Type 2 external route

There is no default value.

no synchronization
synchronization

Disables the synchronization between BGP and your IGP. Usually, a BGP speaker does not advertise a route to an external neighbor unless that route is local or exists in the IGP. The NO SYNCHRONIZATION command allows a router to advertise a network route without waiting for the IGP. This feature allows routers within an AS to have the route before BGP makes it available to other ASes. Use SYNCHRONIZATION if there are routers in the AS that do not speak BGP. The default is SYNCHRONIZATION.

[**no**] **timers basic** *update invalid holddown flush sleeptime*

Adjusts timers. The argument *update* is the rate at which updates are sent. This is the fundamental timing parameter of the routing protocol. The argument *invalid* is an interval of time after which a route is declared invalid; it should be three times the value of update. The argument *holddown* is the interval during which routing information regarding better paths is suppressed. It should be at least three times the value of *update*. The argument *flush* is the amount of time that must pass before the route is removed from the routing table; it must be at least the sum of *invalid* and *holddown*. The argument *sleeptime* is used to postpone routing updates for the specified number of milliseconds. Note that other timing values are specified in seconds. The *sleeptime* value should be less than the *update* time. If the *sleeptime* is greater than the *update* time, routing tables will become unsynchronized. The no form of the command restores the default.

[**no**] **timers bgp** *keepalive holdtime*

Adjusts BGP timers. The argument *keepalive* is the frequency in seconds with which the router sends keepalive messages to its peer (default 60 seconds), and where *holdtime* is the interval in seconds after not receiving a keepalive message that the router declares a peer dead (default 180 seconds). The **no** form of the command restores the default.

[**no**] **timers egp** *hello poll*

Adjusts the EGP timers. The argument *hello* adjusts the interval at which Hello messages are sent. The default is set to 60 seconds. The argument *poll* adjusts the interval at which polling is performed. The default is set to 180 seconds; the **no** form of the command restores the default.

[**no**] **variance** *multiplier*

Controls load balancing in an IGRP-based internet. The argument *multiplier* defines the amount of metric variance that will be accepted. Acceptable

values are nonzero, positive integers. By default, the amount of variance is set to one. The **no** version resets variance to one.

IP Routing Interface Subcommands

This section provides alphabetical lists of the interface subcommands used with the IP routing protocols.

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

Specifies the IP address on an interface. The argument *address* supplies the address; the argument *mask* supplies the subnet mask. The optional keyword **secondary** allows multiple IP addresses per interface. The **no** version of the command removes the specified secondary address association.

[no] ip gdp

Enables or disables GDP routing, with all default parameters. Reporting interval is five seconds for Ethernet and zero seconds for nonbroadcast media. The priority is 100, and the hold time is 15 seconds.

[no] ip gdp holdtime *seconds*

Enables or disables GDP routing, specifying hold time in *seconds* and keeping all other parameters (priority and reporting interval) at their default settings.

[no] ip gdp priority *number*

Enables or disables GDP routing with a priority number you specify. Report time remains at 5 seconds for Ethernets, and the hold time remains 15 seconds.

[no] ip gdp reporttime *seconds*

Enables or disables GDP routing with a report time you specify. The priority remains 100, and the hold time remains 15 seconds.

[**no**] **ip irdp**

The default is for IRDP processing not to be enabled. If you enable IRDP processing, you will use the default parameters.

ip irdp preference *number*
ip irdp maxadvertinterval *seconds*
ip irdp minadvertinterval *seconds*
ip irdp holdtime *seconds*
ip irdp address *address* [*number*]

The **preference** default value is 100. A lower value increases this router's preference level. The allowed range is from 0 to 255. You can modify a particular router's preference value so that it will only be selected if other routers are down or so that it will be the preferred router to home to.

The maximum advertised interval **maxadvertinterval** default value is 600 seconds. This value sets the maximum interval between advertisements.

The minimum advertised interval **minadvertinterval** default value is 400 seconds. If you change **maxadvertinterval**, it defaults to two-thirds of the new value. This value sets the minimum interval between advertisements.

The **holdtime** value determines how long the advertisements are valid.

[**no**] **ip ospf authentication-key** *8-bytes-of-password*

Assigns a specific password to be used by neighboring routers on a wire that are using OSPF's simple password authentication.

The argument *8-bytes-of-password* is any continuous string of characters up to eight bytes in length that you can enter from the keyboard.

[**no**] **ip ospf cost** *cost*

Explicitly specifies the cost of sending a packet on an interface. The argument cost is expressed as the link state metric. It is a dimensionless integer value, that is always greater than zero. This value is advertised as the link cost in the router's router links advertisement. Digital does not support TOS, so you can assign only one cost per interface.

[**no**] **ip ospf dead-interval** *number-of-seconds*

Sets the number of seconds that a router's Hello packets have not been seen before its neighbors declare the router down. This value is advertised in the router's Hello packets in the *DeadInt* field.

