

Router Products Configuration and Reference

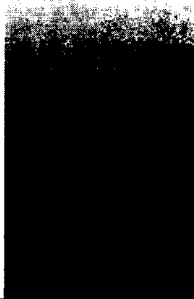


Volume II

Software Release 9.1

September 1992





*Router Products
Configuration and Reference
Volume II*

September 1992
Software Release 9.1

Corporate Headquarters:
1525 O'Brien Drive
Menlo Park, California 94026 USA
1 (415) 326-1941
1-800-553-NETS

Customer Order Number: DOC-R9.1
Cisco Document Assembly Number: 83-0039-01
Text Part Number: 78-0959-01



The products and specifications, configurations, and other technical information regarding the products contained in this manual are subject to change without notice. All statements, technical information, and recommendations contained in this manual are believed to be accurate and reliable but are presented without warranty of any kind, express or implied, and users must take full responsibility for their application of any products specified in this manual. THIS MANUAL IS PROVIDED "AS IS" WITH ALL FAULTS. CISCO DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING THOSE OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL OR INCIDENTAL DAMAGES, INCLUDING WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Some states do not allow limitation or exclusion of liability for consequential or incidental damages or limitation on how long implied warranties last, so the above limitations or exclusions may not apply to you. This warranty gives Customers specific legal rights, and you may also have other rights that vary from state to state.

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. This equipment has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright (c) 1981 Regents of the University of California.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by the University of California, Berkeley. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Point-to-Point Protocol. Copyright (c) 1989 Carnegie Mellon University. All rights reserved.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by Carnegie Mellon University. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Notice of Restricted Rights:

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) of the Commercial Computer Software - Restricted Rights clause at FAR § 52.227-19 and subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS § 252.227-7013.

The information in this manual is subject to change without notice.

AGS+, APPI, ASM, CGS, cisco, Cisco, IGRP, IGS, MGS, NetCentral, NetCentral Station, Netscape, NetWorkers, The Packet, SMARTnet, and the Cisco logo are trademarks of Cisco Systems, Inc.

All other products or services mentioned in this document are the trademarks, service marks, registered trademarks, or registered service marks of their respective owners.

Router Products Configuration and Reference

Copyright 1988-1992, Cisco Systems, Inc.

All rights reserved. Printed in USA.

Table of Contents

About This Manual xlv

- Audience and Scope xlv
- Document Organization and Use xlv
- Document Conventions xlvi
- Related Cisco Documentation xlvii

Service and Support xlix

- Warranty Information xlix
- Maintenance Agreements xlix
- Customer Support l

Obtaining Additional Information li

- Ordering Additional Cisco Publications li
- Obtaining Cisco Technical Information Electronically li
- Obtaining Information from Other Sources lii
 - Obtaining RFCs lii
 - Obtaining Technical Standards liv

Part One Configuring and Managing the Router

Chapter 1 Router Product Overview 1-1

- Capabilities of the Router 1-1
 - Support for Multiple Network Protocols 1-1
 - Dynamic Network Routing 1-2
 - Support for Standard Media 1-3

Chapter 2 First-Time Startup and Basic Configuration 2-1

- Using the Setup Facility for Basic Configuration 2-1
 - Capabilities of the Setup Command Facility 2-2
 - Using the Setup Command Facility 2-2
 - First-Time System Startup 2-3

Using the EXEC Command Interpreter	2-8
Command Syntax	2-8
EXEC Command Levels	2-8
Entering Configuration Mode	2-10
Entering the Configuration Commands	2-11
Examples of Configuration Files	2-11
Creating the Configuration File	2-13
Using TFTP to Load Configuration Files or System Images	2-16
Using the Flash Memory Card for Storing and Booting System Software	2-18
Reloading the Operating System	2-18

Chapter 3 *Using Terminals 3-1*

Making and Managing Terminal Connections	3-1
Making Telnet Connections	3-1
Establishing Multiple Connections	3-2
Listing Connections	3-2
Resuming a Previous Connection	3-3
Naming a Connection	3-3
Exiting a Session	3-3
Disconnecting	3-4
Resetting a Line	3-4
Executing Special Telnet Commands	3-4
Incoming Telnet Connections	3-5
LAT EXEC Commands	3-6
Displaying TCP Connections	3-6
Displaying Active Sessions	3-6
Displaying Information About Active Lines	3-7
Changing Terminal Parameters	3-7
Changing the Terminal Screen Width	3-7
Changing the Terminal Screen Length	3-8
Changing the Terminal Escape Character	3-8
Displaying the Debug Messages on the Console and Terminals	3-8
Establishing Input Notification	3-9
Changing the Character Padding	3-9
Displaying Terminal Parameter Settings	3-9

Setting Widths for International Character Sets at the User Level	3-10
Using the DEC MOP Server	3-11
EXEC Terminal Command Summary	3-12

Chapter 4

Configuring the System 4-1

Configuring the Global System Parameters	4-1
Setting the Host Name	4-1
Displaying Banner Messages	4-2
Setting Default Widths for International Character Sets	4-4
Setting the System Buffers	4-5
Setting Configuration File Specifications	4-7
Storing and Booting System Software Using the Flash Memory Card	4-9
Establishing Passwords and System Security	4-22
Establishing the Privileged-Level Password	4-22
Encrypting Passwords	4-24
Establishing Terminal Access Control	4-25
Establishing Privileged-Level TACACS	4-27
Configuring TACACS Accounting	4-28
Authenticating User Names	4-30
Configuring the Simple Network Management Protocol (SNMP)	4-31
Enabling and Disabling the SNMP Server	4-32
Defining the SNMP Server Access List	4-32
Setting the Community String	4-32
Establishing the Message Queue Length	4-33
Establishing Packet Filtering	4-34
Establishing the TRAP Message Recipient	4-34
Establishing TRAP Message Authentication	4-35
Establishing the TRAP Message Timeout	4-35
Enabling SNMP System Shutdown Feature	4-36
Configuring the Trivial File Transfer Protocol (TFTP) Server	4-36
Tailoring Use of Network Services	4-37
Redirecting System Error Messages	4-38
Enabling Message Logging	4-38
Logging Messages to an Internal Buffer	4-38
Logging Messages to the Console	4-39

Logging Messages to Another Monitor	4-39
Logging Messages to a UNIX Syslog Server	4-40
Limiting Messages to a Syslog Server	4-40
Configuring Console and Virtual Terminal Lines	4-41
Starting Line Configuration	4-41
Configuring the CPU Auxiliary Port	4-42
Establishing Line Passwords	4-43
Setting Widths for International Character Sets for the Interface	4-44
Establishing Connection Restrictions	4-45
Suppressing Banner Messages	4-45
Turning On and Off the Vacant Banner	4-46
Setting the Escape Character	4-46
Setting the Terminal Location	4-47
Setting the EXEC Timeout Intervals	4-47
Setting the Screen Length	4-48
Setting Notification	4-48
Setting Character Padding	4-49
Global System Configuration Command Summary	4-49
Line Configuration Subcommand Summary	4-57

Chapter 5 *Managing and Monitoring the System 5-1*

Monitoring System Processes	5-1
Displaying Buffer Pool Statistics	5-2
Displaying System Memory Statistics	5-3
Displaying Active System Processes	5-4
Displaying Memory Utilization	5-5
Displaying Stack Utilization	5-6
Displaying the System Configuration	5-7
Displaying the Error Logging Conditions	5-7
Displaying Protocol Information	5-8
Obtaining AGS+ System Temperature and Voltage Readings	5-8
Verifying Installation of Flash Memory and Displaying Flash Statistics	5-10
Troubleshooting Network Operations	5-11
Real-time Debugging Support	5-11
Testing Connectivity with the Ping Command	5-13

Checking Routes with the Trace Command	5-14
Writing System Configuration Information	5-14
Testing the System	5-15
16-Mbps Token Ring Card Test	5-15
EXEC System Management Command Summary	5-16

Part Two
Chapter 6

<i>Configuring Interfaces</i>	
<i>Configuring the Interfaces 6-1</i>	
Specifying an Interface	6-1
Adding a Descriptive Name to an Interface	6-2
Shutting Down and Restarting an Interface	6-3
Clearing Interface Counters	6-3
Displaying Information About an Interface	6-4
Displaying the System Configuration	6-4
Displaying Controller Status	6-5
Displaying Interface Statistics	6-7
Serial Interface Support	6-8
Specifying a Serial Interface	6-8
Serial Encapsulation Methods	6-9
Maintaining the Serial Interface	6-10
Monitoring the Serial Interface	6-11
Debugging the Serial Interface	6-14
ISDN Basic Rate Interface (BRI) Support	6-15
Specifying an ISDN BRI	6-15
ISDN BRI Encapsulation Methods	6-15
Maintaining the ISDN BRI	6-16
Monitoring the ISDN BRI	6-16
Debugging the ISDN BRI	6-19
Ethernet Interface Support	6-20
Specifying an Ethernet Interface	6-20
Ethernet Encapsulation Methods	6-21
Maintaining the Ethernet Interface	6-21
Monitoring the Ethernet Interface	6-22
Debugging the Ethernet Interface	6-25

Token Ring Interface Support	6-26
Specifying a Token Ring Interface	6-26
Configuring Early Token Release	6-27
Token Ring Encapsulation Methods	6-28
Maintaining the Token Ring Interface	6-28
Monitoring the Token Ring Interface	6-28
Debugging the Token Ring Interface	6-32
FDDI Support	6-33
Specifying an FDDI	6-33
FDDI Encapsulation Methods	6-34
Monitoring the FDDI	6-34
Debugging the FDDI	6-41
FDDI Special Commands and Configurations	6-41
Controlling FDDI SMT Frame Buffering	6-45
High-Speed Serial Interface (HSSI) Support	6-46
Specifying the HSSI	6-47
HSSI Encapsulation Methods	6-47
Maintaining the HSSI	6-47
Monitoring the HSSI	6-48
Debugging the HSSI	6-51
UltraNet Interface Support	6-51
Specifying an UltraNet Interface	6-52
UltraNet Encapsulation Method	6-52
Maintaining the UltraNet Interface	6-52
Monitoring the UltraNet Interface	6-52
Statically Assigning UltraNet Addresses	6-56
Configuring Dial Backup Service	6-57
Configuring the Dial Backup Line	6-57
Defining the Traffic Load Threshold	6-58
Defining the Backup Line Delay	6-58
Dial Backup Configuration Examples	6-59
Configuring Dial-on-Demand Routing	6-60
Cisco's DDR Implementation	6-61
Specifying Dialer Type	6-62
Controlling Dialer Idle Time	6-63
Setting Idle Time for Busy Interfaces	6-63

Specifying Dialer Enable Time	6-64
Specifying Carrier Wait Time	6-64
Specifying a Single DDR Telephone Number	6-65
Multiple Destinations	6-66
Dialer Rotary Groups	6-69
Assigning Access Lists to a DDR Interface	6-70
Using Access Lists with DDR	6-71
Dial-on-Demand Access-List Configuration Examples	6-72
Monitoring Dial-on-Demand Routing	6-76
Debugging Dial-on-Demand Routing	6-77
Configuring the Point-to-Point Protocol	6-78
Challenge Handshake Access Protocol (CHAP)	6-79
Enabling CHAP on the Interface	6-79
Configuring the Null Interface	6-82
Global Configuration Command Summary	6-83
Interface Support Subcommand Summary	6-84
Interface Support EXEC Command Summary	6-90

Chapter 7

<i>Adjusting Interface Characteristics</i>	7-1
Adjusting Serial Interface Characteristics	7-1
Specifying Transmit Delay	7-1
Configuring DTR Signal Pulsing	7-2
Configuring for DCE Appliques	7-2
Adjusting Characteristics That Apply to All Interface Types	7-3
Configuring Switching and Scheduling Priorities	7-3
Priority Output Queuing	7-4
Controlling Interface Hold Queues	7-10
Setting Bandwidth	7-11
Setting Delay	7-11
Setting Error Count Reset Frequency	7-12
Setting and Adjusting Packet Sizes	7-12
Enabling the Loopback Test	7-13
Enabling Loopback on the HSSI	7-14
Enabling Loopback on an UltraNet Connection	7-17
Enabling Loopback on MCI and SCI Serial Cards	7-17
Enabling Loopback on MCI and MEC Ethernet Cards	7-17

- Enabling Loopback on the CSC-FCI FDDI Card 7-18
- Enabling Loopback on Token Ring Cards 7-18
- Global Configuration Subcommand Summary 7-19
- Interface Configuration Subcommand Summary 7-20

Chapter 8

Configuring Packet-Switched Software 8-1

Configuring LAPB 8-1

- Running a Single Network Protocol 8-2

- Running Multiple Network Protocols 8-2

- Sample Configuration of LAPB Encapsulation 8-2

- Setting the X.25 Level 2 (LAPB) Parameters 8-3

- Setting Frame Parameters 8-4

Configuring X.25 8-5

- Overview of Cisco X.25 Support 8-6

- Transporting LAN Protocols Across an X.25 PDN 8-7

- Routing X.25 Traffic through a LAN 8-17

- Setting the X.25 Parameters 8-23

- Netbooting over X.25 8-34

- X.25 TCP Header Compression 8-35

- Bridging on X.25 8-35

- Configuring Datagram Transport on DDN Networks 8-36

- Maintaining X.25 8-42

- Monitoring X.25 Level 3 Operations 8-42

- Debugging X.25 8-44

Configuring CMNS Routing Support 8-45

- Enabling CMNS 8-46

- Specifying CMNS Address Mappings 8-46

- CMNS Configuration Examples 8-47

- Monitoring CMNS Traffic Activity 8-51

- Debugging CMNS 8-54

Configuring Frame Relay 8-54

- Configuring the Hardware 8-55

- Specifying Frame Relay Encapsulation 8-56

- Enabling ANSI Frame Relay LMI 8-56

- Setting the Frame Relay Keepalive Timer 8-57

- Mapping Between an Address and the DLCI 8-57

Requesting Short Status Messages	8-59
Setting a Local DLCI	8-60
Defining a DLCI for Multicast	8-60
Netbooting over Frame Relay	8-61
Frame Relay Configuration Examples	8-61
Monitoring Frame Relay	8-62
Debugging Frame Relay	8-65
Configuring Switched Multimegabit Data Services (SMDS)	8-65
Special SMDS Requirements	8-66
Configuring SMDS	8-66
Using SMDS Addresses	8-67
Enabling SMDS	8-68
Configuring Specific Protocols	8-71
IP Fast Switching	8-73
SMDS Configuration Examples	8-74
Monitoring SMDS Service	8-75
Debugging SMDS	8-76
Packet-Switched Software Global Configuration Command Summary	8-77
X.25 Global Configuration Command Summary	8-77
Packet-Switched Software Interface Subcommand Summary	8-78
CMNS Interface Subcommand Summary	8-78
Frame Relay Interface Subcommand Summary	8-79
LAPB Interface Subcommand Summary	8-80
SMDS Interface Subcommand Summary	8-81
X.25 Interface Subcommand Summary	8-82
X.25 EXEC Command Summary	8-90

Part Three

Configuring Routing

Chapter 9

Routing Apollo Domain 9-1

Cisco's Implementation of Apollo Domain 9-1

Apollo Domain Addresses 9-2

Configuring Apollo Domain Routing 9-2

 Enabling Apollo Domain Routing 9-3

 Assigning the Apollo Domain Network Numbers 9-3

Configuring Static Routes	9-3
Configuring Maximum Paths	9-4
Setting Apollo Update Timers	9-4
Configuring Apollo Domain Access Lists	9-5
Specifying Apollo Domain Access Lists	9-5
Defining Access Groups	9-6
Monitoring the Apollo Domain Network	9-7
Displaying Apollo Interface Parameters	9-7
Displaying Apollo Routes	9-7
Displaying Apollo Traffic Statistics	9-8
Displaying the Apollo ARP Table	9-8
Debugging the Apollo Domain Network	9-9
Apollo Domain Global Configuration Command Summary	9-9
Apollo Domain Interface Subcommand Summary	9-10

Chapter 10

Routing AppleTalk 10-1

Cisco's Implementation of AppleTalk	10-1
Extended (Phase II) Versus Nonextended (Phase I) AppleTalk	10-4
Nonextended AppleTalk Addressing	10-5
AppleTalk Zones	10-5
Name Binding Protocol (NBP)	10-6
Zone Information Protocol (ZIP)	10-7
Dynamic Configuration (Discovery Mode)	10-7
Extended AppleTalk Addressing	10-9
AppleTalk Name Registration	10-10
AppleTalk Responder Support	10-10
Configuring AppleTalk Routing	10-11
Configuration Overview	10-11
Configuration Guidelines (Compatibility Rules)	10-12
Enabling AppleTalk Routing	10-13
Assigning Nonextended (Phase I) AppleTalk Address	10-13
Assigning a Cable Range for Extended AppleTalk (Phase II)	10-13
Assigning a Zone Name	10-14
Setting and Resetting Discovery Mode	10-15
Configuring IP Encapsulation of AppleTalk Packets	10-16

Configuring IP Encapsulation DDP Socket to UDP Port Mapping	10-17
Checking Packet Routing Validity	10-18
Enabling and Disabling Routing Updates	10-18
Changing Routing Timers	10-18
Assigning a Proxy Network Number	10-19
Generating Checksum Verification	10-20
Specifying the Time Interval Between AppleTalk ARP Transmissions	10-21
Specifying the AARP Retransmission Count	10-22
AppleTalk MacIP Routing and IP Address Management Service	10-22
AppleTalk Access and Distribution Lists	10-27
AppleTalk Access Control Methods	10-28
Creating AppleTalk Access Lists	10-29
Assigning an Access List to an Interface	10-31
Filtering Networks Received in Updates	10-31
Filtering Networks Sent Out in Updates	10-32
Defining Get-Zone-List Filters	10-33
Permitting Partial Zones	10-34
Requiring Specific Route Zones	10-35
Controlling AppleTalk Names Displayed	10-35
AppleTalk Configuration Examples	10-38
Nonextended AppleTalk Routing Between Two Ethernets	10-39
Configuring Transition Mode	10-40
Nonextended AppleTalk Routing over X.25	10-41
Extended AppleTalk Routing Network	10-42
Extended AppleTalk Routing over HDLC	10-43
Configuring SNMP in AppleTalk Networks	10-43
Configuring IPTalk	10-44
AppleTalk Access List Configuration Examples	10-49
Hiding and Sharing Access to Resources with Access Lists	10-57
Monitoring the AppleTalk Network	10-62
Displaying AppleTalk Access List Specifications	10-62
Displaying the Adjacent Routes	10-62
Displaying the ARP Cache	10-63
Displaying the Fast-Switching Cache	10-63

- Displaying Global AppleTalk Information 10-64
- Displaying AppleTalk Interface Information 10-65
- Displaying MacIP Status 10-66
- Displaying Nearby NBP Services 10-70
- Displaying NBP Services Registered by Cisco Routers 10-70
- Displaying Neighboring Routers 10-71
- Displaying the Network Routing Table 10-73
- Displaying Information About the Sockets 10-75
- Displaying AppleTalk Traffic Information 10-76
- Displaying Zone Information 10-78
- The AppleTalk Ping Command 10-78
 - AppleTalk NBP Ping Interface 10-79
- Debugging the AppleTalk Network 10-82
- AppleTalk Global Configuration Command Summary 10-84
- AppleTalk Interface Subcommand Summary 10-88

Chapter 11 ***Routing CHAOSnet 11-1***

- Cisco's Implementation of CHAOSnet 11-1
- CHAOSnet Addresses 11-1
- Configuring CHAOSnet Routing 11-2
- Monitoring CHAOSnet 11-2
- Debugging CHAOSnet 11-3

Chapter 12 ***Routing DECnet 12-1***

- Cisco's Implementation of DECnet 12-1
- DECnet Phase IV Addresses 12-2
- Configuring DECnet Routing 12-4
 - Enabling DECnet Routing 12-4
 - Assigning the Cost 12-5
 - Specifying the Node Type 12-6
 - Specifying Node Numbers and Area Sizes 12-6
 - Specifying the Maximum Route Cost for Interarea Routing 12-7
 - Specifying the Maximum Route Cost for Intra-area Routing 12-8
 - Configuring Maximum Visits 12-9
 - Configuring Path Selection 12-9
 - Altering DECnet Defaults 12-10
 - Configuring DECnet on Token Ring 12-12

Managing Traffic Using DECnet Access Lists	12-13
Configuring DECnet Access Lists	12-13
Configuring Extended Access Lists	12-14
DECnet Connect Initiate Filtering	12-14
Configuring Access Groups	12-18
Configuring In- and Out-Routing Filters	12-18
DECnet Phase IV-to Phase-V Conversion	12-19
Designing a Network to Support Both Phase IV and Phase V	12-21
DECnet Configuration Examples	12-22
Establishing Routing; Setting Interfaces; Maximum Address Space	12-22
Level 1 and Level 2 Routing; Designated Router	12-22
Phase IV to Phase V Conversion	12-23
The Address Translation Gateway	12-24
ATG Command Syntax	12-24
ATG Configuration Examples	12-24
Limitations of the ATG	12-27
DECnet Monitoring Commands	12-27
Displaying DECnet Status	12-27
Displaying the DECnet Address Mapping Information	12-28
Displaying the DECnet Routing Table	12-28
Displaying DECnet Traffic Statistics	12-29
Debugging DECnet	12-31
DECnet Global Configuration Command Summary	12-32
DECnet Interface Subcommand Summary	12-35

Chapter 13

Routing IP 13-1

Cisco's Implementation of IP	13-1
Configuring IP	13-1
Enabling IP Routing	13-2
Assigning IP Addresses	13-2
Internet Address Notation	13-2
Address Classes and Formats	13-3
Allowable Internet Addresses	13-4
Internet Address Conventions	13-4
Subnetting and Routing	13-5

Setting IP Interface Addresses	13-7
Local and Network Addresses: Address Resolution	13-8
Address Resolution Using ARP	13-8
Address Resolution Using Proxy ARP	13-10
Address Resolution Using Probe	13-10
Reverse Address Resolution Using RARP and BootP	13-11
Broadcasting in the Internet	13-11
Internet Broadcast Addresses	13-12
UDP Broadcasts	13-13
Forwarding Broadcast Packets and Protocols	13-13
Flooding IP Broadcasts	13-15
Limiting Broadcast Storms	13-16
Configuring ICMP and Other IP Services	13-16
Generating Unreachable Messages	13-17
Generating Redirect Messages	13-17
Setting and Adjusting Packet Sizes	13-18
MTU Path Discovery	13-18
Using the Ping Function	13-19
Configuring Internet Header Options	13-19
Configuring IP Host Name-to-Address Conversion	13-20
Configuring IP Access Lists	13-23
Configuring Standard Access Lists	13-24
Configuring Extended Access Lists	13-25
Controlling Line Access	13-27
Controlling Interface Access	13-28
Configuring the IP Security Option (IPSO)	13-28
IPSO Definitions	13-29
Disabling IPSO	13-30
Setting Security Classifications	13-30
Setting a Range of Classifications	13-30
Modifying Security Levels	13-31
Default Values for Minor Keywords	13-33
IPSO Configuration Examples	13-33
Debugging IPSO	13-35
Configuring IP Accounting	13-36
Enabling IP Accounting	13-36

Defining Maximum Entries	13-36
Specifying Account Filters	13-36
Controlling the Number of Transit Records	13-37
Special IP Configurations	13-38
Configuring Source Routing	13-38
IP Processing on a Serial Interface	13-38
Configuring Simplex Ethernet Interfaces	13-39
Enabling Fast Switching	13-40
Enabling IP Autonomous Switching	13-40
Compressing TCP Headers	13-41
IP Configuration Examples	13-42
Configuring Serial Interfaces	13-42
Flooding of IP Broadcasts	13-43
Creating a Network from Separated Subnets	13-43
Customizing ICMP Services	13-44
Helper Addresses	13-44
HP Hosts on a Network Segment	13-45
Establishing IP Domains	13-45
Configuring Access Lists	13-45
Configuring Extended Access Lists	13-46
Configuring SLIP for the Router	13-46
Serial Line Internet Protocol (SLIP)	13-46
Cisco's Implementation of SLIP	13-47
Making SLIP Connections	13-48
Configuring SLIP	13-50
Specifying SLIP Access Lists	13-54
Specifying SLIP Extended BootP Requests	13-55
SLIP Configuration Example	13-57
Maintaining the IP Network	13-57
Removing Dynamic Entries from the ARP Cache	13-57
Removing Entries from the Host-Name-and-Address Cache	13-57
Clearing the Checkpointed Database	13-57
Removing Routes	13-58
Maintaining SLIP	13-58
Monitoring the IP Network	13-58
Displaying the IP Show Commands	13-58

-
- Displaying the ARP Cache 13-59
 - Displaying IP Accounting 13-59
 - Displaying Host Statistics 13-60
 - Displaying the Route Cache 13-61
 - Displaying Interface Statistics 13-62
 - Displaying the Routing Table 13-63
 - Displaying Protocol Traffic Statistics 13-64
 - Monitoring TCP Header Compression 13-65
 - Monitoring SLIP 13-67
 - Displaying the Mapped Internet Address 13-67
 - Displaying the IP ARP Cache 13-67
 - Displaying SLIP Line Status 13-68
 - Displaying the Status of SLIP-Configured Lines 13-68
 - Displaying SLIP BootP Parameters 13-69
 - IP Ping Command 13-70
 - IP Trace Command 13-71
 - How Trace Works 13-72
 - Common Trace Problems 13-72
 - Tracing IP Routes 13-72
 - Debugging the IP Network 13-74
 - IP Global Configuration Command Summary 13-76
 - IP Interface Subcommand Summary 13-80
 - IP and SLIP Line Subcommand Summary 13-84

Chapter 14 *The IP Routing Protocols 14-1*

- Cisco-Supported Routing Protocols 14-1
 - Interior and Exterior Protocols ICMP 14-2
 - Autonomous Systems 14-2
 - Multiple Routing Protocols 14-3
 - Multiple IP Routing Processes 14-3
- Configuration Overview 14-4
 - Configuring the Interior Routing Protocols 14-4
 - Configuring the Exterior Routing Protocols 14-4
- Configuring the IGRP Protocol 14-5
 - Interior, System, and Exterior Routes 14-5
 - Creating the IGRP Routing Process 14-5

Unequal-Cost Load Balancing	14-6
Choosing the Gateway of Last Resort	14-8
IGRP Metric Information	14-8
IGRP Updates	14-8
Configuring the OSPF Routing Protocol	14-8
The OSPF Routing Protocol	14-9
The OSPF Routing Domain and Areas	14-9
OSPF Routing Conventions	14-11
Neighbors and Adjacency	14-13
Cisco's OSPF Implementation	14-14
Steps in Configuring OSPF Routing	14-15
Enabling the OSPF Routing Processes and Defining Areas	14-15
Configuring OSPF Area Parameters	14-18
Configuring OSPF Interface-Specific Parameters	14-19
Configuring OSPF for Nonbroadcast Networks	14-23
Configuring the RIP Protocol	14-25
Creating the RIP Routing Process	14-26
Specifying the List of Networks	14-26
Configuring the Hello Protocol	14-27
Creating the Hello Routing Process	14-28
Specifying the List of Hello Networks	14-28
Configuring the BGP Protocol	14-28
Creating the BGP Routing Process	14-28
Specifying the List of BGP Networks	14-29
Specifying the List of Neighbors	14-29
Filtering BGP Routes	14-32
Specifying BGP Version Number	14-33
Specifying BGP Administrative Distance	14-33
Adjusting the BGP Timers	14-34
Clearing BGP Connections	14-35
BGP and IGP Routing Information	14-35
Backdoor Routes	14-36
BGP Route Selection Rules	14-36
BGP Path Attributes	14-37
Using BGP Without IGP Redistribution	14-37
Configuring the EGP Protocol	14-38

Specifying the Autonomous System Number	14-39
Creating the EGP Routing Process	14-39
Specifying the List of Neighbors	14-39
Specifying the Network to Advertise	14-39
Adjusting Timers	14-41
Configuring Third-Party EGP Support	14-42
Configuring a Backup EGP Router	14-42
Generating an EGP Default Route	14-43
Defining a Core Gateway EGP Process	14-43
Filtering Routing Information	14-45
Filtering Outgoing Information	14-45
Filtering Incoming Information	14-49
Directly Connected Routes	14-52
Overriding Static Routes with Dynamic Protocols	14-54
Default Routes	14-54
Redistributing Routing Information	14-55
Special Routing Configuration Techniques	14-63
Configuring Static Routes	14-63
Enabling and Disabling Split Horizon for IP Networks	14-64
IGRP Metric Adjustments	14-66
Keepalive Timers	14-68
Adjustable Routing Timers	14-69
Gateway Discovery Protocol (GDP)	14-70
ICMP Router Discovery Protocol	14-73
Using IRDP Commands	14-73
Displaying IRDP Values	14-74
IP Routing Protocol Configuration Examples	14-74
Static Routing Redistribution	14-74
RIP and Hello Redistribution	14-75
IGRP Redistribution	14-75
Third-Party EGP Support	14-76
Backup EGP Router	14-76
BGP Route Advertisement and Redistribution	14-77
OSPF Routing and Route Redistribution	14-78
Maintaining IP Routing Operations	14-83
Monitoring IP Routing Operations	14-83

Displaying the BGP Routing Table	14-84
Displaying BGP Neighbors	14-85
Displaying Routes Learned from a Neighbor	14-87
Displaying BGP Paths	14-87
Displaying BGP Summaries	14-88
Displaying EGP Statistics	14-88
Displaying Routing Protocol Parameters and Status	14-89
Displaying OSPF Routing Processes	14-90
Displaying the OSPF Database	14-92
Displaying OSPF Interface Parameters	14-98
Displaying OSPF Neighbor Information	14-99
Displaying the Routing Table	14-99
Displaying Network Masks	14-102
Debugging IP Routing	14-102
Global Configuration Command Summary	14-104
Router Subcommand Summary	14-106
IP Routing Interface Subcommands	14-116

Chapter 15

<i>Switching ISO CLNS</i>	15-1
Cisco's Implementation of ISO CLNS	15-1
Switching ISO CLNS	15-2
ISO Routing Terminology	15-2
Configuring ISO CLNS	15-4
CLNS Addresses	15-4
Addressing Rules	15-6
Multihoming in IS-IS Areas	15-7
Enabling CLNS Routing	15-8
Configuring CLNS for the Router	15-8
Configuring CLNS over X.25	15-9
Configuring ES-IS Parameters	15-11
Specifying Hello Packets	15-11
Configuring Static Configuration of ESs	15-11
Configuring Performance Parameters	15-12
Specifying the MTU Size	15-12
Configuring Checksums	15-13
Enabling Fast Switching	15-13

Setting the Congestion Threshold	15-13
Transmitting Error PDUs	15-14
Redirecting PDUs	15-14
Configuring Parameters for Locally Sourced Packets	15-15
Header Options	15-16
NSAP Shortcut Command	15-16
Static NSAP Address Assignment	15-17
Configuring DECnet Cluster Aliases	15-18
Maintaining a CLNS Network	15-18
Monitoring a CLNS Network	15-19
Displaying General CLNS Information	15-19
Displaying the CLNS Routing Cache	15-20
Displaying CLNS Traffic	15-21
Displaying CLNS Redirect Information	15-22
Displaying Status About Specific Interfaces	15-23
Displaying ES and IS Neighbors	15-24
Displaying CLNS IS Neighbors	15-25
Displaying CLNS ES Neighbors	15-26
ISO CLNS Ping Command	15-27
ISO CLNS Trace Command	15-28
How Trace Works	15-28
Tracing CLNS Routes	15-29
Debugging a CLNS Network	15-30
ISO CLNS Global Configuration Command Summary (Switching)	15-31
ISO CLNS Interface Subcommand Summary (Switching)	15-32

Chapter 16	<i>ISO CLNS Routing Protocols 16-1</i>
	Cisco-Supported Routing Protocols 16-1
	Configuring ISO CLNS Routing 16-2
	Configuring CLNS Static Routing 16-2
	Configuring ISO-IGRP CLNS Dynamic Routing 16-4
	Configuring Network Entity Titles 16-5
	Specifying Router Level Support 16-6
	Configuring ISO-IGRP Metrics 16-6
	Redistributing Routes into an ISO-IGRP Domain 16-6
	Redistributing Static Routes 16-7

Specifying Preferred Routes	16-8
Configuring IS-IS CLNS Dynamic Routing	16-8
Enabling IS-IS Routing	16-9
Configuring Network Entity Titles	16-9
Specifying Router Level Support	16-10
Redistributing Routes into an IS-IS Domain	16-10
Configuring IS-IS for an Interface	16-11
Configuring IS-IS Link State Metrics	16-11
Specifying Designated Router Election	16-12
Specifying Interface Circuit Type	16-12
Configuring IS-IS Password Authentication	16-12
CLNS Static, ISO-IGRP, and IS-IS Routing Examples	16-13
Basic Static Routing	16-13
Static Routing	16-13
Static Intradomain Routing	16-13
Static Interdomain Routing	16-15
Routing Within the Same Area	16-16
Dynamic Routing in More Than One Area	16-16
Dynamic Routing in Overlapping Areas	16-17
Dynamic Interdomain Routing	16-18
IS-IS Routing Configuration	16-19
Monitoring a CLNS Network	16-20
Displaying CLNS Routes	16-20
Displaying the IS-IS Level 1 Routing Table	16-22
Displaying Protocol-Specific Information	16-22
Displaying the IS-IS Database	16-24
Debugging ISO CLNS Dynamic Routing Protocols	16-26
ISO CLNS Global Command Summary (Routing Protocols)	16-26
ISO CLNS Router Subcommand Summary	16-27
ISO CLNS Interface Subcommand Summary (Routing Protocols)	16-28
Chapter 17	
<i>Routing Novell IPX</i>	17-1
Cisco's Implementation of Novell IPX	17-1
Novell Addresses	17-1
Configuring Novell Routing	17-2
Novell Configuration Restrictions	17-2

Enabling Novell Routing	17-2
Novell Encapsulation	17-4
Configuring Static Routes	17-4
Configuring Static SAP	17-5
Setting Maximum Paths	17-6
Setting Novell Update Timers	17-7
Filtering Novell Packets	17-8
Configuring Novell IPX Access Lists	17-8
Configuring Extended Novell Access Lists	17-9
Filtering Outgoing Traffic	17-9
Example of Controlling Traffic with Access Lists	17-10
Filtering Novell Routing Updates	17-12
Establishing Input Filters	17-12
Establishing Output Filters	17-13
Establishing Router Filters	17-13
Building SAP Filters	17-13
Defining Access Lists for SAP Filtering	17-14
Configuring Novell SAP Filters	17-15
Example SAP Input Filter	17-15
Example SAP Output Filter	17-17
Novell Broadcast Helper Facilities	17-18
Defining a Helper List	17-18
Specifying Target Novell Servers	17-19
Using Helper Facilities to Control Broadcasts	17-19
Forwarding to an Address	17-19
Forwarding to All Networks	17-21
Enabling Novell Fast Switching	17-22
Restricting SAP Updates	17-22
SAP Update Delays	17-23
Novell Configuration Example	17-24
Monitoring the Novell IPX Network	17-24
Displaying the Novell Cache Entries	17-24
Displaying Novell Interface Parameters	17-25
Displaying the Novell Routing Table	17-25
Displaying Novell Servers	17-25
Displaying Novell Traffic	17-26

The Novell Ping Command 17-27
Debugging the Novell IPX Network 17-27
Novell IPX Global Configuration Command Summary 17-28
Novell IPX Interface Subcommand Summary 17-29

Chapter 18 ***Routing PUP 18-1***
Cisco's Implementation of PUP 18-1
PUP Addresses 18-1
Configuring PUP Routing 18-1
PUP Mapped to IP 18-2
PUP Miscellaneous Services 18-3
The PUP Ping Command 18-3
Monitoring PUP 18-3
Debugging PUP 18-4

Chapter 19 ***Routing VINES 19-1***
Cisco's Implementation of VINES 19-1
Configuring VINES 19-2
VINES Addressing 19-2
The VINES Routing Table Protocol 19-3
Configuring VINES Routing 19-3
Configuring Address Resolution 19-5
Configuring VINES Encapsulation 19-6
Configuring Name-to-Address Mappings 19-7
Configuring VINES Access Lists 19-8
VINES Configuration Examples 19-9
 Propagating Broadcasts 19-9
 Filtering Packets 19-10
Maintaining the VINES Network 19-10
Monitoring the VINES Network 19-11
 Displaying the VINES Name Table 19-11
 Displaying VINES Interface Settings 19-11
 Displaying the VINES Neighbor Table 19-12
 Displaying the VINES Routing Table 19-12
 Displaying VINES Traffic 19-13
The VINES Ping Command 19-13
The VINES Trace Command 19-13

Debugging the VINES Network 19-14
VINES Global Configuration Command Summary 19-15
VINES Interface Subcommand Summary 19-16

Chapter 20

Routing XNS 20-1

Cisco's Implementation of XNS 20-1
 Ethernet and Token Ring 20-2
XNS Addresses 20-2
 Network and Host Numbers 20-2
 Socket Numbers 20-3
Configuring XNS 20-3
Enabling XNS Routing 20-3
 Configuring Static Routing 20-4
 Managing Throughput 20-4
 Configuring XNS over Token Ring 20-6
Configuring Ungermann-Bass Net/One XNS 20-7
 Ungermann-Bass Net/One on a Cisco Router 20-7
 Configuring Ungermann-Bass Net/One Routing 20-8
 Ungermann-Bass Routing Protocol and Interactions with RIP 20-9
Handling XNS Broadcasts, Flooding, and Helpering 20-10
 Overview of XNS Broadcasts 20-10
 Flooding and Helpering 20-11
 Notes and Tips on Configuring XNS Helpering 20-13
Configuring XNS Access Lists and Filters 20-15
 Configuring XNS Access Lists 20-15
 Configuring Extended Access Lists 20-16
 Filtering Outgoing Packets 20-17
 Filtering XNS Routing Updates 20-17
XNS Configuration Examples 20-19
 Creating a Routing Process 20-19
 Setting Timers 20-20
 Configuring for Multiprotocol Routing 20-20
 Configuring for Ungermann-Bass Routing 20-21
 3Com Access List 20-21
Monitoring an XNS Network 20-22
 Displaying Cache Entries 20-22

Displaying Interface Parameters	20-22
Displaying the Routing Table	20-23
Displaying Traffic Statistics	20-23
Debugging an XNS Network	20-24
XNS Global Configuration Command Summary	20-25
XNS Interface Subcommand Summary	20-26

Part Four ***Configuring Bridging***

Chapter 21 ***Configuring Transparent Bridging 21-1***

Bridging Overview	21-1
Cisco's Implementation of Transparent Bridging	21-4
Source-Route Transparent (SRT) Bridging	21-5
Choosing the OUI for Ethernet Type II Frames That Transit Token Rings	21-7
Configuring Transparent Bridging	21-8
Defining the Spanning-Tree Protocol	21-8
Establishing Multiple Spanning-Tree Domains	21-9
Bridging or Routing IP	21-10
Assigning Each Interface to a Spanning-Tree Group	21-10
Adjusting Spanning-Tree Parameters	21-13
Electing the Root Bridge	21-13
Adjusting the Interval Between Hello BPDUs	21-13
Defining the Maximum Idle Interval	21-14
Assigning Path Costs	21-15
Setting an Interface Priority	21-16
Establishing Administrative Filtering	21-16
Administrative Filtering by MAC Layer Address	21-16
Administrative Filtering by Protocol Type	21-19
Administrative Filtering by Vendor Code	21-23
Administrative Filtering of LAT Service Announcements	21-25
Extended Access Lists for Transparent Bridging	21-28
Special Bridging Configurations	21-30
Establishing Load Balancing	21-30
Configuring X.25 Bridging	21-31

Configuring Frame-Relay Bridging	21-33
Configuring LAT Compression	21-35
Transparent Bridging Configuration Examples	21-36
Configuring Ethernet Bridging	21-36
Configuring Source-Route Transparent (SRT) Bridging	21-37
Maintaining the Transparent Bridge	21-39
Monitoring the Transparent Bridge	21-39
Viewing Entries in the Forwarding Database	21-39
Displaying the Known Spanning-Tree Topology	21-40
Debugging the Transparent Bridge	21-41
Transparent Bridging Global Configuration Command Summary	21-42
Transparent Bridging Interface Subcommand Summary	21-44

Part Five *Configuring for IBM Networks*

Chapter 22 *Configuring Source-Route Bridging 22-1*

Source-Route Bridging Overview	22-1
Cisco's Implementation of Source-Route Bridging	22-2
Configuring Source-Route Bridging	22-3
Enabling and Disabling the Source-Route Fast-Switching Cache	22-4
Enabling and Disabling the Source-Route Autonomous-Switching Cache	22-4
Determining the Source-Route Information Field	22-5
Enabling Use of the RIF	22-7
Determining the RIF Timeout Interval	22-8
Configuring a Static RIF Entry	22-9
Configuring Local Source-Route Bridging	22-11
Enabling Local Source-Route Bridging	22-11
Configuring Explorer Packets	22-12
Configuring Ring Groups and Multiport Source-Route Bridges	22-13
Configuring Remote Source-Route Bridging	22-15
Configuring Remote Source-Route Bridging with TCP	22-15
Configuring Remote Source-Route Bridging with Fast Sequenced Transport	22-19

Performance Considerations When Using FST in a Redundant Network Topology	22-22
Configuring Remote Source-Route Bridging with Direct Encapsulation	22-22
Configuring the Largest Frame	22-26
Configuring a Proxy Explorer	22-26
The Local Acknowledgment Function for LLC2	22-27
Notes on the Use of Local Acknowledgment for LLC2	22-29
Configuring Local Acknowledgment for Token Ring Interfaces	22-30
Setting Up Local Acknowledgment for LLC2	22-30
Displaying Local Acknowledgment Statistics	22-32
Debugging Local Acknowledgment	22-33
SNA Local LU Address Prioritization	22-33
Assigning Priority by SNA Local LU Address	22-34
Configuring LOCADDR Priority	22-35
Bridging Between Source-Route Bridge Groups and Transparent Bridge Groups (SR/TLB)	22-36
Overview of SR/TLB	22-37
Configuring SR/TLB	22-38
Further Notes on the Use of SR/TLB	22-40
Enabling Translation Compatibility with IBM 8209 Bridges	22-42
Using SR/TLB in Many IBM LLC2 Environments (0x80d5 Processing)	22-43
Frame Conversions Between Source-Route and Transparent Bridging	22-44
Configuring Administrative Filtering of Source-Routed Frames	22-45
Administrative Filtering by Protocol Type	22-45
Administrative Filtering by Vendor Code or Address	22-49
Configuring NetBIOS Name Caching	22-50
Understanding NetBIOS Name Caching	22-50
Enabling NetBIOS Name Caching	22-51
Configuring "Dead Time" Periods	22-52
Creating Static Entries	22-52
Enabling the NetBIOS Name Cache	22-53
Maintaining the NetBIOS Name Cache	22-54
Debugging the NetBIOS Name Cache	22-54

Configuring NetBIOS Access Control Filtering	22-55
Configuring Access Control Using Station Names	22-55
Configuring Access Control Using a Byte Offset	22-58
Combining Administrative Filters with Boolean Access Expressions	22-61
Configuring Access Expressions	22-62
Interoperability Issues	22-68
PC/3270 Emulation Software	22-68
TI MAC Firmware	22-68
Spurious Frame-Copied Errors	22-69
Cisco's Implementation of LAN Network Manager	22-70
Configuring the Router for LAN Network Manager Support	22-73
Configuring Servers	22-74
Changing Reporting Timers	22-74
Configuring LAN Network Manager	22-75
Monitoring LNM Support	22-78
Displaying Bridge Information	22-78
Displaying LNM Configuration Information	22-78
Displaying Token Ring Information	22-79
Displaying Station-Specific Information	22-81
Debugging LNM Support	22-83
Source-Route Bridging Configuration Examples	22-83
Basic Token Ring Configuration	22-83
Basic Token Ring Source Bridge Configuration	22-84
Source Bridge Only Configuration	22-84
Other Protocols Routed at the Same Time	22-85
Remote Source-Route Bridge with Simple Reliability	22-85
Remote Source-Route Bridge—Complex Configuration, with Local Acknowledgment	22-87
Maintaining the Source-Route Bridge	22-90
Monitoring the Source-Route Bridge	22-90
Displaying the RIF Cache	22-90
Displaying the Current Bridge Configuration	22-91
Displaying Information About the Token Ring Interface	22-93
Displaying Token Ring Interface Statistics	22-93
Debugging the Source-Route Bridge	22-93

	Source-Route Bridge Global Configuration Command Summary	22-96
	Source-Route Bridge Interface Subcommand Summary	22-100
Chapter 23	<i>Configuring Serial Tunneling and SDLC Transport</i>	23-1
	The Cisco Serial Tunnel (STUN) and Serial Transport Functions	23-1
	Configuring STUN	23-4
	Configuring SDLC Serial Tunneling	23-4
	Configuring Non-SDLC Serial Tunneling	23-5
	Notes and Tips About Configuring STUN	23-5
	Enabling Serial Tunneling	23-6
	Defining the STUN Protocol	23-6
	Choosing the SDLC Serial Tunneling Protocol	23-7
	Choosing the Basic Serial Tunneling Protocol	23-7
	Configuring STUN on the Interface	23-7
	Placing the Interface in a STUN Group	23-8
	Defining How Frames Will Be Forwarded	23-8
	STUN Class of Service	23-10
	Configuring STUN Serial Link Address Priorities	23-10
	SDLC Local Acknowledgment	23-15
	Configuring Local Acknowledgment for STUN Interfaces	23-17
	Setting Up Local Acknowledgment	23-18
	Displaying Local Acknowledgment Statistics	23-19
	Debugging Local Acknowledgment	23-20
	SNA Local LU Address Prioritization	23-21
	Assigning Priority by SNA Local LU Address	23-22
	Assigning LOCADDR Priority Groups	23-22
	Using STUN with Multiple-Link Transmission Groups	23-24
	STUN Configuration Examples	23-25
	Expanding the IBM Network Capability Using Cisco Routers	23-25
	Extended IBM Network	23-27
	Prioritizing STUN Traffic	23-31
	Serial Link Address Prioritization	23-31
	Configuring Redundant Links	23-32
	Using Proxy Polling in SDLC Environments	23-35
	Configuring Proxy Polling	23-37
	Enabling Proxy Polling	23-37

	Configuring a Proxy Poll Interval	23-38
	Configuring a Primary Side Pass-Through Interval	23-38
	Defining Your Own STUN Protocols	23-39
	Monitoring STUN	23-41
	Displaying the Current Status of STUN	23-41
	Displaying the Proxy States	23-42
	Debugging STUN	23-43
	STUN Global Configuration Command Summary	23-43
	STUN Interface Subcommand Summary	23-45
Chapter 24	<i>LLC2 and SDLC Link-Level Support 24-1</i>	
	Overview and Comparison of SDLC and LLC2	24-2
	Role of LLC2 and SDLC Stations	24-2
	Addressing LLC2 and SDLC Stations	24-2
	Cisco's Implementation of LLC2	24-3
	Configuring LLC2	24-3
	Monitoring LLC2	24-8
	Debugging LLC2	24-10
	Cisco's Implementation of SDLC	24-11
	Configuring SDLC	24-11
	Monitoring SDLC	24-17
	Debugging SDLC	24-19
	LLC2 Interface Subcommand Summary	24-19
	SDLC Interface Subcommand Summary	24-21
Chapter 25	<i>SDLLC: SDLC to LLC2 Media Translation 25-1</i>	
	The Cisco SDLLC Function	25-1
	Performance Tuning	25-2
	Configuring SDLLC	25-3
	Configuring SDLLC on Serial Interfaces	25-7
	Support for Multipoint	25-8
	Support for Ethernet SDLLC	25-9
	Specifying Optional SDLLC Serial Interface Commands	25-10
	Specifying the Largest Information Frame Size	25-10
	Specifying the Connection Type	25-11
	SDLLC Local Acknowledgment	25-12
	Configuring Local Acknowledgment	25-13

Router Configuration Parameter Considerations	25-13
Complex SDLLC Configuration Example Using Local Acknowledgment	25-14
Monitoring SDLLC	25-18
Displaying Local Acknowledgment Statistics	25-18
Debugging SDLLC	25-21
SDLLC Global Command Summary	25-21
SDLLC Interface Subcommand Summary	25-21

Appendixes

<i>A</i>	<i>Frame Conversions</i>	<i>A-1</i>
<i>B</i>	<i>Access List Summary</i>	<i>B-1</i>
<i>C</i>	<i>Ethernet Type Codes</i>	<i>C-1</i>
<i>D</i>	<i>Pattern Matching</i>	<i>D-1</i>
<i>E</i>	<i>ASCII Character Set</i>	<i>E-1</i>
<i>F</i>	<i>References and Recommended Reading</i>	<i>F-1</i>
<i>G</i>	<i>X.25 Diagnostic Codes</i>	<i>G-1</i>

Index

List of Figures

- Figure 6-1* Dial-on-Demand Interconnection 6-62
- Figure 6-2* Hub-and-Spoke Environment Using Dial-on-Demand 6-67
- Figure 6-3* Dial-on-Demand Routing Configuration 6-82
- Figure 7-1* HSSI Loopback Test 7-14
- Figure 7-2* HSSI External Loopback Request 7-16
- Figure 7-3* UltraNet Loopback Test 7-17
- Figure 8-1* Transporting LAN Protocols Across an X.25 PDN 8-7
- Figure 8-2* Routing X.25 Traffic Through a LAN 8-7
- Figure 8-3* Communicating via Routers Through an X.25 Network 8-8
- Figure 8-4* Parallel Serial Lines to X.25 Network 8-11
- Figure 8-5* Establishing a PVC through an X.25 Network 8-13
- Figure 8-6* X.121 Address Translation Scheme 8-21
- Figure 8-7* Example Network Topology for Switching CMNS over a PDN 8-48
- Figure 8-8* Example Network Topology for Switching CMNS over a Leased Line 8-50
- Figure 8-9* Frame Relay Physical Connection 8-55
- Figure 9-1* Apollo Domain Addresses 9-2
- Figure 10-1* AppleTalk Entities 10-2
- Figure 10-2* AppleTalk and the OSI Reference Model 10-3
- Figure 10-3* Sample AppleTalk Routing Table 10-9
- Figure 10-4* Sample illustration of Inter•Poll Output 10-11
- Figure 10-5* Example Network Topology 10-20
- Figure 10-6* Nonextended AppleTalk Routing Between Two Ethernet Networks 10-39
- Figure 10-7* Routing Between Seed Routers 10-40
- Figure 10-8* Transition Mode Topology and Configuration 10-41
- Figure 10-9* IPTalk Configuration Example 10-48
- Figure 10-10* Example Topology of Partially Obscured Zone 10-56
- Figure 10-11* Controlling Access to Common AppleTalk Network 10-58