The argument *number-of-seconds* is an unsigned integer value.

The default is four times the HelloInterval. As with the HelloInterval, this value must be the same for all routers attached to a common network.

[**no**] **ip ospf hello-interval** *number-of-seconds*

Specifies the length of time, in seconds, between the Hello packets that the router sends on the interface.

The argument *number-of-seconds* is an unsigned integer value. This value is advertised in the router's Hello packets. It must be the same for all routers attached to a common network. The smaller the Hello interval, the faster topological changes will be detected, but more routing traffic will ensue.

[**no**] **ip ospf priority** *8-bit-number*

Sets the router priority, which helps determine the designated router for a network.

The argument *8-bit-number* is an 8-bit unsigned integer.

The default router priority is 1.

[**no**] **ip ospf retransmit-interval** *number-of-seconds*

Sets the number of seconds between link state advertisement retransmissions for adjacencies belonging to the interface.

The value for the *number-of-seconds* argument is an integer number that should be greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links.

The default value is five seconds.

[**no**] **ip ospf transmit-delay** *number-of-seconds*

Sets the estimated number of seconds it takes to transmit a link state update packet on the interface.

Link state advertisements in the update packet must have their age incremented by this amount before transmission. The value assigned should take into account the transmission and propagation delays for the interface.

The argument *number-of-seconds* is an integer value that must be greater than zero. The default value is one second.

[no] ip split-horizon

Enables or disables the split horizon mechanism. For all interfaces except those for which either frame relay or SMDS encapsulation is enabled, the default condition for this command is **ip** split-horizon; in other words, the split horizon feature is active. If the interface configuration includes either the ENCAPSULATION FRAME-RELAY or ENCAPSULATION SMDS commands, then the default is for split horizon to be disabled. Split horizon is *not* disabled by default for interfaces using any of the X.25 encapsulations. If split horizon has been disabled on an interface and you wish to enable it to use the IP SPLIT-HORIZON interface subcommand to restore the split horizon mechanism.

ip router isis *tag*

Configures an IS-IS routing process over an interface. The argument *tag* should be the same routing process name assigned with the ROUTER ISIS command.

[no] isis circuit-type [level-1 | level-1-2 | level-2-only]

Configures the type of adjacency desired for this interface. If **level-1** is specified, at most a Level 1 adjacency can be established if there is at least one area address in common between this system and its neighbors. If **level-1-2** is specified, a Level 1 and 2 adjacency is established if the neighbor is configured as **level-1-2** and there is at least one area in common. If there is no area in common, a Level 2 adjacency is established. If **level-2-only** is specified, a Level 2 adjacency is established if and only if the neighbor is configured exclusively to be a Level 2 router. The default value for this command is **level-1-2**. The NO ISIS CIRCUIT-TYPE command resets the circuit type to Level l and Level 2.

[no] isis csnp-interval seconds [level-1 | level-2]

Specifies the interval of time between transmission of Complete Sequence Number PDUs. The argument value is the length of time between transmissions.

[**no**] **isis hello-interval** *seconds* [**level-1 | level-2**]

Specifies the length of time in seconds between hello packets the router sends on the interface. The argument value is unsigned integer value.

[**no**] **isis metric** *default-metric* [**level-1 | level-2**]

Configures the metric (or cost) for the specified interface. The default value for the *default-metric* argument is ten. You can configure this metric for Level 1 and/or Level 2 routing. The NO ISIS METRIC command resets the *default-metric* value to ten. Specifying the **level-1** or **level-2** optional keywords resets metric only for Level 1 or Level 2 routing, respectively.

[**no**] **isis password** *password* [**level-1 | level-2**]

Configures the authentication password for an interface. Different passwords can be assigned for different routing levels using the optional **level-1** and **level-2** keyword arguments. By default authentication is disabled. The NO ISIS PASSWORD command disables authentication for IS-IS. Specification of the **level-1** or **level-2** optional keywords disables the password only for Level 1 or Level 2 routing, respectively.

[**no**] **isis priority** *value* [**level-1 | level-2**]

Configures the priority to use for designated router election. Priorities can be configured for Level 1 and Level 2 individually. The default value is 64. The NO ISIS PRIORITY command resets priority to 64. The higher the number, the higher the priority. Specification of the **level-1** or **level-2** optional keywords resets priority only for Level 1 or Level 2 routing, respectively.

[**no**] **isis retransmit-interval** *number-of-seconds*

Configures the number of seconds between retransmission of IS-IS LSP retransmission for point-to-point links. The argument *number-of-seconds* is the length of time between retransmissions.

[**no**] **keepalive** [*seconds*]

Adjusts the keepalive timer for a specific interface. If the optional argument *seconds* is not specified, a default of ten seconds is assumed.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

Z