Figure 10-12 Controlling Resource Access Among Multiple AppleTalk Zones 10-59

Figure 12-1 DECnet Nodes and Areas 12-2

Figure 12-2 DECnet Cost Values 12-6

Figure 12-3 ATG Configuration Example 12-25

Figure 13-1 Class A Internet Address Format 13-3

Figure 13-2 Class B Internet Address Format 13-3

Figure 13-3 Class C Internet Address Format 13-4

Figure 13-4 A Class B Address with a 5-Bit Subnet Field 13-5

Figure 13-5 IP Flooded Broadcast 13-12

Figure 13-6 MTU Path Discovery 13-19

Figure 13-7 IPSO Security Levels 13-34

Figure 13-8 Simplex Ethernet Connections 13-39

Figure 13-9 Creating a Network from Separated Subnets 13-43

Figure 13-10 IP Helper Addresses 13-45

Figure 13-11 Sample SLIP Configuration Using the Auxiliary Port 13-48

Figure 14-1 Autonomous System 12 Containing Four Routers 14-3

Figure 14-2 Interior, System, and Exterior Routes 14-5

Figure 14-3 Determining IGRP Path Feasibility 14-7

Figure 14-4 Overview of OSPF Router Types and Relationships 14-11

Figure 14-5 Hop Count in RIP 14-26

Figure 14-6 The Hello Protocol 14-27

Figure 14-7 BGP and IGP Routing 14-35

Figure 14-8 Illustration of Synchronization 14-38

Figure 14-9 EGP and Interior and Exterior Routers 14-38

Figure 14-10 Router in AS 164 Peers with Router in AS 109 14-41

Figure 14-11 Core EGP Third-Party Update Configuration Example 14-45

Figure 14-12 Filtering IGRP Updates 14-46

Figure 14-13 Filtering RIP Updates 14-48

Figure 14-14 Assigning Metrics for Redistribution 14-59

Figure 14-15 Disabled Split Horizon Example for Frame Relay Network 14-65

Figure 14-16 GDP Report Message Packet Format 14-71

Figure 14-17 Sample OSPF Autonomous System Network Map 14-79

Figure 14-18 Interface and Area Specifications for OSPF Example Configuration 14-81

Figure 15-1 ISO CLNS Areas 15-3

Figure 15-2 ISO-IGRP NSAP Addressing Structure 15-5

Figure 15-3 IS-IS NSAP Addressing Structure 15-6

Figure 16-1 CLNS X.25 Intradomain Routing 16-14

Figure 16-2 CLNS Interdomain Static Routing 16-15

Figure 16-3 CLNS Dynamic Routing Within a Single Area 16-16

Figure 16-4 CLNS Dynamic Routing Within Two Areas 16-17

Figure 16-5 CLNS Dynamic Routing Within Overlapping Areas 16-17

Figure 16-6 CLNS Interdomain Dynamic Routing 16-18

Figure 17-1 Multiple Novell Servers Requiring Access Control 17-10

Figure 17-2 SAP Input Filter 17-16

Figure 17-3 SAP Output Filter 17-17

Figure 17-4 Novell Clients Requiring Server Access Through a Cisco Router 17-20

Figure 17-5 Type 10 Broadcast Flooding 17-21

Figure 19-1 Banyan VINES Network-Level Addresses 19-2

Figure 19-2 VINES Client/Server Configuration 19-9

Figure 20-1 Helper Addresses 20-14

Figure 21-1 OSI Model and IEEE 802 Layers 21-2

Figure 21-2 IEEE 802 Frame Formats 21-3

Figure 21-3 Example of Basic Bridging 21-11

Figure 21-4 Network Demonstrating Output Address List Filtering 21-24

Figure 21-5 X.25 Bridging Example 21-32

Figure 21-6 Frame-Relay Bridging Example 21-33

Figure 21-7 Ethernet Bridging Configuration Example 21-36

Figure 21-8 Example Network Configuration 21-38

Figure 22-1 IEEE 802.5 Token Ring Frame Format 22-2

Figure 22-2 Basic RIF Format 22-5

Figure 22-3 RIF Routing Control Format 22-6

Figure 22-4 Routing Descriptor Format 22-6

Figure 22-5 Assigning a RIF to a Source-Route Bridge 22-10

Figure 22-6 Assigning a RIF to a Two-Hop Path 22-10

Figure 22-7 Dual-Port Source-Route Bridge Configuration 22-12

Figure 22-8 Four-Port Source-Route Bridge 22-14

Figure 22-9 Remote Source-Route Bridging Using TCP and FST as a Transport 22-18

Figure 22-10 Remote Source-Route Bridging Using Direct Encapsulation as a Transport 22-24

Figure 22-11 Remote Source-Route Bridge Using All Types of Transport Methods 22-25

Figure 22-12 LLC2 Session Without Local Acknowledgment 22-27

Figure 22-13 LLC2 Session with Local Acknowledgment 22-28

Figure 22-14 SNA Local Address Prioritization 22-34

Figure 22-15 RSRB Configuration Example 22-35

Figure 22-16 Example of a Simple SR/TLB Topology 22-37

Figure 22-17 Example of a Simple SR/TLB Configuration 22-39

Figure 22-18 Example of a Bit-Swapped Address 22-41

Figure 22-19 Specifying a Static Entry 22-53

Figure 22-20 Access Expression Example 22-62

Figure 22-21 Network Configuration Using NetBIOS Access Filters 22-67

Figure 22-22 LNM Linking to a Source-Route Bridge on Each Local Ring 22-70

Figure 22-23 LAN Network Manager Monitoring and Translating 22-71

Figure 22-24 Router with Two Token Rings Configured as a Local Source-Route Bridge 22-76

Figure 22-25 Cisco Router with Three Token Rings Configured as a Multiport Bridge 22-77

Figure 22-26 Remote Source-Route Bridge—Simple Reliability 22-86

Figure 22-27 Remote Source-Route Bridge with Local Acknowledgment—Complex Configuration 22-87

Figure 23-1 IBM Network Configuration With and Without SDLC Transport 23-3

Figure 23-2 STUN Simple Serial Transport 23-11

Figure 23-3 STUN TCP/IP Encapsulation 23-13

Figure 23-4 SDLC Session Without Local Acknowledgment 23-16

Figure 23-5 SDLC Session with Local Acknowledgment 23-16

Figure 23-6 SNA Local Address Prioritization 23-21

Figure 23-7 IBM Network Without Router 23-25

Figure 23-8 IBM Network with Routers and SDLC Links 23-26

Figure 23-9 Extended IBM Network with Routers and SDLC Links 23-28

Figure 23-10 IBM Network with Multiple Groups of Controllers 23-29

Figure 23-11 Serial Link Address Prioritization 23-32

-
- Figure 23-12* Redundant Links in an IBM Network 23-32
- Figure 23-13* Redundant Links in an IBM Network Using IP Addresses for Reference 23-34
- Figure 23-14* Typical IBM SDLC Primary and Secondary Node Configuration 23-36
- Figure 23-15* IBM SDLC Primary and Secondary Nodes with Proxy Polling Feature 23-36
- Figure 23-16* SDLC Frame Format 23-39
- Figure 24-1* Two SDLC Secondary Stations Attached to a Single Serial Interface Through an MSD 24-12
- Figure 25-1* Simple SDLLC Example 25-4
- Figure 25-2* Complex SDLLC Example 25-6
- Figure 25-3* Complex SDLLC Example with Multipoint 25-9
- Figure 25-4* Ethernet SDLLC Support 25-10
- Figure 25-5* SDLLC Local Acknowledgment 25-13
- Figure A-1* Frame Conversion Between IEEE 802.3 and Token Ring A-2
- Figure A-2* Frame Conversion Between Ethernet Type II and Token Ring SNAP A-2
- Figure A-3* Frame Conversion Between Ethernet Type II “0x80D5” Format and Token Ring A-2

List of Tables

- Table 2-1* Configuration Edit Keys 2-11
- Table 3-1* Special Commands in the Cisco Systems Telnet Implementation 3-5
- Table 4-1* Show Flash Field Descriptions 4-12
- Table 4-2* Show Flash All Field Descriptions 4-13
- Table 4-3* Logging Message Keywords and Levels 4-39
- Table 5-1* Show Buffers Field Descriptions 5-3
- Table 5-2* Show Memory Field Descriptions 5-4
- Table 5-3* Characteristics of Each Block of Memory 5-4
- Table 5-4* Show Processes Field Descriptions 5-5
- Table 5-5* Show Processes Memory Field Descriptions 5-6
- Table 5-6* Show Logging Field Descriptions 5-7
- Table 6-1* Show Controllers Field Descriptions 6-6
- Table 6-2* Per-Packet Counted Protocols 6-7
- Table 6-3* Show Interfaces Serial Field Descriptions 6-12
- Table 6-4* Show Interfaces ISDN BRI Field Descriptions 6-16
- Table 6-5* Show Interfaces Ethernet Field Descriptions 6-22
- Table 6-6* Show Interfaces Token Ring Field Descriptions 6-29
- Table 6-7* Debug Token Ring Messages 6-32
- Table 6-8* Show Interfaces FDDI Field Descriptions 6-35
- Table 6-9* FDDI Physical Type Bit Specifications 6-43
- Table 6-10* FDDI Link Confidence Test Duration Bit Specification 6-43
- Table 6-11* Show Interfaces HSSI Field Descriptions 6-48
- Table 6-12* Show Interfaces UltraNet Field Descriptions 6-53
- Table 6-13* CCITT V.25 bis Options 6-66
- Table 7-1* Common TCP Services and Their Port Numbers 7-6
- Table 7-2* Common UDP Services and Their Port Numbers 7-6
- Table 7-3* Default Media MTU Values 7-13

<i>Table 8-1</i>	LAPB Parameters	8-3
<i>Table 8-2</i>	Protocols and Initial Byte of Call User Data	8-14
<i>Table 8-3</i>	Show X25 Map Field Descriptions	8-15
<i>Table 8-4</i>	Pattern Matching	8-20
<i>Table 8-5</i>	Character Matching	8-20
<i>Table 8-6</i>	Range Limit Keywords for the Switched Virtual Circuit Ranges	8-25
<i>Table 8-7</i>	Retransmission Timer Keywords and Defaults	8-28
<i>Table 8-8</i>	DDN Internet/X.121 Address Conventions	8-40
<i>Table 8-9</i>	Protocol Families and Types of Multicasts Needed	8-72
<i>Table 9-1</i>	Show Apollo Traffic Field Descriptions	9-8
<i>Table 10-1</i>	Examples of AppleTalk Addresses	10-7
<i>Table 10-2</i>	Test Condition #1: Routing Tuple of 55	10-50
<i>Table 10-3</i>	Test Condition #2: Testing Routing Tuple of 55-55	10-51
<i>Table 10-4</i>	Test Condition #3: Testing Routing Tuple of 55-60	10-51
<i>Table 10-5</i>	GZL Filter Example Access List Rules	10-54
<i>Table 10-6</i>	Initial Zone-Network Association Table	10-54
<i>Table 10-7</i>	Zone-Network Association Table After Access List Applied to Network	10-55
<i>Table 10-8</i>	Zone-Network Association Table After Distribution List Test	10-55
<i>Table 10-9</i>	Zone and Interface Associations for Partial Zone Advertisement Example	10-56
<i>Table 10-10</i>	Partial-zone Advertisement Control on Network 30	10-57
<i>Table 10-11</i>	Show IP Arp Field Displays	10-63
<i>Table 10-12</i>	MacIP State Table	10-68
<i>Table 10-13</i>	Show Apple Traffic Field Descriptions	10-76
<i>Table 10-14</i>	AppleTalk Ping Characters	10-79
<i>Table 12-1</i>	A Packet Exchange Between Nodes A and D	12-26
<i>Table 13-1</i>	Reserved and Available Internet Addresses	13-4
<i>Table 13-2</i>	Subnet Masks	13-6
<i>Table 13-3</i>	Configuration Register Settings for Broadcast Address Destination	13-13
<i>Table 13-4</i>	IPSO Level Keywords and Bit Patterns	13-29
<i>Table 13-5</i>	IPSO Authority Keywords and Bit Patterns	13-30
<i>Table 13-6</i>	Default Security Keyword Values	13-33
<i>Table 13-7</i>	Security Actions	13-35

<i>Table 13-8</i>	Show IP ARP Field Displays	13-59
<i>Table 13-9</i>	Show IP Interface Field Descriptions	13-63
<i>Table 13-10</i>	Show IP TCP Header Compression	13-66
<i>Table 13-11</i>	Show IP ARP Display Field Descriptions	13-68
<i>Table 13-12</i>	SLIP Statistics Display Field Descriptions	13-69
<i>Table 13-13</i>	Ping Test Characters	13-70
<i>Table 13-14</i>	Trace Test Characters	13-73
<i>Table 14-1</i>	Default Administrative Distances	14-52
<i>Table 14-2</i>	RIP and Hello Metric Transformations	14-56
<i>Table 14-3</i>	Default Bandwidth Values by Media Type	14-67
<i>Table 14-4</i>	Show IP EGP Field Descriptions	14-89
<i>Table 15-1</i>	Ping Test Characters	15-27
<i>Table 15-2</i>	Trace Test Characters	15-29
<i>Table 16-1</i>	Sample Routing Table Entries	16-3
<i>Table 16-2</i>	Hierarchical Routing Examples	16-3
<i>Table 17-1</i>	Sample Novell SAP Services	17-14
<i>Table 19-1</i>	Interfaces and Example Delay Metric Values	19-4
<i>Table 19-2</i>	Vines Address Assignment Method	19-5
<i>Table 19-3</i>	Trace Test Characters	19-14
<i>Table 20-1</i>	XNS Traffic Statistics Field Descriptions	20-24
<i>Table 21-1</i>	Bridge OUI Codes	21-7
<i>Table 21-2</i>	Media and Path Cost Values	21-15
<i>Table 21-3</i>	Optional MAC Address Filtering Keywords	21-17
<i>Table 21-4</i>	Forwarding Database Display Field Descriptions	21-40
<i>Table 22-1</i>	Show Local-Ack Field Descriptions	22-32
<i>Table 22-2</i>	Common RSRB Services and Their Port Numbers	22-35
<i>Table 22-3</i>	Source-Bridge Command Field Descriptions	22-40
<i>Table 22-4</i>	Station Name Pattern Matching Characters	22-56
<i>Table 22-5</i>	Access Expression Terms	22-64
<i>Table 22-6</i>	Boolean Operators for Access Expression Terms	22-64
<i>Table 22-7</i>	Show LNM Bridge Field Descriptions	22-78
<i>Table 22-8</i>	Show LNM Configuration Field Descriptions	22-79

Table 22-9 Show LNM Interface Field Descriptions 22-79

Table 22-10 Show LNM Station Field Descriptions 22-82

Table 22-11 RIF Cache Display Field Description 22-90

Table 22-12 Current Bridge Configuration Field Descriptions 22-91

Table 23-1 STUN Priority Port Numbers 23-13

Table 23-2 Show Stun Field Descriptions 23-19

Table 23-3 Debug SDLC Local-Ack Debugging Levels 23-20

Table 23-4 STUN Priority Port Numbers 23-22

Table 23-5 Address Length Limitations 23-40

Table 23-6 Allowable Schema Formats 23-40

Table 23-7 STUN Status Display Field Descriptions 23-41

Table 23-8 Node States 23-42

Table 23-9 Allowable Schema Formats 23-45

Table 24-1 LLC2 Parameters 24-4

Table 24-2 Show LLC2 Field Descriptions 24-8

Table 24-3 SDLC Parameters 24-13

Table 24-4 Show Interfaces Fields and Descriptions 24-18

Table 24-5 SDLC Secondary Descriptions 24-18

Table 25-1 Additional LLC2 Parameters 25-2

Table 25-2 Other SDLC Parameters 25-3

Table 25-3 Show Local-Ack Field Descriptions 25-19

Table 25-4 SDLLC Parameters 25-20

Table B-1 Summary of Numerical Ranges B-6

Table D-1 Special Symbols Used as Atoms D-3

Table D-2 Special Symbols Used with Pieces D-4

Table E-1 ASCII-to-Decimal Translation Table E-1

Table G-1 Annex E International Problem Diagnostic Code Differences G-1

Table G-2 X.25 Cause and Diagnostic Field Descriptions G-2

Router Products Configuration and Reference

Part 3

**Configuring
Routing**



Chapter 9

Routing Apollo Domain

9

Cisco's Implementation of Apollo Domain 9-1

Apollo Domain Addresses 9-2

Configuring Apollo Domain Routing 9-2

Enabling Apollo Domain Routing 9-3

Assigning the Apollo Domain Network Numbers 9-3

Configuring Static Routes 9-3

Configuring Maximum Paths 9-4

Setting Apollo Update Timers 9-4

Configuring Apollo Domain Access Lists 9-5

Specifying Apollo Domain Access Lists 9-5

Defining Access Groups 9-6

Monitoring the Apollo Domain Network 9-7

Displaying Apollo Interface Parameters 9-7

Displaying Apollo Routes 9-7

Displaying Apollo Traffic Statistics 9-8

Displaying the Apollo ARP Table 9-8

Debugging the Apollo Domain Network 9-9

Apollo Domain Global Configuration Command Summary 9-9

Apollo Domain Interface Subcommand Summary 9-10

Chapter 9

Routing Apollo Domain

9

This chapter describes how to configure Cisco Systems' 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.

Cisco's Implementation of Apollo Domain

The Apollo Domain routing protocol is the native-mode networking protocol for Apollo workstations. The Cisco routing software implementation supports packet forwarding and routing for the Apollo Domain network protocols on Ethernet, FDDI and serial interfaces using HDLC or X.25 encapsulation. Direct attachment to the 12-megabit Domain Token Ring is not supported.

The following restrictions apply to the Cisco implementation:

- If bridging is enabled on an Ethernet for which Apollo Domain routing is also enabled, special care must be taken. You must specify an Ethernet type code access list that filters out datagrams with the Apollo Domain type code (hexadecimal 8019). This restriction applies to MCI cards running microcode Version 1.5 or earlier.
- An IP address must be set on all media that use the ARP protocol (Ethernet and FDDI, for example). This is because Domain ARP uses the same Ethernet type value as IP ARP.
- The Cisco 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 Cisco routers. Cisco routers do not support the **rtchk** and **lcnod** 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 9-1.

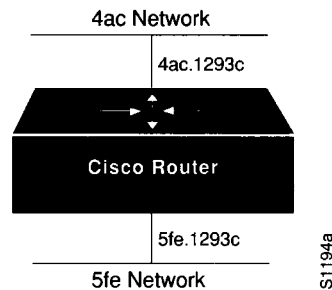


Figure 9-1 Apollo Domain Addresses

Configuring Apollo Domain Routing

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

- Step 1:** Use the global configuration command **apollo routing** to enable routing.
- Step 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 Ethernet 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 ethernet 0  
apollo network 5f  
interface ethernet 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 ethernet 1
  apollo update-time 25
```

Note: Only use this command in an all-Cisco environment, 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 first 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 first 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, l 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, Ethernet1
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 9-1 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 9-1 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 Cisco 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.

Chapter 10

Routing AppleTalk

10

Cisco's Implementation of AppleTalk 10-1

- Extended (Phase II) Versus Nonextended (Phase I) AppleTalk 10-4
- Nonextended AppleTalk Addressing 10-5
- AppleTalk Zones 10-5
- Name Binding Protocol (NBP) 10-6
- Zone Information Protocol (ZIP) 10-7
- Dynamic Configuration (Discovery Mode) 10-7
- Extended AppleTalk Addressing 10-9
- AppleTalk Name Registration 10-10
- AppleTalk Responder Support 10-10

Configuring AppleTalk Routing 10-11

- Configuration Overview 10-11
- Configuration Guidelines (Compatibility Rules) 10-12
- Enabling AppleTalk Routing 10-13
- Assigning Nonextended (Phase I) AppleTalk Address 10-13
- Assigning a Cable Range for Extended AppleTalk (Phase II) 10-13
- Assigning a Zone Name 10-14
- Setting and Resetting Discovery Mode 10-15
 - Using Discovery Mode 10-16
- Configuring IP Encapsulation of AppleTalk Packets 10-16
- Configuring IP Encapsulation DDP Socket to UDP Port Mapping 10-17
- Checking Packet Routing Validity 10-18
- Enabling and Disabling Routing Updates 10-18
- Changing Routing Timers 10-18
- Assigning a Proxy Network Number 10-19
- Generating Checksum Verification 10-20
- Specifying the Time Interval Between AppleTalk ARP Transmissions 10-21
- Specifying the AARP Retransmission Count 10-22
- AppleTalk MacIP Routing and IP Address Management Service 10-22
 - Exceptions to Draft RFC 10-23
 - Configuring MacIP 10-24
 - Enabling MacIP Servers 10-24
 - Specifying Addresses for Dynamic MacIP Clients 10-25
 - Specifying Addresses for Static MacIP Clients 10-26
 - MacIP Configuration and Address Assignment Considerations 10-27

AppleTalk Access and Distribution Lists 10-27

- AppleTalk Access Control Methods 10-28
 - Zone-Based AppleTalk Access Control 10-28
 - Network Number-Based AppleTalk Access Control 10-28
- Creating AppleTalk Access Lists 10-29
- Assigning an Access List to an Interface 10-31
- Filtering Networks Received in Updates 10-31
- Filtering Networks Sent Out in Updates 10-32
- Defining Get-Zone-List Filters 10-33

Permitting Partial Zones 10-34

Requiring Specific Route Zones 10-35

Controlling AppleTalk Names Displayed 10-35

- Setting Service Types Cached 10-36
- Setting the Service Name Lookup Interval 10-38

AppleTalk Configuration Examples 10-38

- Nonextended AppleTalk Routing Between Two Ethernets 10-39
- Configuring Transition Mode 10-40
- Nonextended AppleTalk Routing over X.25 10-41
- Extended AppleTalk Routing Network 10-42
- Extended AppleTalk Routing over HDLC 10-43
- Configuring SNMP in AppleTalk Networks 10-43
- Configuring IPTalk 10-44
 - IPTalk Configuration Steps 10-44
 - IPTalk Configuration Note 10-47
- AppleTalk Access List Configuration Examples 10-49
- Hiding and Sharing Access to Resources with Access Lists 10-57
 - Establishing Free Access to Common AppleShare Servers 10-57
 - Restricting Resource Availability with Access Lists 10-59

Monitoring the AppleTalk Network 10-62

- Displaying AppleTalk Access List Specifications 10-62
- Displaying the Adjacent Routes 10-62
- Displaying the ARP Cache 10-63
- Displaying the Fast-Switching Cache 10-63
- Displaying Global AppleTalk Information 10-64
- Displaying AppleTalk Interface Information 10-65
- Displaying MacIP Status 10-66
 - Monitoring MacIP Servers 10-66
 - Monitoring MacIP Clients 10-69
 - Monitoring MacIP Traffic 10-69
- Displaying Nearby NBP Services 10-70
- Displaying NBP Services Registered by Cisco Routers 10-70
- Displaying Neighboring Routers 10-71

Displaying the Network Routing Table 10-73
Displaying Information About the Sockets 10-75
Displaying AppleTalk Traffic Information 10-76
Displaying Zone Information 10-78

The AppleTalk Ping Command 10-78

AppleTalk NBP Ping Interface 10-79

Help Subcommand 10-80

Parms Subcommand 10-80

Lookup Subcommand 10-81

Poll Subcommand 10-81

Zones Subcommand 10-82

Debugging the AppleTalk Network 10-82

AppleTalk Global Configuration Command Summary 10-84

AppleTalk Interface Subcommand Summary 10-88

Chapter 10

Routing AppleTalk

10

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.
- Cisco's 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."

Cisco's 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*. Cisco routers support 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 10-1.

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.

All routers from Cisco Systems support the AppleTalk network protocol (both extended and nonextended) over FDDI, Ethernet, Token Ring, synchronous serial, and X.25 interfaces.

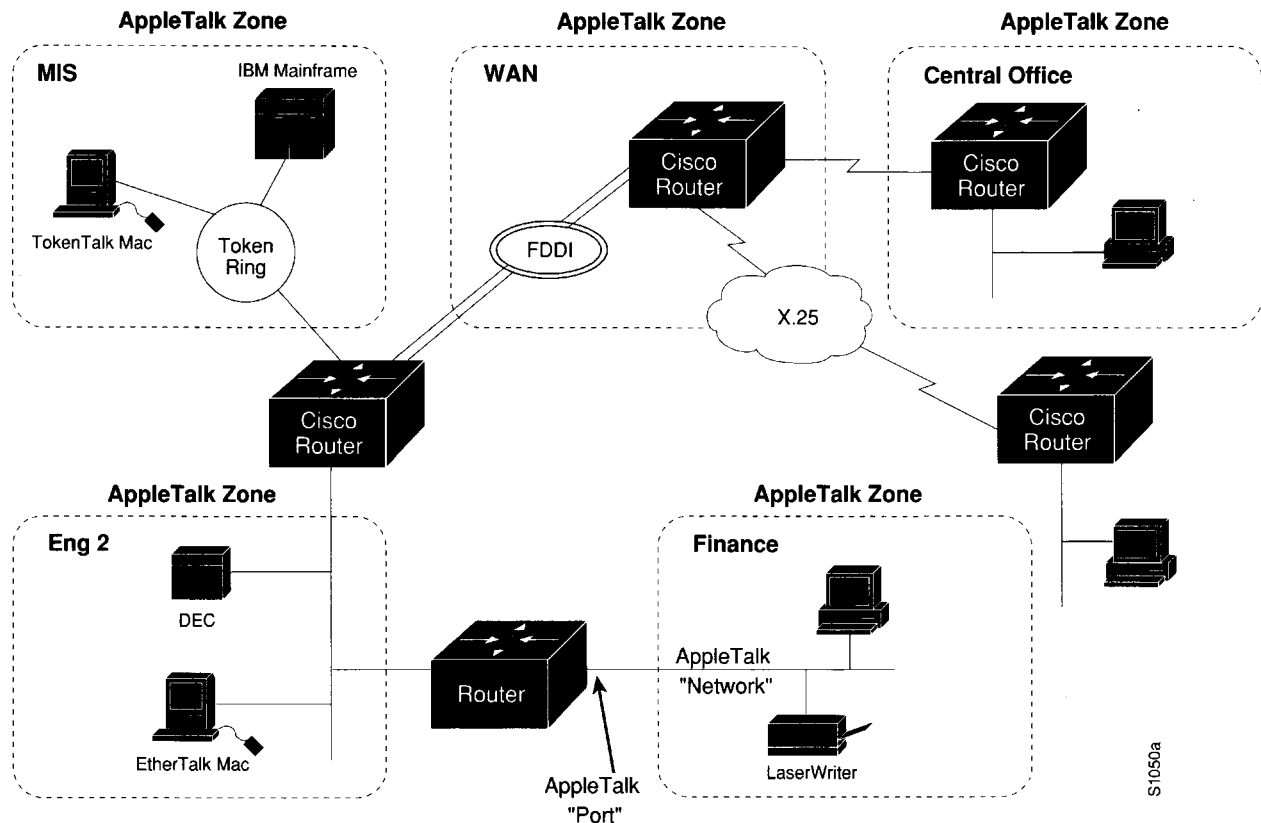


Figure 10-1 AppleTalk Entities

Figure 10-2 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 Cisco 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)

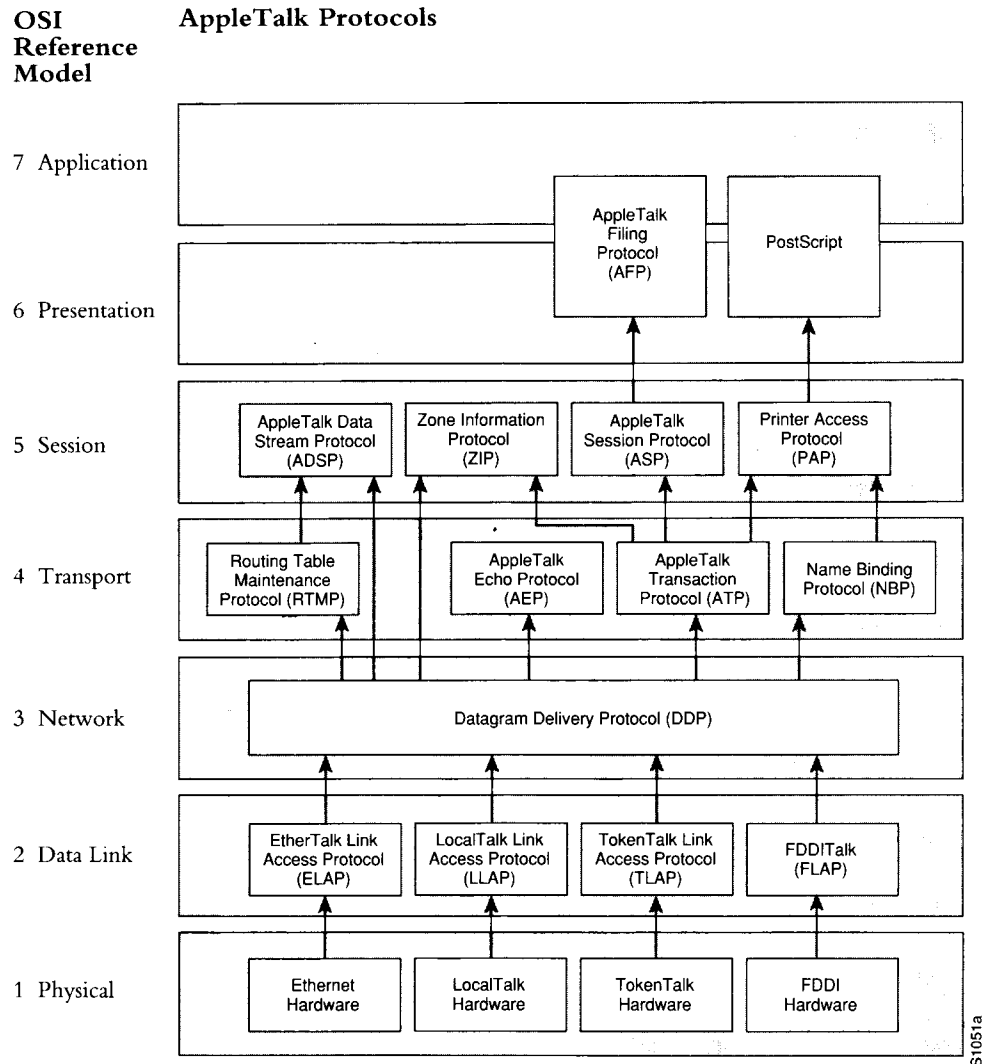


Figure 10-2 AppleTalk and the OSI Reference Model

The Cisco 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. Cisco Systems 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 Macintoshes 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 Cisco), 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” later in this chapter.

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.

Cisco routers running software Release 8.2 or later support 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 internetwork supports AppleTalk Phase 2 (ATp2), you must observe the compatibility rules described in the “Configuration Guidelines (Compatibility Rules)” section of this chapter. 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 Cisco 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 colocated. 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. The section “Configuring AppleTalk Routing” later in this chapter describes the subcommands to use in the zone-naming process.

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” later in this chapter).

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. A Cisco router 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 10-1 illustrates AppleTalk addresses that feature unspecified addressing.

Table 10-1 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 10-3 shows a sample RTMP table and the corresponding network topology.

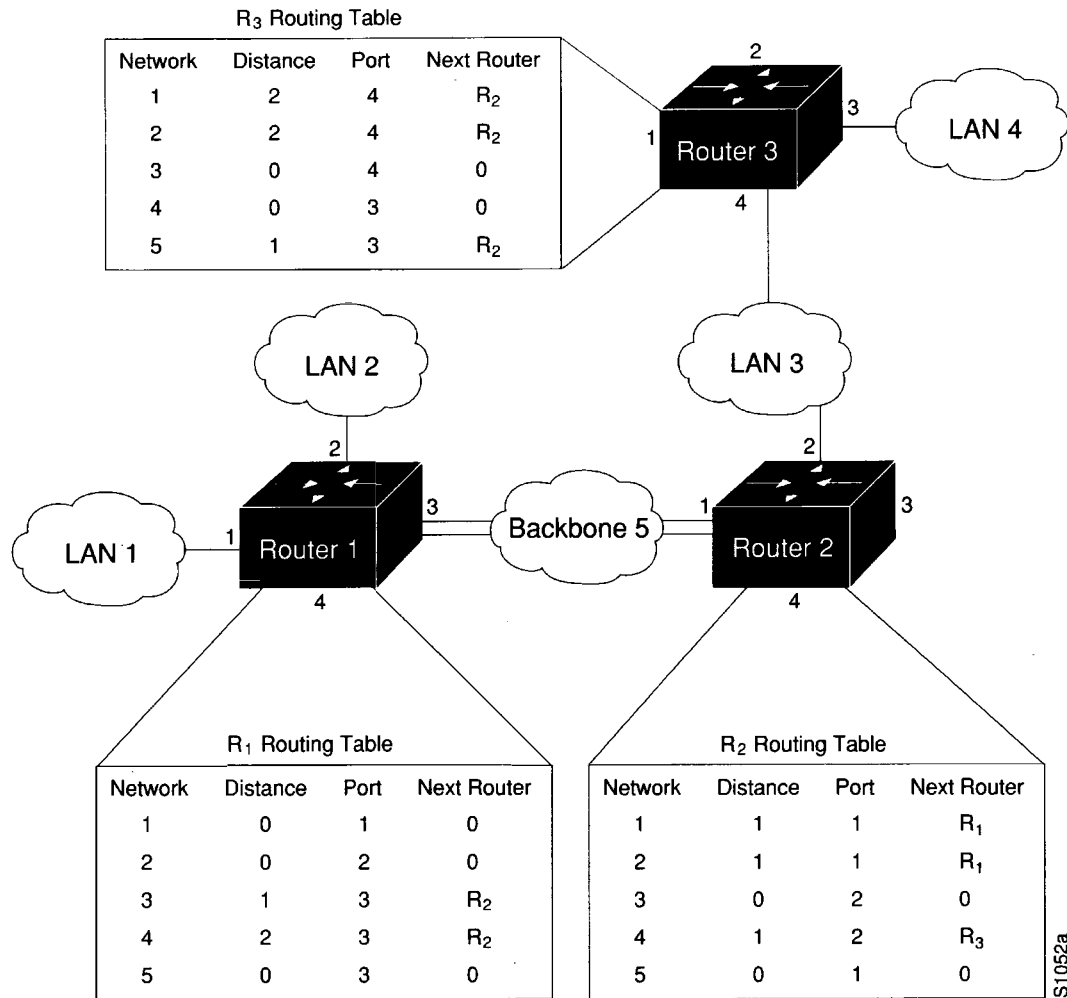


Figure 10-3 Sample AppleTalk Routing Table

Extended AppleTalk Addressing

AppleTalk addresses, as explained in the section “Nonextended AppleTalk Addressing” earlier in this chapter, are composed of a 16-bit network and an 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

Cisco routers 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:ciscoRouter@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.Ethernet3	ciscoRouter	Engineering
4042	8	8	sloth	SNMP Agent	Engineering
4028	8	254	sloth.Ethernet2	ciscoRouter	Engineering
4035	8	254	sloth.TokenRing0	ciscoRouter	Engineering
4036	8	254	sloth.TokenRing1	ciscoRouter	Engineering
4038	8	254	sloth.Hssi0	ciscoRouter	Narrow Beam
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 (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 10-4 illustrates a typical output display for *Inter•Poll* that lists this information.

Device: Net: 4042 Node: 9
gluttony.Ethernet3 - ciscoRouter - Engineering

Packets: Using: Echo Pkts
 Printer Status Packets
 System Info Packets

Interval: Secs

Timeout: Secs

Packets Sent: Rcvd: 4 Lost: 0
Left: 16 Total: 4

	Current	Average	Minimum	Maximum
Hops Away	3	3.00	3	3
Delay (secs)	0.02	0.02	0.02	0.02

Status:
System Bootstrap, Version 4.4(0.5), BETA TEST SOFTWARE © 1986-1991 b...
GS Software (GS3-BFX), Version 9.0(3110), Development Software © 19...
Responder INIT Version: 100
AppleTalk Driver Version: 56 AppleShare not installed

Figure 10-4 Sample illustration of *Inter•Poll* Output

Configuring AppleTalk Routing

This section provides an overview on how to configure Cisco 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:

- Step 1:* Enable AppleTalk routing with the **appletalk routing** command.
- Step 2:* Assign the nonextended AppleTalk addresses with the **appletalk address** interface subcommand.
- Step 3:* Assign the zone name with the **appletalk zone** interface subcommand.

Configuring an extended AppleTalk interface involves these steps:

- Step 4:* Enable AppleTalk routing with the **appletalk routing** command.
- Step 5:* Assign the extended AppleTalk cable range parameters with the **appletalk cable-range** command.
- Step 6:* 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 Cisco router:

- 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 Cisco routers 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 Cisco's representation of AppleTalk addresses is necessary before configuring your router. Refer to the sections "Cisco's Implementation of AppleTalk," "Nonextended AppleTalk Addressing," and "Extended AppleTalk Addressing," earlier in this chapter 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 Cisco's 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 *cisco*zone*.

```
appletalk zone cisco: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

Cisco's 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 Cisco implementation of IPTalk assumes that AppleTalk is already being routed on the backbone, because there is currently no LocalTalk hardware interface for Cisco routers.

The Cisco 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 (ones 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 Cisco's 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 Cisco router'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 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 zonenumber  
no appletalk proxy-nbp network-number zonenumber
```

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 *zonenumber* 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 10-5. 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.

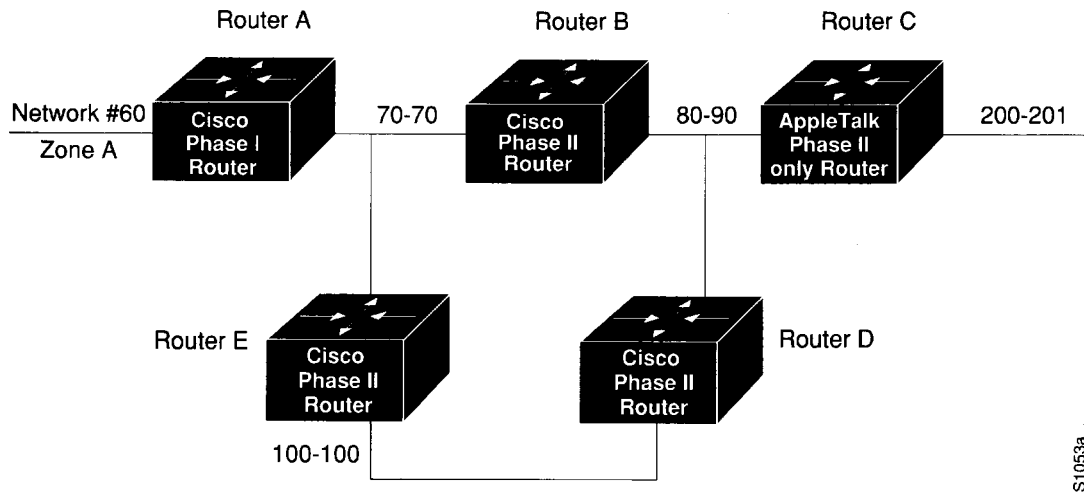


Figure 10-5 Example Network Topology

If Router C generates a 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 a NBP proxy.

If you configured router D with a 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.

S1053a

Cisco routers no longer 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

Cisco routers allow 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 Cisco routers are described in detail in Draft Internet RFC, *A Standard for the Transmission of Internet Packets over AppleTalk Networks*.

Some situations may *require* the use of MacIP; for example, when the attachment media used by Macintoshes does not have device driver support for IP. Cisco and Apple currently support attachment media that do not have native IP software drivers for Macintoshes, but where AppleTalk drivers are available. If your network falls into this category, then configuring MacIP services may provide a simple solution to IP connectivity requirements for Macintoshes.

Another 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 Cisco Router. 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 Cisco router, 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 Cisco implementation of MacIP conforms 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 support for MacIP 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:

- Step 1:** Establish a MacIP server for a specific zone.
- Step 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

Cisco's AppleTalk access lists provide network security by permitting or denying certain packets access to a network interface. Cisco's 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 Cisco router to control access to or from specific networks or zones for a number of services.

Cisco's 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

Cisco 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. Cisco routers permit 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, Cisco routers do 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 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.

Note: In software versions predating SW Release 9.0, the **access-list** command did not include the **network** keyword. If entered into a configuration or found in a boot file, this prior form of the command is transformed into the new form outlined above. The pre-9.0 syntax is accepted as if the **network** keyword were present. Similarly, the network number -1 (previously used to specify *any* network) is accepted as representing the **other-access** keyword. The forms documented here are generated for the **write** command.

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 the removal 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 via **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 via 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” later in this chapter for more information. See the section “Filtering Networks Sent Out in Updates” later in this chapter 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 3.

```
access-list 601 deny network 10
access-list 601 permit other-access
!
interface ethernet 3
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 via 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 via 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 any non-Cisco routers on the network, 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 Macintoshes 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 non-Cisco routers on the network, the command **appletalk getzonelist-filter** is not likely to be useful unless the non-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.

Note: This feature provides IP-style access control with similar functionality to the new AppleTalk-style access control lists. If enabled, the access control list behavior associated with prior software releases is restored. In addition, NBP protocol cannot ensure consistency and uniqueness of name bindings.

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 **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.

- **ciscoRouter**—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**—Cisco's OEM router name
- **SNMP**—Identifies node supporting IP SNMP (only done by some vendors and now considered obsolete)

Note: These non-Cisco services only appear in a display generated using the **show appletalk name-cache** command. Also, if a neighboring router is not one of Cisco's routers, or is running pre-9.0 software, 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 **ciscoRouter** is the only type initially enabled; it cannot be disabled. Cisco routers generate requests to the network segments to which they are 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 via each 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 Cisco routers) 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 **show appletalk name-cache** display.

```
appletalk lookup GatorBox
! In addition to ciscoRouter, 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 ciscoRouter 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 **ciscoRouter**.

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 (five 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 between two Ethernets
- Setting up a transitional configuration where routing is occurring between nonextended and extended AppleTalk networks
- 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 Between Two Ethernets

This example configuration illustrates how to configure routing between two Ethernets. Ethernet 0 is on network 1 at node 128. Ethernet 1 is on network 2 at node 154. The two networks are in the Twilight and No Parking zones, respectively. See Figure 10-6 for an illustration.

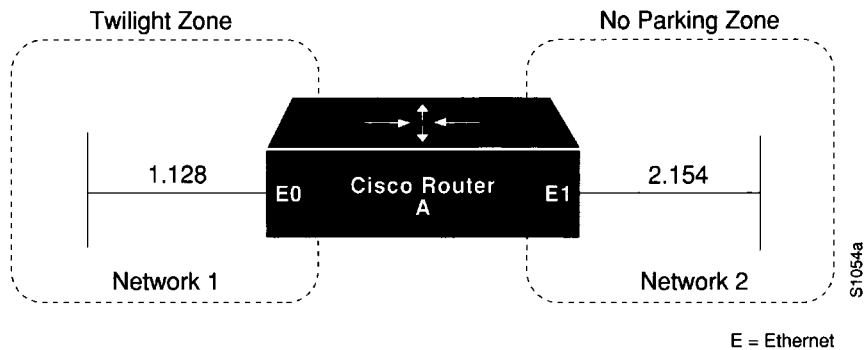


Figure 10-6 Nonextended AppleTalk Routing Between Two Ethernet Networks

```

!
appletalk routing
!
interface ethernet 0
  appletalk address 1.128
  appletalk zone Twilight
!
interface ethernet 1
  appletalk address 2.154
  appletalk zone No Parking
!

```

The next example is a variation of the preceding configuration. It differs in that it has other seed routers on both networks to provide the zone and network number information. In this way, the Cisco router discovers the information dynamically. Refer to Figure 10-7 for an illustration.

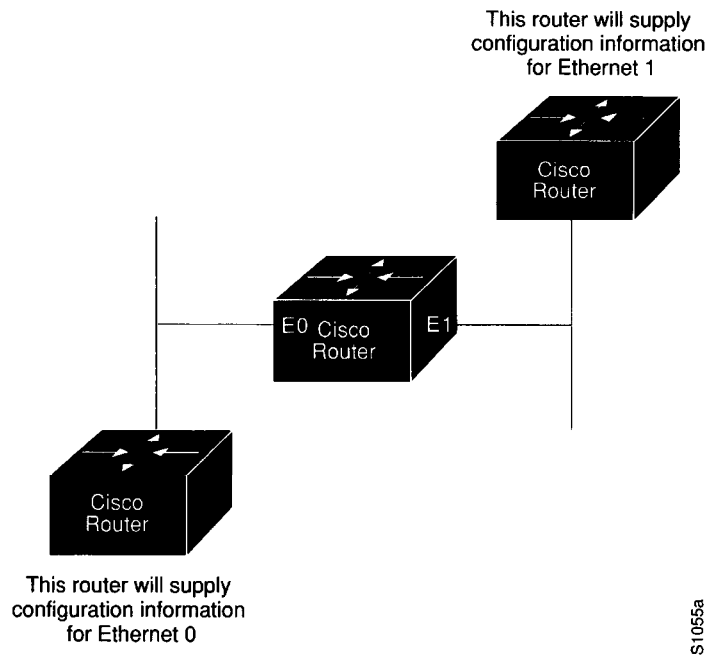


Figure 10-7 Routing Between Seed Routers

```

!
appletalk routing
!
interface ethernet 0
appletalk address 0.0
!
interface ethernet 1
appletalk address 0.0
!

```

Configuring Transition Mode

The Cisco router can be used to route between extended and nonextended AppleTalk networks that exist on the same cable. Other vendors have coined the term *transition mode* for this type of routing.

To do this on the Cisco router, you must have two ports connected to the same physical cable. One port is configured as a nonextended AppleTalk network and the other as an extended AppleTalk network.

Both ports must have unique network numbers, because you are actually routing between two separate AppleTalk networks: an extended and a nonextended network. Figure 10-8 shows an example of the topology and configuration of such a connection.

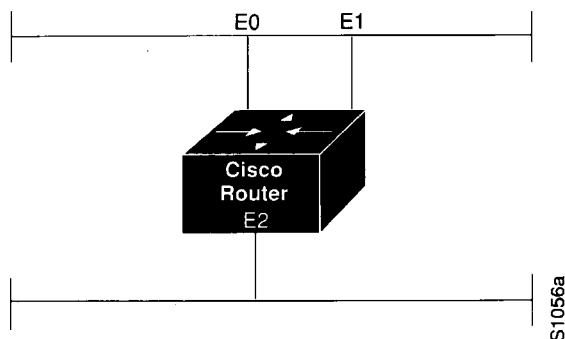


Figure 10-8 Transition Mode Topology and Configuration

```

interface ethernet 0
 appletalk cable-range 2-2
 appletalk zone No Parking
 !
interface ethernet 1
 appletalk address 3.128
 appletalk zone Twilight
 !
interface ethernet 2
 appletalk cable-range 4-4
 appletalk zone Do Not Enter

```

Note: Networks 2-2 and 4-4 in the preceding example have a cable range of one and a single zone in their zone lists. This must be true to maintain compatibility with the nonextended network, Network 3.

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** (see the section “Configuring the Datagram Transport on Commercial X.25 Networks” in Chapter 8).

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.

Note: Refer to Chapter 4 of this manual for information about configuration SNMP for the router. The **snmp-server** commands are global configuration commands.

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 IP'Talk

IP'Talk is AppleTalk *encapsulated* in IP datagrams. IP'Talk 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 Cisco router to use CAP IP'Talk and other IP'Talk implementations.

Note: If your system is a Sun or DEC 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 IP'Talk support. However, other UNIX systems for which EtherTalk support is not available in CAP must run CAP in a mode that depends upon IP'Talk.

IP'Talk Configuration Steps

The procedure that follows outlines the basic steps for setting up Cisco routers and UNIX hosts for operation using IP'Talk 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 Cisco 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.

- Step 1:** Set up the Cisco routers for AppleTalk. The routers talk to each other and to Apple products using more standard protocols, such as EtherTalk or TokenTalk. IP Talk 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 IP Talk. 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.
- Step 2:** Ensure that IP is enabled on the interface to be used to communicate with the UNIX system. Refer to Chapters 13 and 14 of this manual for more information about configuring IP. Since IP Talk 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.
- Step 3:** Allocate an AppleTalk network number for IP Talk. A separate AppleTalk network number is needed for each IP subnet that is to run IP Talk. It is possible to have a number of UNIX machines on the same subnet. They all use the same AppleTalk network number for IP Talk. They must have their own individual node ids. It is possible for the same router to have IP Talk enabled on several interfaces. Each interface must have a different AppleTalk network number assigned for use by IP Talk, since each interface will be using a different IP subnet.
- Step 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

- Step 5:** Decide on a zone name for IP Talk. There are no special constraints on choice of zone name. The same zone name can be used for several networks. IP Talk and normal networks can be combined in the same zone if desired.
- Step 6:** Decide which UDP ports you are going to use for IP Talk. 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 Cisco router to use the same ports.

Step 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:

- Via EtherTalk phase 2 using network number 1792 and zone MIS-Development
- Via 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.

Step 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 Cisco router. 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 Cisco router 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 Cisco router 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 Cisco router.

Step 9: Make sure that your CAP software is using the same UDP port numbers as the Cisco router. Currently, CAP's default is the same as Cisco'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
```

Step 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 Cisco's IPTalk support, the file *atalkatab* is neither necessary nor desirable. Cisco's 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.

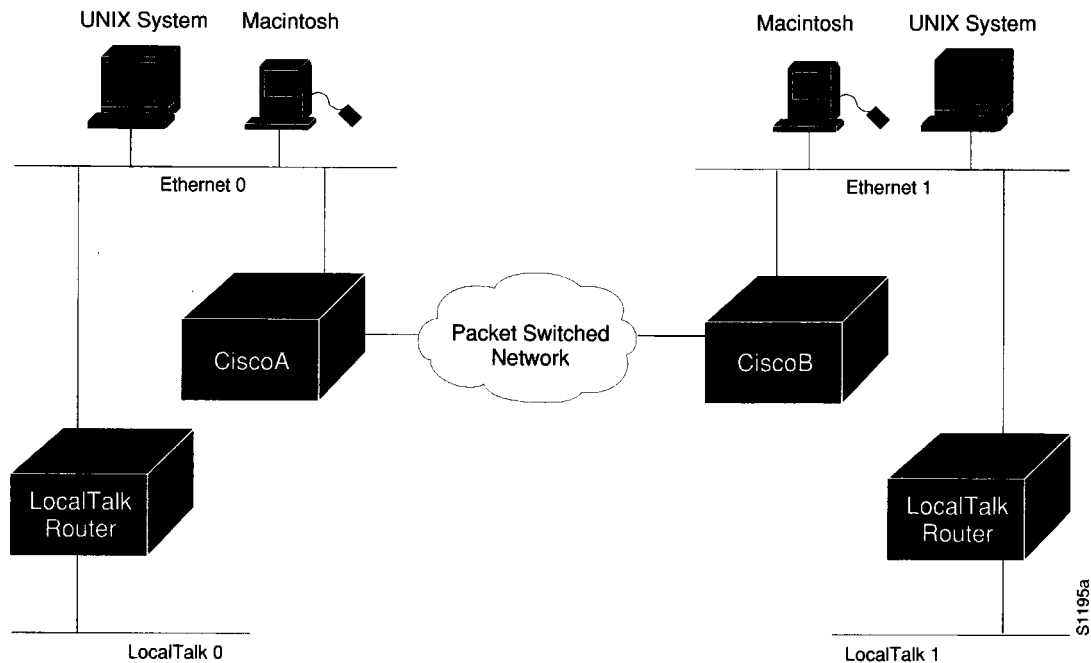


Figure 10-9 IPTalk Configuration Example

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 Cisco routers, this is not necessary. The Cisco implementation of IPTalk integrates IPTalk into the normal AppleTalk network routing. Consider the example network diagram illustrated in Figure 10-9.

In this example, CiscoA and CiscoB must enable both standard AppleTalk (EtherTalk) and IPTalk on the Ethernets shown. They will use EtherTalk to communicate with the LocalTalk Routers and Macintoshes, 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 Cisco router. 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 Cisco routers 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 10-2 through Table 10-4 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 10-2 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 10-3 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 10-4 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 50-60	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 Cisco's 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 discusses 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 of 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. An Ethernet backbone exists on each side which connects these other routers to Gatekeeper. (E0 is the World backbone and E1 is the Corporate backbone).

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 (GZL) 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 10-5 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 10-5 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 10-6 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 10-6 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 10-5).
- a3 is eliminated by rule 3.

Table 10-7 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 10-7 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 10-8 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 10-8 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 via 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.

Partial Zone Advertisement Configuration Example

Figure 10-10 illustrates a situation where you might want to allow for the partial advertisement of a particular zone.

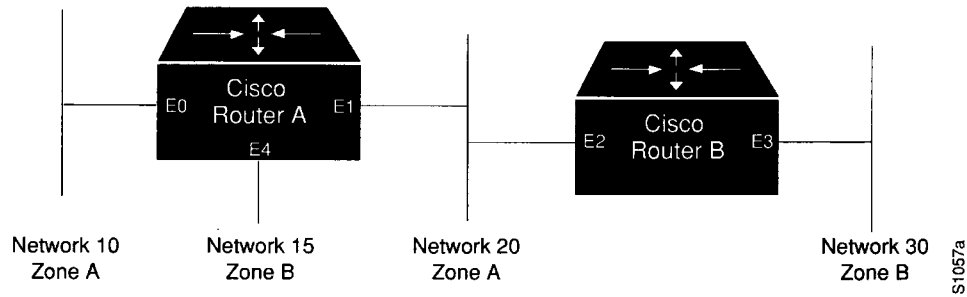


Figure 10-10 Example Topology of Partially Obscured Zone

Assume that Router B includes a distribution filter (applied with the **appletalk distribute-list** interface subcommand) on the interface Ethernet 3 that excludes Network 10. The associated commands might look like this:

```
access-list 612 deny network 10
int eth 3
appletalk distribute-list 612 out
appletalk distribute-list 612 in
```

For Network 30, normal (default) behavior would be for Network 10 and Network 20 to be eliminated from any routing updates sent, although Network 15 would be included in routing updates (same zone as Network 30). Using the **appletalk permit-partial-zones** configuration command has the following effects:

- If permit-partial-zones is enabled (**appletalk permit-partial-zones**), the routing updates exclude Network 10, but *includes* Network 15 and Network 20.
- If permit-partial-zones is disabled (**no appletalk permit-partial-zones**), the routing updates exclude both Network 10 and Network 20, but still include Network 15. This is generally considered the preferred behavior and is the default.

Table 10-9 provides an overview of the associations between the networks illustrated in Figure 10-10. Table 10-10 details the effects of enabling and disabling partial-zone advertisement with the **permit-partial-zones** command.

Table 10-9 Zone and Interface Associations for Partial Zone Advertisement Example

	Network 10	Network 15	Network 20	Network 30
Zone	A	B	A	B
Interface(s)	Ethernet 0	Ethernet 4	Ethernet 1 Ethernet 2	Ethernet 3

Table 10-10 Partial-zone Advertisement Control on Network 30

Command Condition	Network 10	Network 15	Network 20	Network 30
Enabled	Not Advertised on Network 30	Advertised on Network 30	Advertised on Network 30	—
Disabled	Not Advertised on Network 30	Advertised on Network 30	Not Advertised on Network 30	—

Hiding and Sharing Access to Resources with Access Lists

The following examples illustrate the use of AppleTalk access lists to manage access to certain resources.

Establishing Free Access to Common AppleShare Servers

Consider a situation in which you wish to provide access to several AppleShare servers on a network directly connected to two routers but want to restrict cross-access among other networks that are connected to these routers. Figure 10-11 illustrates an environment that reflects this situation. The configuration example and associated discussion describe how access lists can be used to provide control.

The following configuration listing provides the configuration for Router A and Router B in Figure 10-11. Only interface Ethernet 1 (E1 on Router A) and interface Ethernet 2 (E2 on Router B) are configured in order to provide the control described here.

In this example, the goal of network administrators is to allow all users on the various networks connected to both Router A and Router B to be able to access the AppleShare servers AS1 and AS2 in the zone FreeAccessZone. A second requirement is to block cross-access through this zone. In other words, users in zones MIS1, MIS2, and LocalTalk (connected to Router A, interface E0) are not allowed to have access to any of the resources on networks connected to interface E4 on Router B. Similarly, users in zones Engineering, Test, and LocalTalk (connected to Router B, interface E4) are not allowed to have access to any of the resources on networks connected to interface E0 on Router A.

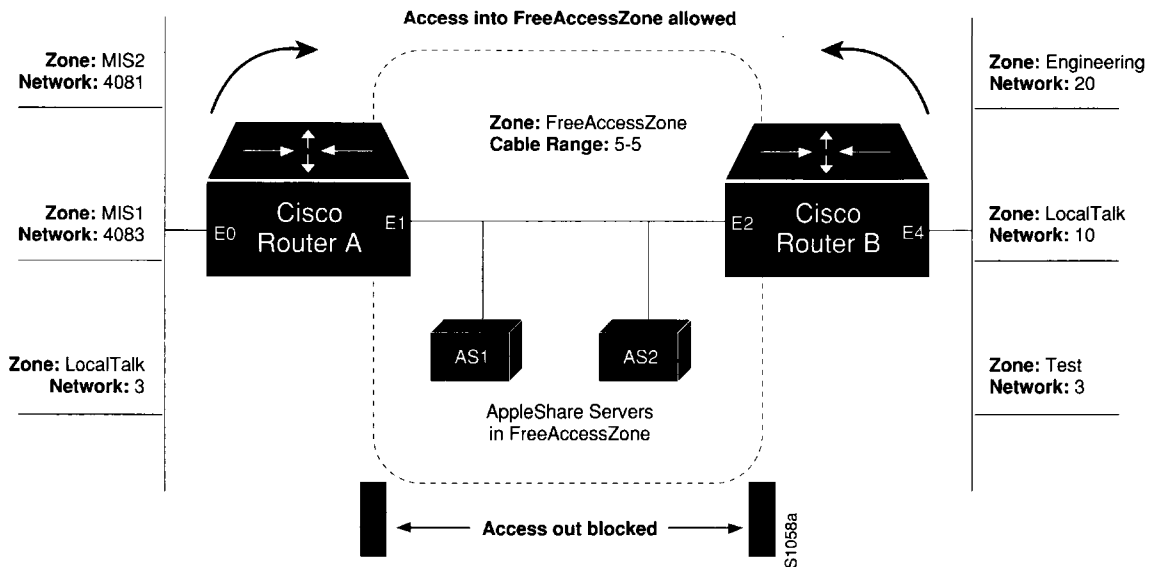


Figure 10-11 Controlling Access to Common AppleTalk Network

Note: Although there are networks that share the same number on interfaces E0 and E4, and zones that have the same name, none have the same network number and zone specification (except FreeAccessZone). The two routers do *not* broadcast information about these networks through FreeAccessZone. The routers only broadcast the cable range 5-5. As configured, FreeAccessZone only sees itself. However, since no other limitations have been placed on advertisements, the FreeAccessZone range of 5-5 propagates out to the networks attached to E0 (RouterA) and E4 (RouterB); thus, resources in FreeAccessZone are made accessible to users on all those networks.

Router A Configuration

```
! Global configuration specification of access list 601
access-list 601 permit cable 5-5
access-list 601 deny other-access
!
interface ethernet 1
  appletalk cable-range 5-5
  appletalk zone FreeAccessZone
  appletalk distribute-list 601 out
  appletalk distribute-list 601 in
```

Router B Configuration

```
! Global configuration specification of access list 601
! (access lists are identical to RouterA)
access-list 601 permit cable 5-5
access-list 601 deny other-access
!
interface ethernet 2
```

```

! Specifications for Ethernet 2 on RouterB are same
! as specifications for Ethernet 1 on RouterA
appletalk cable-range 5-5
appletalk zone FreeAccessZone
appletalk distribute-list 601 out
appletalk distribute-list 601 in

```

Restricting Resource Availability with Access Lists

In the preceding example, shared-resource access was granted to all users in the various AppleTalk zones connected to the two routers. At the same time, access between resources on either side of the common zone was completely denied. There may be instances where a greater degree of control is required—possibly where resources in some zones are to be allowed access to resources in certain other zones, but denied access to other specific zones. Figure 10-12, the accompanying discussion, and the configuration command examples that follow the figure, illustrate such a situation.

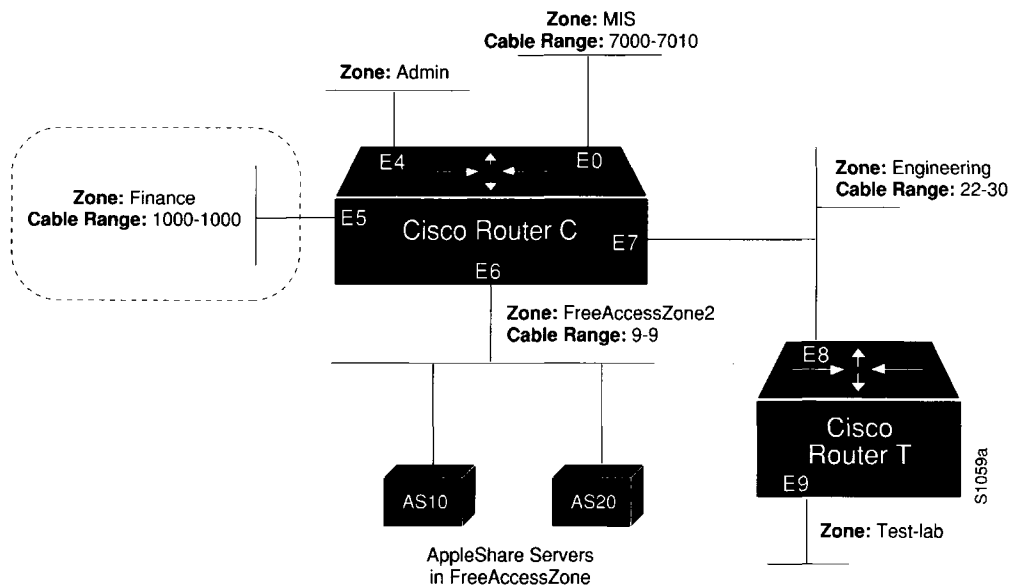


Figure 10-12 Controlling Resource Access Among Multiple AppleTalk Zones

The following guidelines are the administrative objectives for this example:

- Users in zones Engineering (E7) and MIS (E0) are to be allowed free access to each other.
- All users in all zones are to be allowed access to FreeAccessZone2 (E6).
- No users in any zone, with the exception of users in Finance, are to be allowed access to resources in Finance.

Router C Access List Configuration

The preceding specifications require three access lists. The configuration commands associated with each are as follows:

```
! Global configuration specification of access list 601:
access-list 601 permit cable 9-9
access-list 601 deny other-access
! Global configuration specification of access list 610:
access-list 610 permit zone Finance
access-list 610 permit zone FreeAccessZone2
access-list 610 deny additional-zones
! Global configuration specification of access list 620:
access-list 620 deny zone Finance
access-list 620 permit zone MIS
access-list 620 deny additional-zones
access-list 620 permit zone FreeAccessZone2
access-list 620 permit zone Engineering
```

The effects of these access lists can be briefly defined as follows:

- Access list 601 mirrors the access control list provided in the preceding example (illustrated in Figure 10-11). It is intended to be used to allow access to resources on Free-AccessZone2.
- Access list 610 is intended to be used to control access in and out of the zone Finance.
- Access list 620 is intended to be used to accommodate the requirement to allow users in zones Engineering and MIS to mutually access network resources.

The specification of the **access-list** global configuration command is only the first half of the access control process. The second half of the process is to then apply the access lists to interfaces using the various **appletalk** interface subcommands. The assignment of access lists to specific interfaces for this example follows.

Router C Interface Ethernet 0 Configuration

Interface Ethernet 0 is associated with MIS. The configuration command listing is as follows:

```
interface ethernet 0
  appletalk cable-range 7000-7010
  appletalk zone MIS
  appletalk distribute-list 620 out
  appletalk distribute-list 620 in
```

By specifying access list 620 using **distribute-list** interface subcommands, the following results:

- FreeAccessZone2 is advertised into MIS.
- Advertisements of Finance are blocked.
- Advertisements between Engineering and MIS are allowed.

Router C Interface Ethernet 5 Configuration

Interface Ethernet 5 is associated with Finance. This zone requires some specific controls. The configuration command listing for Ethernet 5 is as follows:

```
interface ethernet 5
  appletalk cable-range 1000-1000
  appletalk zone Finance
  appletalk distribute-list 610 out
  appletalk access-group 610
```

The configuration for Ethernet 5 requires using both a **distribute-list out** interface subcommand and an **access-group** subcommand. Using these commands has the following effects:

- With the **distribute-list out** subcommand, Finance is limited to accessing Finance and FreeAccessZone2 only.
- The **access-group** subcommand filters packet traffic, thus it blocks access to any devices in *Finance* from outside of this zone.

Router C Interface Ethernet 6 Configuration

FreeAccessZone2 is on interface Ethernet 6. Since users in all zones are to have access to resources in this zone, the configuration is as follows:

```
interface ethernet 6
  appletalk cable 9-9
  appletalk zone FreeAccessZone2
  appletalk distribute-list 601 out
  appletalk distribute-list 601 in
```

Router C Interface Ethernet 7 Configuration

Ethernet 7 has a configuration that mirrors the configuration for Ethernet 0, because the users in zones MIS and Engineering are to have mutual resource access. The configuration would be as follows:

```
interface ethernet 7
  appletalk cable-range 22-30
  appletalk zone Engineering
  appletalk distribute-list 620 out
  appletalk distribute-list 620 in
```

Implicit Configuration of Zones Admin and Test-Lab

Conspicuously omitted from this configuration listing are any specific configuration listings pertaining to the Test-Lab (on Router T interface E9, and connected to Router C via E7) and Admin (on Router C interface E4). These zones are omitted because there are no requirements relating to them listed in the original objectives. Some access control is implicitly handled with the assignment of the stated access lists:

- Users in zone Admin can see the Finance zone, but cannot see resources in that zone. However, as for all zones, resources in FreeAccessZone2 are available, but none of the users in any of the other zones can access resources in Admin.

- In the absence of the assignment of access lists on Router T, users in Test-Lab can access the resources in FreeAccessZone2 and zone Engineering. With the exception of Engineering, no other zones can access resources in Test-Lab.

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 than 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, Ethernet1, no zone set
C Net 4160-4160 directly connected, Ethernet0, zone Low End SW Lab
C Net 4172-4172 directly connected, TokenRing0, zone Low End SW Lab
R Net 6160 [1/G] via urk, 0 sec, TokenRing0, 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 10-11 describes the fields.

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
AppleTalk	4172.30	-	0000.3080.84ab	SNAP	TokenRing0
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	Ethernet1
AppleTalk	4172.224	206	0000.3080.8453	SNAP	TokenRing0
AppleTalk	4160.150	-	0000.0c00.d8dd	SNAP	Ethernet0

Table 10-11 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 (Address Resolution Protocol): 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 Ethernet1 AA000400013400000C000E8C809B84BE02
*    33.000 Ethernet1 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 Phasel, 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 propogated 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.

This 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 1 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:

```
TokenRing 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 1 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 1 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 apple 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_upRe-
source #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 10-12.
- **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:

- Step 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.
- Step 2:** Next, execute a **show appletalk interface** command; with this command you can determine the status of AppleTalk routing and the specific interface itself.
- Step 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 10-12.

Table 10-12 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. Warning: 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** command 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 a 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 last 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 (described in Chapter 13 of this manual).

Displaying Nearby NBP Services

Use the `show appletalk name-cache EXEC` command to display 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                         ciscoRouter    Low End SW Lab
  4160  21  254  bill                               ciscoRouter    Low End SW Lab
  4160  150 254  pag.Ethernet0                     ciscoRouter    Low End SW Lab
  4172  30  254  pag.TokenRing0                   ciscoRouter    Low End SW Lab
  4172  224 254  urk                               ciscoRouter    Low End SW Lab
  6160  69  254  urk                               ciscoRouter    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: The routers listed in this display (except pag) are running software images that predate Cisco SW Release 9.0—which accounts for name differences in this display. Non-Cisco routers also 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 Cisco 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                   ciscoRouter    Low End SW Lab
  4160 211   8  pag                               SNMP Agent     Low End SW Lab
```

4172	84	254	pag.TokenRing0	ciscoRouter	LES Tokenring
4172	84	8	pag	SNMP Agent	LES Tokenring
200	75	254	myrouter.Ethernet1	ciscoRouter	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 Cisco service types are:
 - **ciscoRouter**—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, Ethernet8, uptime 133:28:06, last update 1 sec ago
81.82, Fddi0, uptime 266:11:44, last update 7 secs ago
81.81, Fddi0, uptime 267:30:28, last update 958334 secs ago
Neighbor is down.
```

```
29.200, Ethernet3, uptime 263:45:50, last update 948440 secs ago
Neighbor has restarted 2 times in 267:59:53.
Neighbor is down.
81.80, Fddi0, uptime 268:00:08, last update 963617 secs ago
Neighbor is down.
17.128, Ethernet2, uptime 133:26:43, last update 2 secs ago
Neighbor has restarted 1 time in 268:00:21.
69.163, Ethernet0, 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       TokenRing0, 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, TokenRing0, 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:ciscoRouter@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, Ethernet1, 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 ten-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, Ethernet1, zone Localtalk
C Net 4 directly connected, Ethernet0, zone Twilight
C Net 6 directly connected, Ethernet3, zone Heavenly
R Net 11 [3/G] via 254.163, 8 sec, Ethernet1, zone UDP
R Net 17 [1/G] via 254.163, 8 sec, Ethernet1, 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, Ethernet1, zone idontcare
R Net 55 [1/G] via 254.130, 9 sec, Ethernet1, 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, Ethernet1, zone Empty Guf
C Net 80 directly connected, Ethernet4, zone Light
R Net 99 [2/G] via 4.129, 1 sec, Ethernet0, zone BammBamm
C Net 254 directly connected, Ethernet1, 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, Serial3, zone Magnolia Estates
R Net 2003 [4/G] via 80.129, 6 sec, Ethernet4, zone Bldg-13
R Net 2004 [2/G] via 80.129, 6 sec, Ethernet4, 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, Ethernet1, zone UDP
R Net 2024 [4/G] via 80.129, 3 sec, Ethernet4, zone Bldg-17
R Net 3004 [1/G] via 80.129, 3 sec, Ethernet4, 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, Ethernet4, zone Bldg-17
R Net 3880 [1/G] via 999.2, 0 sec, Serial3, zone Magnolia Estates
R Net 5002 [2/G] via 80.129, 3 sec, Ethernet4, 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, Ethernet4, 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 - RIMP 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 cisco
R Net 560 [4/G] via gatekeeper, 4 sec, Ethernet0, zone outside cisco
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, Ethernet1, 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 3**, the display looks like this:

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

C Net 999 directly connected, Serial3, zone Magnolia Estates
R Net 3880 [1/G] via 999.2, 3 sec, Serial3, 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]
```

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 10-13.

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 10-13 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.

Field	Description
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.
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
CISCO 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 cisco	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 CISCO 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 [5]:
Datagram size [100]:
Timeout in seconds [2]:
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. Cisco routers only support *HearSelf* on Ethernet.

The **ping** command uses the characters in Table 10-14 to indicate the success or failure of each packet in the **ping** sequence.

Table 10-14 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, type **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 [5]: 100
interval [5]: 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 **parms** 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 [5]: 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 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 Cisco 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** configuration 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 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 Cisco 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 **ciscoRouter**.

[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 via 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 an 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.

Chapter 11

Routing CHAOSnet

11

Cisco's Implementation of CHAOSnet 11-1

CHAOSnet Addresses 11-1

Configuring CHAOSnet Routing 11-2

Monitoring CHAOSnet 11-2

Debugging CHAOSnet 11-3

Chapter 11

Routing CHAOSnet

11

This chapter describes Cisco Systems' implementation of the CHAOSnet routing protocol.

Cisco's 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 Cisco Systems 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 Cisco Systems 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 Cisco Systems 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, Cisco Systems 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. See Chapter 14 of this manual for more information about the **router** command.

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 14 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 these 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.

Chapter 12

Routing DECnet

12

Cisco's Implementation of DECnet 12-1

DECnet Phase IV Addresses 12-2

Configuring DECnet Routing 12-4

Enabling DECnet Routing 12-4

Assigning the Cost 12-5

Specifying the Node Type 12-6

Specifying Node Numbers and Area Sizes 12-6

Specifying the Maximum Route Cost for Interarea Routing 12-7

Specifying the Maximum Route Cost for Intra-area Routing 12-8

Configuring Maximum Visits 12-9

Configuring Path Selection 12-9

Altering DECnet Defaults 12-10

 Adjusting Timers and the Route Cache 12-11

 Specifying the Designated Router 12-12

Configuring DECnet on Token Ring 12-12

Managing Traffic Using DECnet Access Lists 12-13

Configuring DECnet Access Lists 12-13

Configuring Extended Access Lists 12-14

DECnet Connect Initiate Filtering 12-14

 Connect Initiate Filter Configuration Considerations 12-16

 Connect Initiate Filtering Examples 12-17

Configuring Access Groups 12-18

Configuring In- and Out-Routing Filters 12-18

DECnet Phase IV-to Phase-V Conversion 12-19

Designing a Network to Support Both Phase IV and Phase V 12-21

DECnet Configuration Examples 12-22

Establishing Routing; Setting Interfaces; Maximum Address Space 12-22

Level 1 and Level 2 Routing; Designated Router 12-22

Phase IV to Phase V Conversion 12-23

The Address Translation Gateway 12-24

ATG Command Syntax 12-24

ATG Configuration Examples 12-24

Limitations of the ATG 12-27

DECnet Monitoring Commands 12-27

Displaying DECnet Status 12-27

Displaying the DECnet Address Mapping Information 12-28

Displaying the DECnet Routing Table 12-28

Displaying DECnet Traffic Statistics 12-29

Debugging DECnet 12-31

DECnet Global Configuration Command Summary 12-32

DECnet Interface Subcommand Summary 12-35

Chapter 12

Routing DECnet

12

This chapter describes Cisco Systems' implementation of DECnet Phase IV for the Cisco network server product line. Topics and tasks described in this chapter include:

- Cisco's 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, which is described in Chapter 15 and Chapter 16. Support for DECnet Phase IV/Phase V conversion is discussed in this chapter.

Cisco's Implementation of DECnet

Digital Equipment Corporation (DEC) designed the DECnet stack of protocols in the 1970s as part of its Digital Network Architecture (DNA). DECnet support on a Cisco router includes local area and wide area DECnet Phase IV routing over Ethernet, Token Ring, FDDI, and serial lines as follows:

- The router uses HDLC framing rather than DEC's DDCMP framing for point-to-point lines. If you construct a network using both Cisco Systems and DEC equipment, you must ensure that each point-to-point line has the same type of equipment on both ends.
- Cisco and DECnet Phase IV routers have incompatible X.25 support. As with point-to-point lines, you must use a single vendor's equipment on the X.25 portion of your network.
- The method of DECnet encapsulation over Token Ring differs between Cisco and DEC equipment. You can configure your Cisco equipment to interoperate with DEC equipment, or you can configure your Cisco equipment to operate with other Cisco routers that use prior versions of router software.
- Cisco gives you additional security options through access lists.

Cisco routers 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.

DEC uses some nonroutable protocols that are not part of the DECnet stack. Neither Cisco nor DEC routers can route protocols like MOP (discussed later in this chapter) and LAT, the DEC terminal server protocol. These protocols must be bridged; bridging concepts are described in Chapter 21 of this manual.

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. Figure 12-1 shows the DECnet approach.

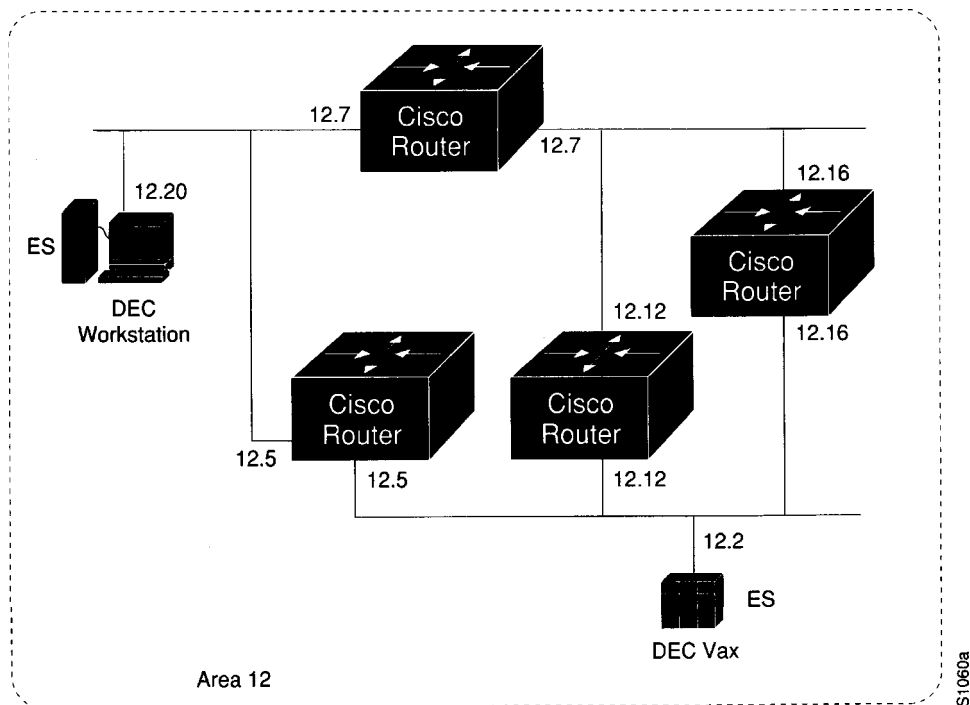


Figure 12-1 DECnet Nodes and Areas

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: $(1024 * \text{area}) + \text{node}$. 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:

$$12 * 1024 + 75 = 12363 \text{ (base 10)} = 304B \text{ (base 16 or hex notation)}$$

The resulting MAC address is AA00.0400.4B30.

Note: 802.5 Token Ring media transmit bytes in bit-reversed order; the MAC address defined above (for Ethernet) would be 5500.2000.D20C for Token Ring

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 Cisco router 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 Cisco Systems' implementation of DECnet are a subset of the parameters you can modify in DEC's Network Control Program (NCP). Cisco 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; Cisco's DECnet implementation does not set parameters by communicating with NCP.

Configuring DECnet Routing

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

- Step 1:** Enable DECnet routing and specify system-wide host addresses; use the **decnet routing** global configuration command.
- Step 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.
- Step 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.
- Step 4:** You can alter the maximum node number and maximum area number with the optional **decnet max-address** and **decnet max-area** commands.
- Step 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, Phase IV/V conversion, and so on. 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
```

Note: Enabling DECnet changes the MAC addresses of the router's interfaces. This is not a problem on routers equipped with nonvolatile memory. On systems that attempt to get their IP network addresses from network servers rather than from nonvolatile memory, there may be a problem with the hardware addresses changing and confusing other IP-speaking hosts. If you are attempting to use DECnet on such a configuration, be sure to set all global DECnet parameters before enabling DECnet routing on the interfaces.

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
```

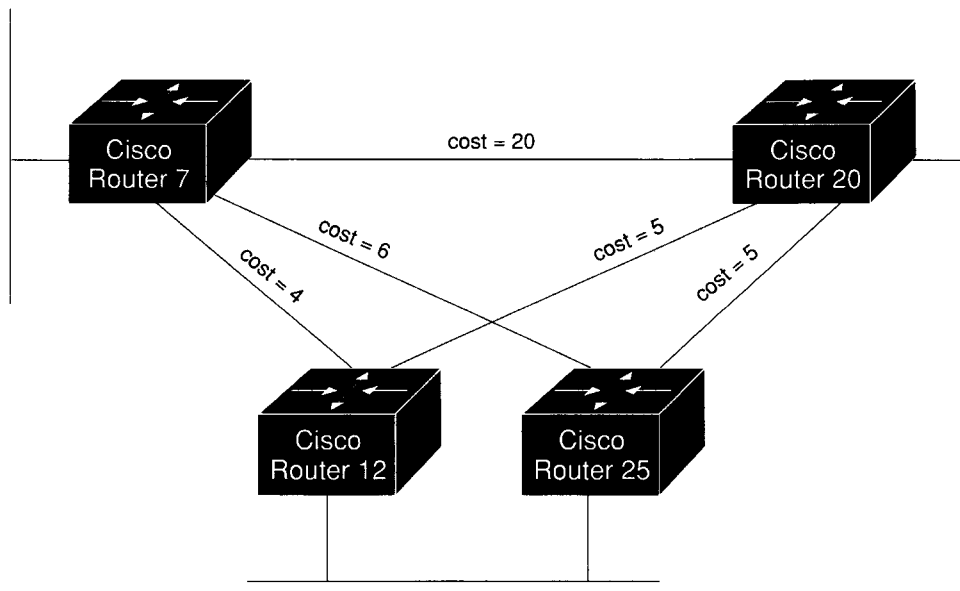
The argument *cost-value* is an integer from 1 to 63. There is no default cost for an interface, although suggested costs are 1 for FDDI, 4 for Ethernet, and greater than 10 for serial links. Use the **no decnet cost** subcommand to disable DECnet routing for an interface.

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
```

Figure 12-2 shows four routers, three Ethernets, and the various routes linking them. Each link has a different cost associated with it. The least expensive route from Router 7 to Router 20 is via Router 12.



S1061a

Figure 12-2 DECnet Cost Values

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 DEC 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 *value* keyword can vary from 1 to 63 (the default). If a packet exceeds *value*, the router discards the packet. DEC 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 keyword *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, Cisco's 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, slow serial links, and so on), 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. DEC calls this parameter the *broadcast routing timer* because they use a different timer for serial lines; Cisco's 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
!
```

Configuring DECnet on Token Ring

The routers support DECnet on Token Ring interfaces. When you configure a Cisco router to support DECnet over Token Ring, you can choose whether to support Cisco interoperation with non-Cisco equipment.

Configuring a Cisco router for DECnet on Token Ring is similar to configuring DECnet on Ethernet. The only difference is the specification of a Token Ring rather than an Ethernet interface. The syntax for the interface command is as follows:

```
interface tokenring number
```

The parameter *number* refers to the interface number.

Example

This example sets interface Token Ring 0 for DECnet routing:

```
!
interface tokenring 0
decnet cost 4
!
```

To configure a Cisco router for operation on the same Token Ring with routers running software versions prior to 9.1, specify Cisco-style encapsulation. The syntax for the subcommand follows:

```
decnet encapsulation {pre-dec|dec}
```

Example

This example sets Cisco-style encapsulation for DECnet routing, which means that Cisco and DEC equipment will not interoperate over Token Ring:

```
!
interface tokenring 0
decnet encapsulation pre-dec
decnet cost 4
!
```

In this mode, you can use the Token Ring as a backbone or transit network for DECnet routing but you cannot communicate with other non-Cisco DECnet nodes on the Token Ring until all Cisco routers use software release 9.1 or later. The default is to have interoperation enabled.

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 13 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-spec*

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:
 - `eq 17`—Equal to 17
 - `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 1.

```
!  
interface ethernet 1  
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 1 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 1  
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 Chapter 15 and Chapter 16 for an explanation of configuring OSI CLNP routing and for a review of the terminology.

DEC 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 DEC 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 Chapter 15 of this manual for details about DECnet Phase V cluster alias support.

Cisco's implementation is identical to DEC's in the conversion algorithms listed. Cisco also handles cluster aliases in the same way.

Cisco's implementation differs from DEC's in how reachability information is advertised. Cisco's 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 DEC's implementation converts as soon as possible.

Cisco routers interoperate with DEC routers, and DEC hosts do not differentiate between a Cisco router and a DEC router.

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 Cisco 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

DEC Phase V hosts can use either Phase IV or Phase V packet format. A DEC Phase V host chooses the format to use based on the type of router hello packets that it sees.

Cisco routers with conversion enabled advertise reachability to both Phase IV hosts and Phase V hosts in both Phase IV and Phase V routing updates.

Cisco 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 that support 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 Cisco router. 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 Ethernet interfaces are given costs of four.

Example 1

```
!  
decnet routing 6.10  
decnet node area  
!  
interface ethernet 0  
decnet cost 4  
interface ethernet 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
```

```

dechnet cost 4
dechnet router-priority 127
!
interface serial 0
dechnet cost 20
!

```

In the third example, the router is a Level 1 router in area 7. The serial link is slower than in the previous example (9.6 vs. 56 kbps), so it has a higher cost.

Example 3

```

!
dechnet routing 7.12
dechnet node routing-iv
!
interface ethernet 0
dechnet cost 4
interface ethernet 1
dechnet cost 4
interface serial 0
dechnet 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 **dechnet 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 1, 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

```

!
dechnet routing 54.6
clns routing
router iso-igrp Field
net 47.0006.0200.0000.0000.0100.0036.AA00.0400.06D8.00
dechnet conversion 47.0006.0200.0000.0000.0100
interface ethernet 0
dechnet 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 Cisco router 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 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 12-3, the Cisco router is connected to two DECnet networks using Ethernet. The examples following Figure 12-3 refer to the configuration in the figure.

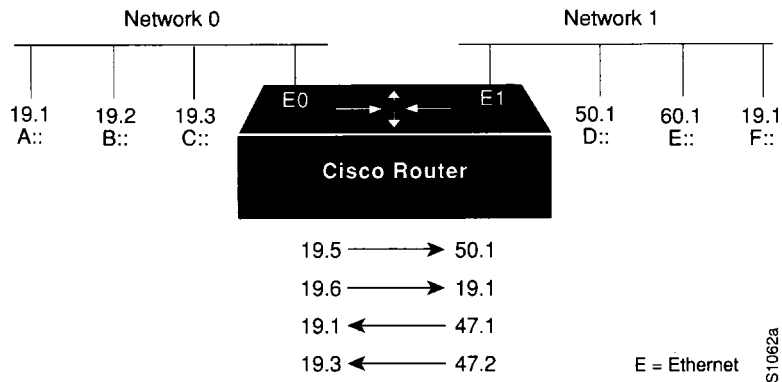


Figure 12-3 ATG Configuration Example

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 ethernet 0
decnet 0 cost 1
!
decnet 1 routing 50.5
decnet 1 node area
interface ethernet 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 12-1 defines the parameters for the packet exchange translation map.

Table 12-1 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: Cisco 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 are 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
Ethernet 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	Ethernet2 ->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 packet 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 senders’ 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 Cisco 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, routes that expire, and so on.

DECnet Global Configuration Command Summary

This section provides an alphabetically arranged summary of all the DECnet global interface commands. These commands may appear anywhere in the configuration file.

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

Creates an access. 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 lists.

[no] access-list *list* {**permit**|**deny**} *source source-mask destination destination-mask*

Creates an extended access lists. 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 an 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.

decnet encapsulation {pre-dec|dec}

The keyword **pre-dec** configures routers for operation on the same Token Ring with routers running software versions prior to 9.1, referred to as Cisco-style encapsulation. The keyword **dec** provides encapsulation that is compatible with other DEC equipment.

[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. 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.

Chapter 13

Routing IP

13

Cisco's Implementation of IP 13-1

Configuring IP 13-1

Enabling IP Routing 13-2

Assigning IP Addresses 13-2

Internet Address Notation 13-2

Address Classes and Formats 13-3

Allowable Internet Addresses 13-4

Internet Address Conventions 13-4

Subnetting and Routing 13-5

 Creating a Single Network from Separated Subnets 13-6

 Subnet Masks 13-6

Setting IP Interface Addresses 13-7

 Using Subnet Zero 13-7

Local and Network Addresses: Address Resolution 13-8

Address Resolution Using ARP 13-8

 Tailoring ARP: Static Entries and Timing 13-8

Address Resolution Using Proxy ARP 13-10

Address Resolution Using Probe 13-10

Reverse Address Resolution Using RARP and BootP 13-11

Broadcasting in the Internet 13-11

Internet Broadcast Addresses 13-12

UDP Broadcasts 13-13

Forwarding Broadcast Packets and Protocols 13-13

Flooding IP Broadcasts 13-15

Limiting Broadcast Storms 13-16

Configuring ICMP and Other IP Services 13-16

Generating Unreachable Messages 13-17

Generating Redirect Messages 13-17

Setting and Adjusting Packet Sizes 13-18

MTU Path Discovery 13-18

Using the Ping Function 13-19

Configuring Internet Header Options 13-19

Configuring IP Host Name-to-Address Conversion 13-20

- Defining Static Name-to-Address Mappings 13-20
- Configuring Dynamic Name Lookup 13-20
- HP Probe Proxy Support 13-21
- Establishing Domain Lists 13-22

Configuring IP Access Lists 13-23

- Configuring Standard Access Lists 13-24
 - Implicit Masks 13-24
- Configuring Extended Access Lists 13-25
- Controlling Line Access 13-27
- Controlling Interface Access 13-28

Configuring the IP Security Option (IPSO) 13-28

- IPSO Definitions 13-29
- Disabling IPSO 13-30
- Setting Security Classifications 13-30
- Setting a Range of Classifications 13-30
- Modifying Security Levels 13-31
 - Ignore Authority Field 13-31
 - Accept Unlabeled Datagrams 13-31
 - Accept Datagrams with Extended Security Option 13-32
 - Adding or Removing a Security Option by Default 13-32
 - Prioritizing the Presence of a Security Option 13-32
- Default Values for Minor Keywords 13-33
- IPSO Configuration Examples 13-33

Debugging IPSO 13-35

Configuring IP Accounting 13-36

- Enabling IP Accounting 13-36
- Defining Maximum Entries 13-36
- Specifying Account Filters 13-36
- Controlling the Number of Transit Records 13-37

Special IP Configurations 13-38

- Configuring Source Routing 13-38
- IP Processing on a Serial Interface 13-38
- Configuring Simplex Ethernet Interfaces 13-39
- Enabling Fast Switching 13-40
- Enabling IP Autonomous Switching 13-40
- Compressing TCP Headers 13-41

IP Configuration Examples 13-42

- Configuring Serial Interfaces 13-42
- Flooding of IP Broadcasts 13-43
- Creating a Network from Separated Subnets 13-43
- Customizing ICMP Services 13-44

- Helper Addresses 13-44
- HP Hosts on a Network Segment 13-45
- Establishing IP Domains 13-45
- Configuring Access Lists 13-45
- Configuring Extended Access Lists 13-46

Configuring SLIP for the Router 13-46

- Serial Line Internet Protocol (SLIP) 13-46
- Cisco's Implementation of SLIP 13-47
 - SLIP and Broadcasts 13-48
- Making SLIP Connections 13-48
 - Using a Dedicated SLIP Line 13-49
 - Using a Permanent SLIP Address 13-49
 - Using Dynamic SLIP Addresses 13-49
 - Using a Default SLIP Address 13-50
- Configuring SLIP 13-50
 - Disabling SLIP on the Auxiliary Port 13-51
 - Specifying a SLIP Address 13-51
 - Configuring a Dedicated SLIP Line 13-51
 - Configuring the SLIP Line in Interactive Mode 13-52
 - Configuring Dynamic Address Assignment 13-52
 - Configuring Dynamic Address Assignment with a Default Address 13-52
 - Setting the Baud Rate 13-53
 - Configuring the Hold Queue 13-53
 - Configuring the MTU Size of Internet Packets 13-53
- Specifying SLIP Access Lists 13-54
- Specifying SLIP Extended BootP Requests 13-55
- SLIP Configuration Example 13-57

Maintaining the IP Network 13-57

- Removing Dynamic Entries from the ARP Cache 13-57
- Removing Entries from the Host-Name-and-Address Cache 13-57
- Clearing the Checkpointed Database 13-57
- Removing Routes 13-58
- Maintaining SLIP 13-58

Monitoring the IP Network 13-58

- Displaying the IP Show Commands 13-58
- Displaying the ARP Cache 13-59
- Displaying IP Accounting 13-59
- Displaying Host Statistics 13-60
- Displaying the Route Cache 13-61
- Displaying Interface Statistics 13-62
- Displaying the Routing Table 13-63
- Displaying Protocol Traffic Statistics 13-64
- Monitoring TCP Header Compression 13-65

Monitoring SLIP 13-67

Displaying the Mapped Internet Address 13-67

Displaying the IP ARP Cache 13-67

Displaying SLIP Line Status 13-68

Displaying the Status of SLIP-Configured Lines 13-68

Displaying SLIP BootP Parameters 13-69

IP Ping Command 13-70

IP Trace Command 13-71

How Trace Works 13-72

Common Trace Problems 13-72

Tracing IP Routes 13-72

Debugging the IP Network 13-74

IP Global Configuration Command Summary 13-76

IP Interface Subcommand Summary 13-80

IP and SLIP Line Subcommand Summary 13-84

Chapter 13

Routing IP

13

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

- Configuring IP
- Assigning IP addresses, address resolution, and broadcast addresses
- Configuring access and security
- Configuring accounting
- Configuring the Serial Line Internet Protocol (SLIP) for the router

See Chapter 14 of this manual for information on the various routing protocols, how they have evolved, and how they are best used in complex internetworks.

Cisco's Implementation of IP

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

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

Configuring IP

The process of configuring your router for IP routing differs from the procedures for configuring other protocols in that you 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.

- Step 1:** Enter an address for the interface on which you will be routing IP using the **ip address** interface subcommand.
- Step 2:** Consider addressing options and broadcast packet handling, using commands described in the “Setting IP Interface Addresses” and “Broadcasting in the Internet” sections.
- Step 3:** Optionally, configure packet sizes and other performance parameters as well as ICMP and other IP services. Information for these tasks are in the section “Configuring ICMP and Other IP Services.”
- Step 4:** Configure access lists and other security options, if desired.
- Step 5:** Configure routing. The IP routing protocols are discussed in Chapter 14.

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 Chapters 21 and 22. 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 eight 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 13-1 illustrates the Class A address format.

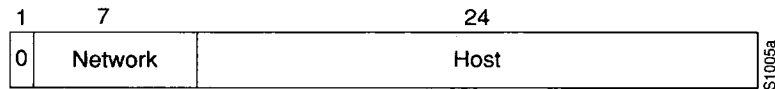


Figure 13-1 Class A Internet Address Format

The Class B Internet address format allocates the highest 16 bits to the network field and sets the two highest-order bits to 1,0. The remaining 16 bits form the host field. Over 16,000 Class B networks can exist, and each Class B network can have over 65,000 hosts (65,534).

Figure 13-2 illustrates the Class B address format.



Figure 13-2 Class B Internet Address Format

The Class C Internet address format allocates the highest 24 bits to the network field and sets the three highest-order bits to 1,1, and 0. The remaining eight bits form the host field. Over two million Class C networks can exist, and each Class C network can have up to 254 hosts. Figure 13-3 illustrates the Class C address format.

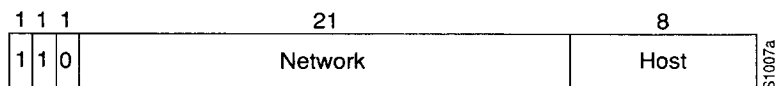


Figure 13-3 Class C Internet Address Format

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

The Class E Internet address format is reserved for future use. In Class E addresses, the four highest-order bits are set to 1,1,1, 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 13-1 lists ranges of Internet addresses and shows which addresses are reserved and which are available for use.

Table 13-1 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” later in this chapter. You also can find general information on broadcasts in Chapter 14 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” later 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 13-4 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.”

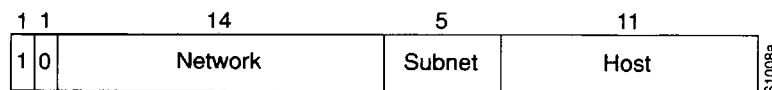


Figure 13-4 A Class B Address with a 5-Bit Subnet Field

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 13-4, the subnet mask is 255.255.248.0. Table 13-2 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 13-2 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” later 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 and is always **snap** for FDDI and token ring interfaces and **ultra** for UltraNet interfaces. See the discussions of the individual interface types for more information on possible encapsulations.

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

Example

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

```
arp 192.31.7.19 0800.0900.1834 arpa
```

The **no arp** subcommand removes the specified entry from the ARP cache. To remove all 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 four hours (14,400 seconds). A value of zero 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 12000 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” earlier in this chapter. This command is disabled by default.

Note: Cisco's support for HP Probe proxy support changed as of Software Release 8.3 (2) and subsequent software releases. The command **no arp probe** is now the default. All interfaces that will use Probe must now be explicitly configured for **arp probe**.

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.

Cisco routers support two kinds of broadcasting: *directed broadcasting* and *flooding*. A directed broadcast is a packet sent to a specific network or series of networks, while a flooded broadcast packet is sent to every network, as shown in Figure 13-5. The packet that is incoming from interface E0 is flooded to interfaces E1, E2 and Serial 0. A directed broadcast address includes the network or subnet fields.

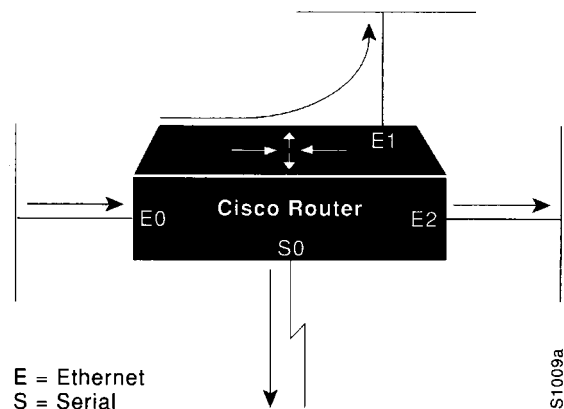


Figure 13-5 IP Flooded Broadcast

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 jumpers in the processor configuration register. Setting bit 10 causes the

router to use all zeros. Bit 10 interacts with bit 14, which controls the network and subnet portions of the broadcast address. Setting bit 14 causes the router to include the network and subnet portions of its address in the broadcast address. Table 13-3 shows the combined effect of setting bits 10 and 14.

Table 13-3 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 appropriate hardware installation and maintenance manual for your system.

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 for 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 1
 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 1
 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: The transparent bridging option must be in the routing software, and 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 Chapter 21 of this manual 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 Cisco router, 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 to deliver to its ultimate destination because it knows of no route to the destination address, it replies to the originator of that datagram with an ICMP Host Unreachable message. Use the **ip unreachable** interface subcommand to enable or disable the sending of these messages. The full syntax for this command follows.

```
ip unreachable  
no ip unreachable
```

The default is to send unreachable messages. The **no ip unreachable** 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 the sending of these messages, as follows:

```
ip redirects  
no ip redirects
```

The default is to send redirects. The **no** version disables the sending of 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  
no ip mtu
```

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 (described in Chapter 7) 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

On all Cisco routers running software Release 8.3 or later, the IP MTU Path Discovery mechanism is running by default. IP Path MTU Discovery allows a host to dynamically discover and cope with differences in the maximum allowable MTU size of the various links along the path. Sometimes a router is unable to forward a datagram because 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 13-6.

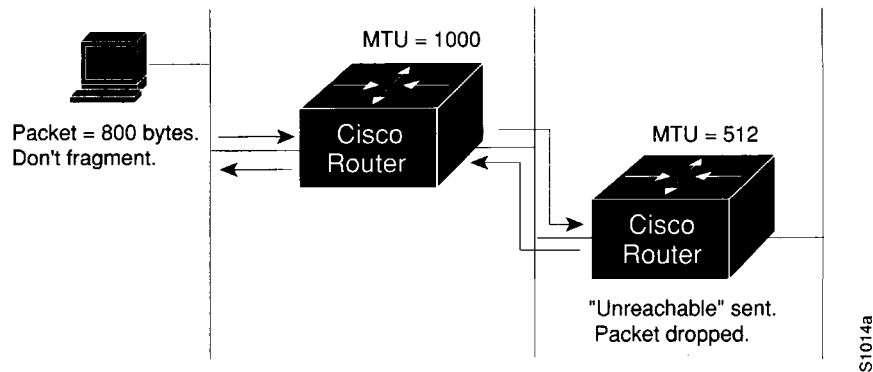


Figure 13-6 MTU Path Discovery

MTU Path Discovery is useful when a link in a network goes down, forcing use of another, different MTU-sized link (and different routers). As an example, suppose one were trying to send IP packets over a network where the MTU in the first router is set to 1500 bytes, but then reaches a router where the MTU is set to 512 bytes. If the datagram's "Don't fragment" bit is set, the datagram would be dropped because the 512-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 "The IP Ping Command" later in this chapter for more information about the use of the **ping** command.

Configuring Internet Header Options

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

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

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

Configuring IP Host Name-to-Address Conversion

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

Defining Static Name-to-Address Mappings

To define a static host name-to-address mapping in the host cache, use the **ip host** global configuration command, as 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 *cisco.com* as the default name.

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

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

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 ip name-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" later 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:

- Step 1:** Create a list.
- Step 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 to 199 distinguishes an extended access list from a standard access list. The condition keywords **permit** and **deny** determine whether the router allows or disallows a connection when a packet matches an access condition. The router stops checking the extended access list after a match occurs. All conditions must be met to make a match.

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

- ip
- tcp
- udp
- icmp

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

The argument *source* is an Internet source address in dotted-decimal format. The argument *source-mask* is a mask 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 Cisco device) 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 1

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

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

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

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

Example 2

The following example defines an 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 Cisco IPSO support addresses both the Basic and Extended security options described in a draft of the IPSO circulated by the Defense Communications Agency. This draft document is an early version of RFC 1108. The following list summarizes the differences between Cisco's implementation and RFC 1108:

- DIA Authority is equivalent to SCI Authority.
- Cisco supports SCI.
- Cisco does not support DOE Authority keyword.
- Cisco only accepts a four-byte IPSO.

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

- Defines security level on a per-interface basis.
- Defines single-level or multilevel interfaces.
- Provides a label for incoming datagrams.
- Strips labels on a per-interface basis.
- 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 13-4 lists the level keywords used by the Cisco software and their corresponding bit patterns.
- **authority**—An organization that defines the set of security levels that will be used in a network. For example, the Genser authority consists of level names defined by the Defense Communications Agency (DCA). Table 13-5 lists the authority keywords used by the Cisco software and their corresponding bit patterns.
- **label**—A combination of a security level and an authority or authorities.

Table 13-4 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 13-5 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 13-4 and Table 13-5.

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* field is the empty set, then a datagram is required to specify any one or 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; and otherwise it is *off*.

Table 13-6 provides a list of all default values.

Table 13-6 Default Security Keyword Values

Type	Level	Authority	Implicit	Add
None	None	None	On	Off
Dedicated	Unclassified	Genser	On	Off
Dedicated	Any	Any	Off	On
Multilevel	Any	Any	Off	On

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

IPSO Configuration Examples

In this first example, three Ethernet interfaces are presented. These interfaces are running at security levels of Confidential Genser, Secret Genser, and Confidential to Secret Genser, as shown in Figure 13-7.

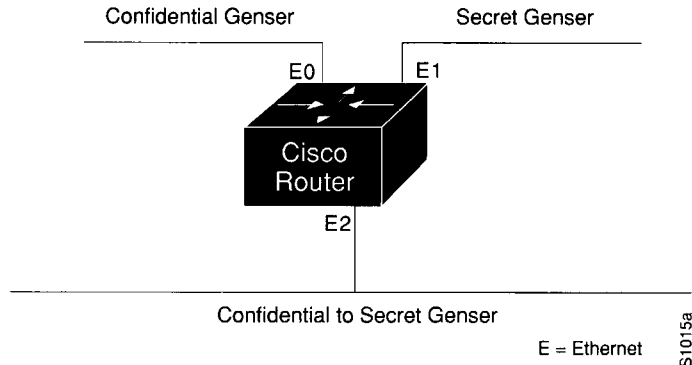


Figure 13-7 IPSO Security Levels

Example 1

The following commands set up interfaces for the configuration in Figure 13-7.

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

It is possible for the setup to be much more complex.

Example 2

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

```
interface ethernet 0
ip security dedicated confidential genser
ip security implicit-labelling
ip security strip
interface ethernet 1
ip security dedicated secret genser
ip security extended-allowed
!
interface ethernet 2
ip security multilevel confidential genser to secret genser
ip security extended-allowed
ip security first
```

Debugging IPSO

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

Security failure 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 gensec*. Then this label is compared to the interface range, and the appropriate action is taken. See Table 13-7.

Table 13-7 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” later 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.

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
ip unnumbered ethernet 0
```

Configuring Simplex Ethernet Interfaces

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

transmit-interface *interface-name*

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

Example

Figure 13-8 illustrates how to configure IP on two routers sharing transmit-only and receive-only Ethernet connections.

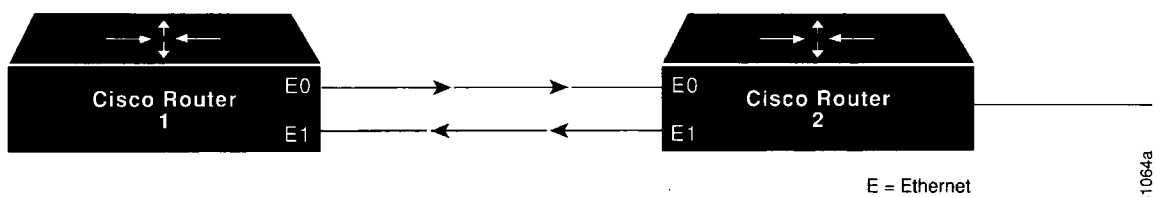


Figure 13-8 Simplex Ethernet Connections

Example for Router 1

```
interface ethernet 0
ip address 128.9.1.1
!
interface ethernet 1
ip address 128.9.1.2
```

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

Example for Router 2

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

Enabling Fast Switching

The **ip route-cache** interface subcommand controls the use of 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.

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

Enabling IP Autonomous Switching

Autonomous switching gives a router faster packet processing by allowing the cBus to switch packets independently without interrupting the system processor.

Note: Autonomous switching works only in AGS+ systems with high-speed network controller cards, such as the CSC-HSCI, CSC-MEC, CSC-FCI, and CSC-C2/FCIT, and with a cBus controller card running microcode Version 1.4 or later. (See the information on microcode revisions in the software release notes accompanying this publication for other microcode revision requirements.)

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

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

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

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

ip route-cache cbus

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

no ip route-cache

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

no ip route-cache cbus

To return to the default, use the standard **ip route-cache** command. This turns fast switching on and autonomous switching off.

ip route-cache

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]
no ip tcp header-compression [passive]

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

When compression is enabled, fast switching is disabled. 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" later 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
```

IP 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 (two Ethernet and two serial). No bridging is permitted. The access list denies all protocols. No specific UDP protocols are listed by a separate **ip forward-protocol udp** 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 ethernet 1
bridge-group 1
bridge-group 1 input-type-list 201
interface serial 0
bridge-group 1
bridge-group 1 input-type-list 201
interface serial 1
bridge-group 1
bridge-group 1 input-type-list 201
```

Creating a Network from Separated Subnets

In the following example, networks 131 and 192 are separated by a backbone, as shown in Figure 13-9. The two networks are brought into the same logical network through the use of secondary addresses.

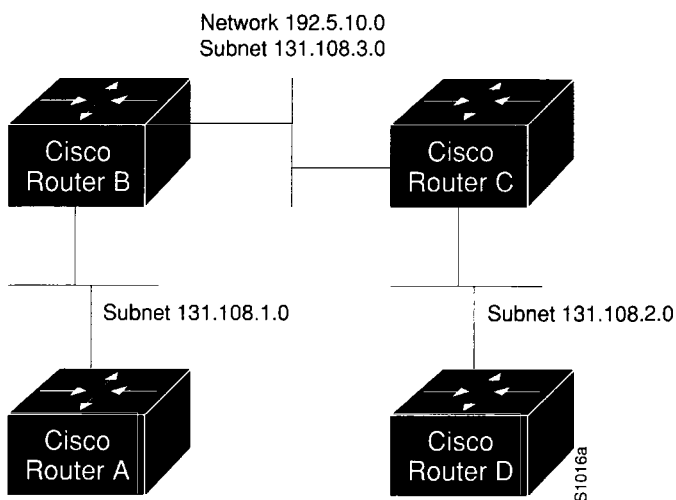


Figure 13-9 Creating a Network from Separated Subnets

Example—Router B

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

Example—Router C

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

Customizing ICMP Services

The example 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
```

Helper Addresses

In this example, one server is on network 191.24.1.0 and the other is on network 110.44.0.0, and you want to permit IP broadcasts from all hosts to reach these servers. Figure 13-10 illustrates how to configure the router that connects network 110 to network 191.

Example

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

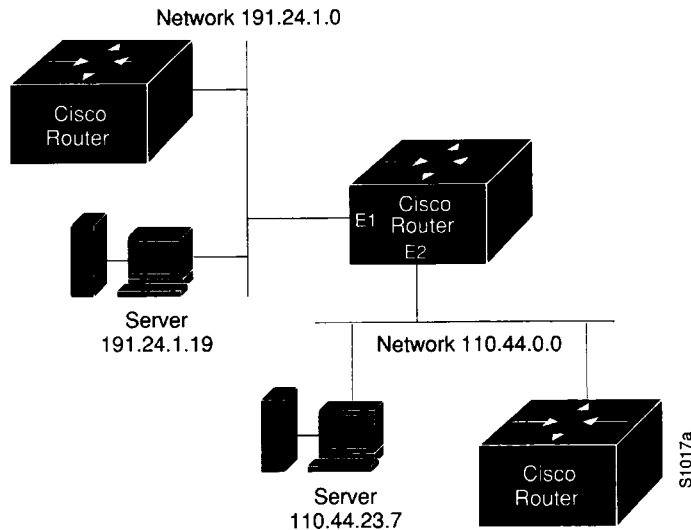


Figure 13-10 IP Helper Addresses

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

Configuring Access Lists

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

Configuring SLIP for the Router

This section describes the Serial Line Internet Protocol (SLIP) and its implementation on the router. Information provided here includes the following:

- Definition of SLIP
- Overview of Cisco's SLIP implementation for the router
- Steps in making SLIP connections
- Configuring the auxiliary line for SLIP access
- Configuring SLIP access controls
- Specifying extended Boot Protocol (BootP) requests
- Maintaining the SLIP line

Serial Line Internet Protocol (SLIP)

SLIP defines a method of sending Internet packets over standard RS-232 asynchronous serial lines. It is a de facto standard used for point-to-point serial connections running TCP/IP. SLIP is commonly used on dedicated serial links with line speeds between 1,200 and 19,200 bps. It allows mixes of hosts and routers to communicate with one another, so that host-to-host, host-to-router, and router-to-router are all common SLIP network configurations.

The version of SLIP described in this manual was originally implemented by researchers at the University of California at Berkeley in their 4.2 BSD version of the UNIX operating system. Although variants have been proposed, the Berkeley version has emerged as a de facto standard. Refer to RFC 1055 for more information about SLIP.

Cisco's Implementation of SLIP

Cisco has provided an implementation of SLIP over the auxiliary port (asynchronous serial line) of its chassis-based router products. Its intended use is to provide access by a network management workstation to a router in a network where one or more routers are inaccessible.

For example, Figure 13-11 illustrates a workstation and a portion of a network with three routers, A, B, and C. If Router A or B should go offline, the network management workstation would not be able to monitor events between itself and Router C, or even reach Router C. However, by dialing in to the modem connected to the auxiliary port on Router C and enabling SLIP, the workstation can run SNMP transparently and continue its job of failure isolation from both sides.

Note: SLIP is supported on all asynchronous serial ports except the console line.

You need to order a special cable from Cisco Systems for connections to the auxiliary port; see the section “Configuring Console and Virtual Terminal Lines” in Chapter 4 for more information.

In addition to implementing the de facto standard, Cisco's implementation of SLIP offers both dedicated and dynamic address assignment, configurable hold queue and IP packet sizes, extended BootP requests, and permit/deny conditions for controlling access to the line.

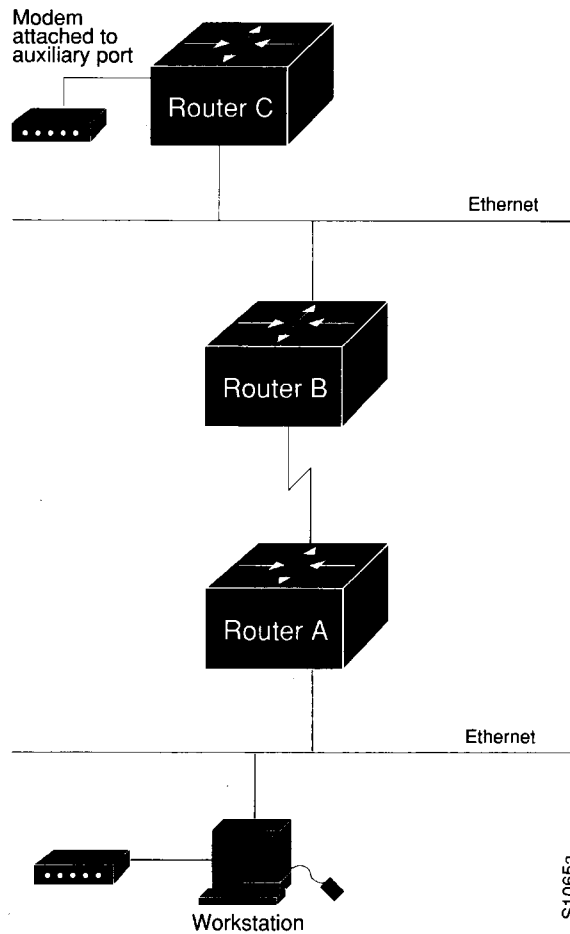


Figure 13-11 Sample SLIP Configuration Using the Auxiliary Port

SLIP and Broadcasts

Cisco routers recognize a variety of Internet broadcast addresses. When a router receives an Internet packet with one of these addresses from a SLIP client, it rebroadcasts the packet onto the network without changing the Internet header. The router does not alter the packet's broadcast address to match the form of broadcast address it prefers.

The router receives a copy of SLIP client broadcasts, and responds to BootP requests with the Internet address of the line that received them. This facility allows the SLIP client software to automatically determine its own Internet address.

Making SLIP Connections

Use one of two EXEC commands to make a SLIP connection, depending on how your line is set up:

```
slip
slip default
```

It also is possible to configure a dedicated SLIP line, in which case no EXEC command is required to make the connection.

The following paragraphs describe the ways in which the SLIP lines can be configured and which command to use to make the connection.

Using a Dedicated SLIP Line

A line can be permanently configured for SLIP using the **slip dedicated** and **slip address** line configuration commands. In this case, an EXEC is not started on the line, so users need not enter a command to initiate SLIP. They need only to make the physical connection to the line and then are immediately placed in SLIP mode. No prompt or status message is issued.

Using a Permanent SLIP Address

A line can be assigned a permanent SLIP address with the **slip address** line configuration command. In this case, the user issues the **slip EXEC** command to put the line into SLIP mode. If the server has been configured to authenticate SLIP connections, the user is prompted for a password before the line is placed in SLIP mode.

Sample Session

This sample illustrates how to enter the **slip EXEC** command to make a connection when a permanent SLIP address has been assigned. Once a correct password is entered, SLIP mode is entered, and the IP address is displayed.

```
Router> slip  
  
Password:  
Entering SLIP mode.  
Your IP address is 192.31.7.28, MTU is 1524 bytes
```

Using Dynamic SLIP Addresses

A line can be configured for dynamic assignment of SLIP addresses with the **slip address dynamic** command. In this case, the user enters the **slip EXEC** command and is prompted for the IP address or a logical host name to use. This address is validated via TACACS (when enabled), and the line is put into SLIP mode using the address requested.

This feature is useful in a situation where the user needs to know the IP address of a line. A personal computer running an application that automatically dials in using SLIP and polls for electronic mail messages would be an example of this. The application can be set up to dial in periodically and enter the required IP address and password.

Sample Session

This sample illustrates the “IP address or hostname?” prompt displayed and the response required when dynamic addressing is used to assign the SLIP address.

```
Router> slip

IP address or hostname? 192.31.6.15
Password:
Entering SLIP mode
Your IP address is 192.31.6.15, MTU is 1524 bytes
```

Using a Default SLIP Address

A line also can be given a default address to use. In this case, the user issues the **slip default EXEC** command, the transaction is validated by the TACACS server (when enabled), and the line is put into SLIP mode using the address configured with the IP address argument of the **slip address dynamic** configuration command.

This feature is useful when it is not reasonable for all users to know the IP address needed to gain access to a system. A server that is available to many students on a campus would be an example of this. Instead of requiring each to know an IP address, they need only enter the EXEC **slip default** command and let the server select the line to use.

Sample Session

In this sample session, the address 192.31.6.15 has been assigned as the default. Password verification still is required before SLIP mode can be enabled.

```
Router> slip default

Password:
Entering SLIP mode
Your IP address is 192.31.6.15, MTU is 1524 bytes
```

Note: When TACACS authentication of SLIP addresses is used, the configuration command **tacacs-server optional-passwords** can be used to suppress the password prompt if your TACACS server supports validation of addresses without passwords. See “Configuring the System” in Chapter 4 for information about TACACS verification.

Configuring SLIP

Following are the basic steps for configuring SLIP on the router.

- Step 1:** Enable SLIP on the auxiliary port using the **slip dedicated**, **slip address**, or **slip address dynamic** line subcommands. See the section “Making SLIP Connections” earlier in this chapter for descriptions of the different types of SLIP connections.
- Step 2:** Make the appropriate settings for the line. Line settings include baud rate, addressing, and packet size limits.

Step 3: Specify access lists for control of traffic to or from the SLIP-enabled auxiliary port, if needed.

Step 4: Specify extended BootP requests, if needed.

The following sections describe how to configure the router. The EXEC commands used to monitor and maintain a SLIP link are described at the end of this chapter.

Disabling SLIP on the Auxiliary Port

Once a line is configured for SLIP, the EXEC responds to the **slip** or **slip default** EXEC commands by turning on SLIP, displaying the Internet address and the size of the largest Internet packet the SLIP support can handle. The line exits SLIP mode when the modem is hung up or a **clear line** command is issued.

The **no slip** line subcommand disables SLIP mode. It has this syntax:

```
no slip
```

Use this command to disable SLIP on the auxiliary port that previously had SLIP enabled.

Specifying a SLIP Address

The **slip address** line subcommand specifies the Internet address assigned to the SLIP client at the other end of the serial line connection. The command has this syntax:

```
slip address internet-address
```

The argument *internet-address* must be on the same network or subnet as one of the router's network interfaces.

To put the auxiliary port into SLIP mode, use the EXEC command **slip**; see “Making SLIP Connections” earlier in this chapter.

Example

This example sets IP address 182.32.7.51 on the auxiliary line.

```
line aux 0  
slip address 182.32.7.51
```

Configuring a Dedicated SLIP Line

The **slip dedicated** line subcommand puts the auxiliary port in SLIP mode permanently. It has this simple syntax:

```
slip dedicated
```

The router will not create an EXEC on this port, so it is not available for normal interactive use. No **slip** EXEC command is necessary to enable SLIP mode.

Example

The **slip dedicated** command permanently places the auxiliary port into SLIP mode.

```
line aux 0
slip address 182.32.7.5
slip dedicated
```

Configuring the SLIP Line in Interactive Mode

The **slip interactive** line subcommand allows the port to be used in either SLIP mode or interactive mode.

slip interactive

The **slip interactive** subcommand generally is used to void a **slip dedicated** line subcommand. It is the equivalent in function to a **no slip dedicated** command (although there is no such command). To put the port into SLIP mode, use the EXEC command **slip**; see “Making SLIP Connections” earlier in this chapter.

Hanging up the modem or clearing the port puts it back into interactive mode.

Configuring Dynamic Address Assignment

Dynamic address assignment requires that the user enter an IP address to enter SLIP mode. The full syntax of the line subcommand to set this up is as follows:

slip address dynamic

To put the port into SLIP mode, use the EXEC command **slip**.

This feature is supported when a TACACS server is used. The host name sent in a TACACS request will be in all uppercase letters.

Example

In the following example, the auxiliary port is configured for dynamic address assignment.

```
line aux 0
slip address dynamic
```

Configuring Dynamic Address Assignment with a Default Address

If a line is configured for dynamic address assignment, it can also be given a default address to use. With this configuration, the user enters the EXEC command **slip default** to make connection. The syntax for this is as follows:

slip address dynamic IP-address

The argument *IP-address* is the IP address to use.

Example

This example illustrates how to enter a default SLIP address.

```
line aux 0
slip address dynamic 108.91.3.4
```

Setting the Baud Rate

Use the **speed** line subcommand to set the transmit and receive speeds for the SLIP line.

speed *baud*

The argument *baud* can be 100, 1200, 2400, 4800, 9600, 19200, or 38400. Whether or not a higher baud rate improves performance depends on the SLIP client's ability to handle the interrupt load. The default is 9600 baud.

Example

This example sets the baud rate to 9600.

```
line aux 0
slip address 182.32.7.5
speed 9600
```

Configuring the Hold Queue

The **slip hold-queue** line subcommand specifies the limit of the SLIP output queue, which stores packets received from the network waiting to be sent to the SLIP client. The syntax is as follows:

slip hold-queue *packets*

The argument *packets* is the maximum number of packets. The default is three packets; it is recommended that the queue size not exceed 10.

Example

This example changes the packet queue length to five packets.

```
line aux 0
slip address 182.32.7.5
slip hold-queue 5
```

Configuring the MTU Size of Internet Packets

The **slip mtu** line subcommand specifies the size of the largest Internet packet that the SLIP support can handle.

slip mtu *bytes*

The argument *bytes* is the maximum number of bytes. The default is 1500 MTU.

You might want to change to a smaller MTU size if the SLIP application at the other end does not support packets of that size or you want to assure a lower delay by using shorter packets. This can be desirable when the host Telnet echoing takes longer than 0.2 second. For instance, at 9600 baud, a 1500-byte packet takes about 1.5 seconds to transmit, so this delay would indicate that you want an MTU size of about 200.

On the other hand, the MTU size can be negotiated by TCP, regardless of what the terminal settings are; this method is preferable. The router performs IP fragmentation of packets larger than the specified MTU. Therefore, do not use this command unless the SLIP implementation supports reassembly of IP fragments. Since each fragment occupies a spot in the output queue, it also may be necessary to increase the size of the SLIP hold queue.

Example

This example sets the packet MTU size to 200 bytes.

```
line aux 0
slip address 182.32.7.5
slip mtu 200
```

Specifying SLIP Access Lists

Access lists allow the system administrator to control the hosts that can access a router. Because SLIP is different than a connection, separate access lists can be defined for SLIP and for normal connections. The software allows separate access lists to be defined for use when the line is running SLIP.

To configure an access list to be used on packets *from* the SLIP host, use this line subcommand:

slip access-class *number* in

When this command is entered, the IP destination address of each packet is run through the access list for acceptability and then dropped or passed. The argument *number* is the IP access list number (see the section “Configuring IP Access Lists” earlier in this chapter for information about IP access lists).

To specify an access list to be used on packets being sent *to* the SLIP host, use this line subcommand:

slip access-class *number* out

When this command is entered, the IP source address is compared against the access list, and only those packets allowed by the access list are transmitted on the asynchronous line. The argument *number* is the IP access list number (see the section “Configuring IP Access Lists” earlier in this chapter for more information about IP access lists).

Example

This example assumes that SLIP users are restricted to certain devices designated as SLIP servers, but that other users can access anything on the local network.

```

! access list for normal connections
access-list 1 permit 131.108.0.0 0.0.255.255
!
! access list for SLIP packets.
access-list 2 permit 131.108.42.55
access-list 2 permit 131.108.111.1
access-list 2 permit 131.108.55.99
!
!Define the line with the SLIP address and appropriate access list
line aux 0
slip address dynamic
access-class 1 out
slip access-class 2 in
!

```

Specifying SLIP Extended BootP Requests

The Boot Protocol (BootP) server for SLIP supports the extended BootP requests specified in RFC 1084. These requests are specified with the **async-bootp** global configuration command. The full syntax for this command follows:

```

async-bootp tag [:hostname] data ...
no async-bootp

```

The argument *tag* is the item being requested and is one of the following expressed as file name, integer, or IP dotted decimal address:

- **bootfile**—Specifies use of a server boot file from which to download the boot program. Use the optional *:hostname* and *data* arguments to specify the file name.
- **subnet-mask** *mask*—Dotted decimal address specifying the network and local subnet-mask (as defined by RFC 950).
- **time-offset** *offset*—A signed 32-bit integer specifying the time offset of the local sub-network in seconds from Universal Coordinated Time (UTC).
- **gateway** *address*—Dotted decimal address specifying the IP addresses of gateways for this subnetwork. A preferred gateway should be listed first.
- **time-server** *address*—Dotted decimal address specifying the IP address of time servers (as defined by RFC 868).
- **IEN116-server** *address*—Dotted decimal address specifying the IP address of name servers (as defined by IEN 116).
- **DNS-server** *address*—Dotted decimal address specifying the IP address of Domain Name Servers (as defined by RFC 1034).
- **log-server** *address*—Dotted decimal address specifying the IP address of an MIT-LCS UDP log server.
- **quote-server** *address*—Dotted decimal address specifying the IP address of Quote of the Day servers (as defined in RFC 865).
- **lpr-server** *address*—Dotted decimal address specifying the IP address of Berkeley UNIX Version 4 BSD servers.

- **impress-server** *address*—Dotted decimal address specifying the IP address of Impress network image servers.
- **rlp-server** *address*—Dotted decimal address specifying the IP address of Resource Location Protocol (RLP) servers (as defined in RFC 887).
- **hostname** *name*—The name of the client, which may or may not be domain qualified, depending upon the site.
- **bootfile-size** *value*—A two-octet value specifying the number of 512 octet (byte) blocks in the default boot file.

Use the optional argument *:hostname* to indicate that this entry applies only to the host specified. The argument *:hostname* accepts both an IP address and logical host name.

The argument *data* can be a list of IP addresses entered in dotted decimal notation or as logical host names, a number, or a quoted string.

If no extended BootP commands are entered, by default the software generates a gateway and subnet mask appropriate for the local network.

Use the EXEC command **show async-bootp** to list the configured parameters. Use the **no async-bootp** command to clear the list.

Example 1

This example illustrates how to specify different boot files: one for a PC, and one for a Macintosh.

```
async-bootp bootfile :128.128.1.1 "pcboot"
async-bootp bootfile :mac "macboot"
```

With this configuration, a BootP request from the host on 128.128.1.1 results in a reply listing the boot file name as *pcboot*. A BootP request from the host named *mac* results in a reply listing the boot file name as *macboot*.

Example 2

This example specifies a subnet mask of 255.255.0.0.

```
async-bootp subnet-mask 255.255.0.0
```

Example 3

This example specifies a negative time offset of the local subnetwork of -3600 seconds.

```
async-bootp time-offset -3600
```

Example 4

This example specifies the IP address of a time server.

```
async-bootp time-server 128.128.1.1
```

SLIP Configuration Example

The following configuration example assigns an Internet address to a SLIP line and puts the line in SLIP mode permanently.

```
!  
line aux 0  
location auxiliary port  
speed 9600  
slip address 182.32.7.51  
slip dedicated  
!
```

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.

Maintaining SLIP

This section describes the EXEC commands for maintaining SLIP support on the router.

The **clear line** EXEC command disables SLIP mode and starts an EXEC on a nondedicated SLIP line. Enter this command at the EXEC prompt:

```
clear line line-number
```

The argument *line-number* specifies the line. This command is the only way to exit SLIP mode on a line without modem control.

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 13-8 describes the fields seen.

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	131.108.62.192	187	0800.2010.a3b6	ARPA	Ethernet3
Internet	131.108.62.245	68	0800.200e.28f8	ARPA	Ethernet3
Internet	131.108.1.140	139	0000.0c01.2812	ARPA	Ethernet0
Internet	131.108.62.160	187	0800.200e.4dab	ARPA	Ethernet3
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	Ethernet2
Internet	192.31.7.17	118	2424.c01f.0711	ARPA	Ethernet2
Internet	192.31.7.18	135	0000.0c01.2817	ARPA	Ethernet2
Internet	192.31.7.21	119	2424.c01f.0715	ARPA	Ethernet2
Internet	131.108.1.33	1	0800.2008.c52e	ARPA	Ethernet0
Internet	131.108.62.1	-	0000.0c00.750f	ARPA	Ethernet3
Internet	131.108.31.35	119	0800.2010.8c5b	ARPA	Ethernet7
Internet	131.108.62.7	14	0000.0c00.33ce	ARPA	Ethernet3
Internet	131.108.1.55	155	0800.200e.e443	ARPA	Ethernet0

Table 13-8 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 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 CISCO.COM
Name/address lookup uses domain service
Name servers are 255.255.255.255
Host                Flags      Age Type  Address(es)
SLAG.CISCO.COM      (temp, OK) 1  IP    131.108.4.10
CHAR.CISCO.COM      (temp, OK) 8  IP    192.31.7.50
CHAOS.CISCO.COM     (temp, OK) 8  IP    131.108.1.115
DIRT.CISCO.COM      (temp, EX) 8  IP    131.108.1.111
DUSTBIN.CISCO.COM   (temp, EX) 0  IP    131.108.1.27
DREGS.CISCO.COM     (temp, EX) 24 IP    131.108.1.30
```

In the display:

- A `temp` entry in the `Flags` field is entered by a name server; the router removes the entry after 72 hours of inactivity.
- A `perm` entry is entered by a configuration command and is not timed out. Entries marked `OK` are believed to be valid. Entries marked `??` are considered suspect and subject to revalidation. Entries marked `EX` are expired.
- The `Age` field indicates the number of hours since the router last referred to the cache entry.
- The `Type` field identifies the type of address, for example, `IP`, `CLNS`, or `X.121`. 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
                                40300000300010000000000000000000AAAA030000000800
```

In the display:

- The * designates valid routes.
- The Destination field shows the destination IP address.
- The Interface field specifies the interface type and number (serial 1, Ethernet 2, 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 13-9 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 unreachable 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 13-9 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 unreachable	Specifies whether unreachable messages will be sent on this interface.
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, Ethernet1
```

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

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

Displaying Protocol Traffic Statistics

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

```
show ip traffic
```

Following is sample output:

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

```

    0 parameter, 0 timestamp, 0 info request, 0 other
Sent: 0 redirects, 3 unreachable, 0 echo, 0 echo reply
    0 mask requests, 0 mask replies, 0 quench, 0 timestamp
    0 info reply, 0 time exceeded, 0 parameter problem
UDP statistics:
  Rcvd: 56 total, 0 checksum errors, 55 no port
  Sent: 18 total, 0 forwarded broadcasts
TCP statistics:
  Rcvd: 0 total, 0 checksum errors, 0 no port
  Sent: 0 total
EGP statistics:
  Rcvd: 0 total, 0 format errors, 0 checksum errors, 0 no listener
  Sent: 0 total
IGRP statistics:
  Rcvd: 73 total, 0 checksum errors
  Sent: 26 total
HELLO statistics:
  Rcvd: 0 total, 0 checksum errors
  Sent: 0 total
ARP statistics:
  Rcvd: 20 requests, 17 replies, 0 reverse, 0 other
  Sent: 0 requests, 9 replies (0 proxy), 0 reverse
Probe statistics:
  Rcvd: 6 address requests, 0 address replies
    0 proxy name requests, 0 other
  Sent: 0 address requests, 4 address replies (0 proxy)
    0 proxy name replies

```

In the display:

- A format error is a gross error in the packet format, such as an impossible Internet header length.
- A bad hop count occurs when a packet is discarded because its time-to-live (TTL) field was decremented to zero.
- An encapsulation failure usually indicates that the router 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 13-10 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 13-10 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.
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.

Monitoring SLIP

Use the EXEC **show** commands described in this section to obtain displays of activity on the SLIP line.

Displaying the Mapped Internet Address

The **show ip aliases** command displays Internet addresses mapped to TCP ports (*aliases*) and SLIP addresses, which are treated similarly to aliases. Enter this command at the EXEC prompt:

show ip aliases

To distinguish a SLIP address from a normal alias address, the command output uses the form SLIP TTY1 for the “port” number, where 1 is the auxiliary port.

Sample output follows:

```
IP Address      Port
131.108.29.245  SLIP TTY1
```

The display lists the IP address and corresponding port number.

Displaying the IP ARP Cache

The **show ip arp** EXEC command displays the Address Resolution Protocol (ARP) cache, where SLIP addresses appear as permanent ARP table entries. To display the IP ARP cache, use the following EXEC command:

show ip arp

An Address Resolution Protocol establishes correspondences between network addresses (an IP address, for example) and LAN hardware addresses (MAC 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 13-11 describes the fields displayed.

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	131.108.29.246	-	0000.0c00.6a19	ARPA	Ethernet0
Internet	131.108.29.245	-	0000.0c00.6a19	ARPA	Ethernet0
Internet	131.108.29.6	0	0000.0c01.0ec3	ARPA	Ethernet0

Table 13-11 Show IP ARP Display Field Descriptions

Field	Description
Protocol	Protocol for network address in Address field
Address	The network address that corresponds to Hardware Addr
Age (min)	Age, in minutes, of the last update of the cache entry
Hardware Addr	LAN hardware (or MAC) address that corresponds to network address
Type	Type of encapsulation: ARPA = Ethernet SNAP = RFC 1042 ISO1 = IEEE 802.3

Displaying SLIP Line Status

The **show line EXEC** command displays SLIP status for a line running in SLIP mode. Enter this command at the EXEC prompt:

```
show line line-number
```

The argument *line-number* specifies the line.

Following is sample output:

Tty	Typ	Tx/Rx	A	Modem	Roty	AccO	AccI	Uses	Noise	OVERRUNS
0	CTY		-	-	-	-	-	0	1	0
S 1	AUX	9600/9600	-	-	-	-	-	0	0	0
* 2	VTY	9600/9600	-	-	-	-	-	4	0	0
* 3	VTY	9600/9600	-	-	-	-	-	1	0	0
4	VTY	9600/9600	-	-	-	-	-	0	0	0
5	VTY	9600/9600	-	-	-	-	-	0	0	0
6	VTY	9600/9600	-	-	-	-	-	0	0	0

Displaying the Status of SLIP-Configured Lines

The **show slip EXEC** command displays the status of all lines configured for SLIP support. Enter this command at the EXEC prompt:

```
show slip
```

Following is sample output:

```
Slip statistics:
  Rcvd: 465 packets, 0 bytes, 27 escapes
        0 format errors, 0 checksum errors, 0 overrun, 0 no buffer
  Sent: 316 packets, 0 bytes, 27 escapes, 45 dropped

  Tty Mod  Address      Istate  Ostate  Qd InPack OutPac Inerr Dropped MTU Qsz
  * 1 - 131.108.29.245 RECV   IDLE    0 465 316 0 0 1524 2
```

Table 13-12 describes the fields displayed by this command.

Table 13-12 SLIP Statistics Display Field Descriptions

Field	Description
Rcvd:	Statistics on packets received
packets	Packets received
bytes	Total number of bytes
escapes	Count of escape characters received
format errors	Packets with a bad IP header, even before the checksum is calculated
checksum errors	Count of checksum errors
overrun	Number of giants received
no buffer	Number of packets received when no buffer was available
Sent	Statistics on packets sent
packets	Packets sent
bytes	Total number of bytes
escapes	Count of escape characters sent
dropped	Number of packets dropped
Tty mod	Type of modem control
IState and OState	Used by Cisco engineers for troubleshooting
Qd	Number of packets on hold queue (Qsz is max)
InPack	Number of packets input for asynchronous line
OutPac	Number of packets sent to asynchronous line
Inerr	Number of total input errors; sum of format errors, checksum errors, overruns and no buffers
Dropped	Number of packets received that would not fit on the hold queue
*	A line currently in SLIP mode

Displaying SLIP BootP Parameters

The **show async-bootp** EXEC command displays the parameters that have been configured for SLIP extended BootP requests. Enter this command at the EXEC prompt:

```
show async-bootp
```

Following is sample output:

```
The following extended data will be sent in BOOTP responses:
```

```
bootfile (for address 128.128.1.1) "pcboot"  
bootfile (for address 131.108.1.111) "dirtboot"  
subnet-mask 255.255.0.0  
time-offset -3600  
time-server 128.128.1.1
```

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

The **ping** command uses the notation shown in Table 13-13 to indicate the responses it sees.

Table 13-13 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 13-14 describes the output from the trace test.

Table 13-14 Trace Test Characters

Char	Meaning
<i>nn msec</i>	For each node, the round-trip time (in <i>nn</i> milliseconds) for the specified number of probes
*	The probe timed out.
?	Unknown packet type.
Q	Source quench.
P	Protocol unreachable.
N	Network unreachable.
U	Host unreachable.

Sample Session 1

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

```
chaos# trace ABA.NYC.mil
Type escape sequence to abort.
Tracing the route to ABA.NYC.mil (26.0.0.73)
 0 DEBRIS.CISCO.COM (131.108.1.6) 1000 msec 8 msec 4 msec
 1 BARRNET-GW.CISCO.COM (131.108.16.2) 8 msec 8 msec 8 msec
 2 EXTERNAL-A-GATEWAY.STANFORD.EDU (192.42.110.225) 8 msec 4 msec 4 msec
 3 BB2.SU.BARRNET.NET (131.119.254.6) 8 msec 8 msec 8 msec
 4 SU.ARC.BARRNET.NET (131.119.3.8) 12 msec 12 msec 8 msec
 5 MOFFETT-FLD-MB.in.MIL (192.52.195.1) 216 msec 120 msec 132 msec
 6 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)
 0 DEBRIS.CISCO.COM (131.108.1.6) 1000 msec 4 msec 4 msec
 1 BARRNET-GW.CISCO.COM (131.108.16.2) 16 msec 4 msec 4 msec
 2 EXTERNAL-A-GATEWAY.STANFORD.EDU (192.42.110.225) 16 msec 4 msec 4 msec
 3 NSS13.BARRNET.NET (131.119.254.240) 112 msec 8 msec 8 msec
 4 SALT_LAKE_CITY.UT.NSS.NSF.NET (129.140.79.13) 72 msec 64 msec 72 msec
 5 ANN_ARBOR.MI.NSS.NSF.NET (129.140.81.15) 124 msec 124 msec 140 msec
 6 PRINCETON.NJ.NSS.NSF.NET (129.140.72.17) 164 msec 164 msec 172 msec
 7 ZAPHOD-GATEWAY.JVNC.NET (128.121.54.72) 172 msec 172 msec 180 msec
 8 HOTBLACK-GATEWAY.JVNC.NET (130.94.0.78) 180 msec 192 msec 176 msec
 9 CAPITAL1-GATEWAY.JVNC.NET (130.94.1.9) 280 msec 192 msec 176 msec
10 CHEESESTEAK2-GATEWAY.JVNC.NET (130.94.33.250) 284 msec 216 msec 200 msec
11 CHEESESTEAK1-GATEWAY.JVNC.NET (130.94.32.1) 268 msec 180 msec 176 msec
12 BEANTOWN2-GATEWAY.JVNC.NET (130.94.27.250) 300 msec 188 msec 188 msec
13 NEAR-GATEWAY.JVNC.NET (130.94.27.10) 288 msec 188 msec 200 msec
14 IHTFP.MIT.EDU (192.54.222.1) 200 msec 208 msec 196 msec
15 E40-03GW.MIT.EDU (18.68.0.11) 196 msec 200 msec 204 msec
16 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 and lines in SLIP line mode. For each **debug** command, there is an corresponding **undebug** command that turns the display off. In general, you need use these commands only during troubleshooting sessions with Cisco personnel, 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**.

debug slip

The **debug slip** EXEC command enables logging of all SLIP activity. The high volume of SLIP debugging output, which amounts to several lines per packet, noticeably affects overall system performance.

debug slip-event

The **debug slip-event** EXEC command enables logging of selected SLIP events, such as various types of errors, enabling and disabling of SLIP mode on a line, and so on. The volume of output for this command is much lower than that for the **debug slip** command.

Note: For complete SLIP debugging output, use both the **debug slip** and the **debug slip-event** commands.

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; **snap** for FDDI and Token Ring interfaces; or **ultra** for the UltraNet interfaces. The optional keyword **alias** indicates that the router should respond to ARP requests as if it were the owner of the specified IP address. The **no** version removes the specified entry from the ARP cache.

[no] async-bootp *tag [:hostname] data ...*

Specifies extended BootP requests defined in RFC 1084 (used in configuring the router for SLIP support). The argument *tag* is the item being requested, and is one of the following expressed as file name, integer, or IP dotted decimal address:

- **bootfile**—Specifies use of a server boot file from which to download the boot program. Use the optional *:hostname* and *data* arguments to specify the file name.
- **subnet-mask** *mask*—Dotted decimal address specifying the network and local sub-network mask (as defined by RFC 950).
- **time-offset** *offset*—A signed 32-bit integer specifying the time offset of the local subnetwork in seconds from Universal Coordinated Time (UTC).
- **gateway** *address*—Dotted decimal address specifying the IP addresses of N/4 gateways for this subnetwork. A preferred gateway should be listed first.
- **time-server** *address*—Dotted decimal address specifying the IP address of N/4 time servers (as defined by RFC 868).
- **IEN116-server** *address*—Dotted decimal address specifying the IP address of N/4 name servers (as defined by IEN 116).
- **DNS-server** *address*—Dotted decimal address specifying the IP address of N/4 Domain Name Servers (as defined by RFC 1034).
- **log-server** *address*—Dotted decimal address specifying the IP address of N/4 MIT-LCS UDP log server.
- **quote-server** *address*—Dotted decimal address specifying the IP address of N/4 Quote of the Day servers (as defined in RFC 865).
- **lpr-server** *address*—Dotted decimal address specifying the IP address of N/4 Berkeley UNIX Version 4 BSD servers.
- **impress-server** *address*—Dotted decimal address specifying the IP address of N/4 Impress network image servers.

- **rlp-server address**—Dotted decimal address specifying the IP address of N/4 Resource Location Protocol (RLP) servers (as defined in RFC 887).
- **hostname name**—The name of the client, (which may or may not be domain qualified, depending upon the site).
- **bootfile-size value**—A two-octet value specifying the number of 512 octet (byte) blocks in the default boot file.

The optional argument *:hostname* indicates that this entry applies only to the host specified. The argument *:hostname* accepts both an IP address and logical host name. The argument *data* can be a list of IP addresses entered in dotted decimal notation or as logical host names, a number, or a quoted string. Use the **no async-bootp** command to clear the list.

[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, FDDI, and Token Ring 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 13-4 and Table 13-5 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 13-4 and Table 13-5 for the *level* and *authority* arguments.

[no] ip security strip

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

[no] ip tcp compression-connections *number*

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

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

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

[no] ip unnumbered *interface-name*

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

[no] ip unreachable

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*.

IP and SLIP Line Subcommand Summary

Following is an alphabetically arranged list of SLIP line configuration subcommands and line configuration subcommands used to configure IP routing.

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

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

no slip

Cancels SLIP support on the line.

slip access-class *number* {**in** | **out**}

Configures an access list to be used on packets to or from the SLIP host. The argument *number* is the IP access list number. Keyword **in** configures list for packets from the SLIP host; keyword **out** compares the IP source address against the access list, and only those packets allowed by the access list are transmitted on the asynchronous line. The **no** version removes the specified list.

slip address *internet-address*

Specifies the Internet address assigned to the SLIP client at the other end of the serial line connection. The argument *internet-address* must be on the same network or subnet as the router's network interface.

slip address dynamic [*IP-address*]

Without an IP address, allows the IP address associated with a SLIP line to be assigned upon access. This feature is supported when a TACACS server is used.

With an IP address, allows a default address to be specified upon access.

slip dedicated

Places the line in SLIP mode permanently. The router will not create an EXEC on this line, so it is not available for normal interactive use.

slip hold-queue *packets*

Specifies the limit of the SLIP output queue, which stores packets received from the network waiting to be sent to the SLIP client. The argument *packets* is the maximum number of packets. Default is two packets.

slip interactive

Allows the line to be used in either SLIP mode or interactive mode. Hanging up the modem or clearing the line puts the line back into interactive mode.

slip mtu *bytes*

Specifies the size of the largest Internet packet that the SLIP support can handle. The argument *bytes* is the maximum number of bytes. Default is 1500 bytes.

speed *baud*

Sets the transmit and receive speeds for the line. The argument *baud* is 100, 1200, 2400, 4800, 9600, 19200, or 38400. Whether or not a higher baud rate improves performance depends on the SLIP client's ability to handle the interrupt load. The default is 9600.

Chapter 14

The IP Routing Protocols

14

Cisco-Supported Routing Protocols 14-1

- Interior and Exterior Protocols ICMP 14-2
- Autonomous Systems 14-2
- Multiple Routing Protocols 14-3
- Multiple IP Routing Processes 14-3

Configuration Overview 14-4

- Configuring the Interior Routing Protocols 14-4
- Configuring the Exterior Routing Protocols 14-4

Configuring the IGRP Protocol 14-5

- Interior, System, and Exterior Routes 14-5
- Creating the IGRP Routing Process 14-5
- Unequal-Cost Load Balancing 14-6
 - IGRP Variance Command 14-7
- Choosing the Gateway of Last Resort 14-8
- IGRP Metric Information 14-8
- IGRP Updates 14-8

Configuring the OSPF Routing Protocol 14-8

- The OSPF Routing Protocol 14-9
- The OSPF Routing Domain and Areas 14-9
 - OSPF Backbones 14-10
 - OSPF Router Classifications 14-10
- OSPF Routing Conventions 14-11
 - OSPF Physical Network Support 14-11
 - Support for IP Subnetting with OSPF 14-12
 - Intra-Area Routing 14-12
 - Interarea Routing 14-12
 - External Routing 14-13
 - OSPF Support of Stub Areas 14-13
- Neighbors and Adjacency 14-13
 - The Hello Protocol 14-13
 - Designated Routers 14-14
 - Virtual Links 14-14
- Cisco's OSPF Implementation 14-14
- Steps in Configuring OSPF Routing 14-15

- Enabling the OSPF Routing Processes and Defining Areas 14-15
 - Enabling OSPF Routing 14-15
 - Defining OSPF on Networks and Assigning Area IDs 14-16
- Configuring OSPF Area Parameters 14-18
 - Setting Simple OSPF Area Authentication 14-18
 - Defining a Stub Area 14-18
 - Consolidating Advertised Addresses 14-19
- Configuring OSPF Interface-Specific Parameters 14-19
 - Specifying OSPF Path Cost 14-20
 - Setting the Link State Retransmission Interval 14-20
 - Setting the Transmission Time for Link State Updates 14-21
 - Setting Router Priority 14-21
 - Setting the Advertised Hello Interval 14-21
 - Setting the Router Dead Interval 14-22
 - Specifying the OSPF Authentication Key 14-22
- Configuring OSPF for Nonbroadcast Networks 14-23
 - Creating Virtual Links 14-24

Configuring the RIP Protocol 14-25

- Creating the RIP Routing Process 14-26
- Specifying the List of Networks 14-26

Configuring the Hello Protocol 14-27

- Creating the Hello Routing Process 14-28
- Specifying the List of Hello Networks 14-28

Configuring the BGP Protocol 14-28

- Creating the BGP Routing Process 14-28
- Specifying the List of BGP Networks 14-29
- Specifying the List of Neighbors 14-29
 - Basic Neighbor Specification 14-30
 - Setting Route Weights 14-31
 - Filtering BGP Advertisements 14-31
- Filtering BGP Routes 14-32
 - Defining a BGP Access List 14-32
 - Specifying BGP Route Filters 14-32
- Specifying BGP Version Number 14-33
- Specifying BGP Administrative Distance 33
- Adjusting the BGP Timers 14-34
- Clearing BGP Connections 14-35

BGP and IGP Routing Information 14-35

- Backdoor Routes 14-36
- BGP Route Selection Rules 14-36
- BGP Path Attributes 14-37
- Using BGP Without IGP Redistribution 14-37

Configuring the EGP Protocol 14-38

- Specifying the Autonomous System Number 14-39
- Creating the EGP Routing Process 14-39
- Specifying the List of Neighbors 14-39
- Specifying the Network to Advertise 14-39
- Adjusting Timers 14-41
- Configuring Third-Party EGP Support 14-42
- Configuring a Backup EGP Router 14-42
- Generating an EGP Default Route 14-43
- Defining a Core Gateway EGP Process 14-43

Filtering Routing Information 14-45

- Filtering Outgoing Information 14-45
 - Suppressing Updates on an Interface 14-45
 - Filtering Outbound Updates 14-47
 - Point-to-Point Updates 14-48
 - Adjusting Metrics 14-49
- Filtering Incoming Information 14-49
 - Filtering Received Updates 14-49
 - Filtering Sources of Routing Information 14-50
- Directly Connected Routes 14-52
 - Treatment of Directly Connected Routes 14-52
 - Multiple Interface Addresses 14-53
- Overriding Static Routes with Dynamic Protocols 14-54
- Default Routes 14-54
 - Generating Default Routes 14-54
 - Picking a Default Route 14-55
- Redistributing Routing Information 14-55
 - Supported Metric Translations 14-56
 - Passing Routing Information Among Protocols 14-57
 - Setting Default Metrics 14-58
 - Redistributing Routes into OSPF 14-59
 - Generating Default AS Boundary Router Routes for OSPF 14-61
 - Redistributing OSPF Routes into Other Domains 14-62

Special Routing Configuration Techniques 14-63

- Configuring Static Routes 14-63
- Enabling and Disabling Split Horizon for IP Networks 14-64
- IGRP Metric Adjustments 14-66
- Keepalive Timers 14-68
- Adjustable Routing Timers 14-69
- Gateway Discovery Protocol (GDP) 14-70
 - Using GDP Commands 14-72

ICMP Router Discovery Protocol 14-73

Using IRDP Commands 14-73

Displaying IRDP Values 14-74

IP Routing Protocol Configuration Examples 14-74

Static Routing Redistribution 14-74

RIP and Hello Redistribution 14-75

IGRP Redistribution 14-75

Third-Party EGP Support 14-76

Backup EGP Router 14-76

BGP Route Advertisement and Redistribution 14-77

OSPF Routing and Route Redistribution 14-78

Maintaining IP Routing Operations 14-83

Monitoring IP Routing Operations 14-83

Displaying the BGP Routing Table 14-84

Displaying BGP Neighbors 14-85

Displaying Routes Learned from a Neighbor 14-87

Displaying BGP Paths 14-87

Displaying BGP Summaries 14-88

Displaying EGP Statistics 14-88

Displaying Routing Protocol Parameters and Status 14-89

Displaying OSPF Routing Processes 14-90

Displaying the OSPF Database 14-92

Displaying OSPF Interface Parameters 14-98

Displaying OSPF Neighbor Information 14-99

Displaying the Routing Table 14-99

Displaying Network Masks 14-102

Debugging IP Routing 14-102

Global Configuration Command Summary 14-104

Router Subcommand Summary 14-106

IP Routing Interface Subcommands 14-116

Chapter 14

The IP Routing Protocols

14

This chapter describes routing protocol options for the Internet Protocol (IP) suite. Chapter 13, “Routing IP,” contains all the information you need for configuring IP. This chapter focuses on IP routing protocols. Other protocol stacks—DECnet, Novell, Apollo, and so on—are described in their own chapters. Topics in this chapter include:

- An introduction to the 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

Cisco-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.

Cisco routers offer 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 14-1). 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.

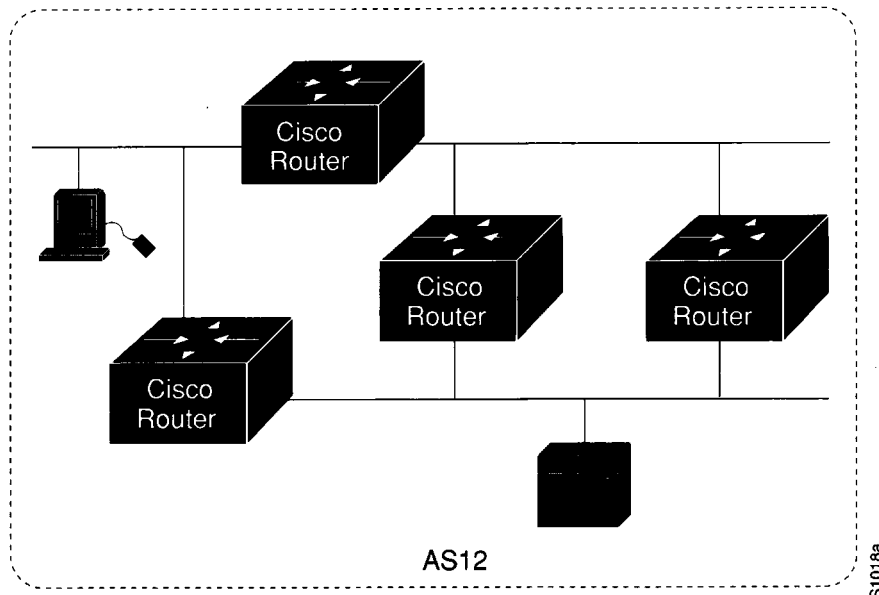


Figure 14-1 Autonomous System 12 Contains Four Routers

Multiple Routing Protocols

The multiple routing protocol support in the Cisco routers 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” later in this chapter.

Multiple IP Routing Processes

Cisco routers 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:

- Step 1:* Create the routing process with one of the **router** commands.
- Step 2:* Configure one or more **network** router subcommands.
- Step 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 the IGRP Protocol

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. The IGRP protocol can advertise all connected and IGRP-derived networks for a particular autonomous system.

Interior, System, and Exterior Routes

IGRP advertises three types of routes: interior, system, and exterior, as shown in Figure 14-2. 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.

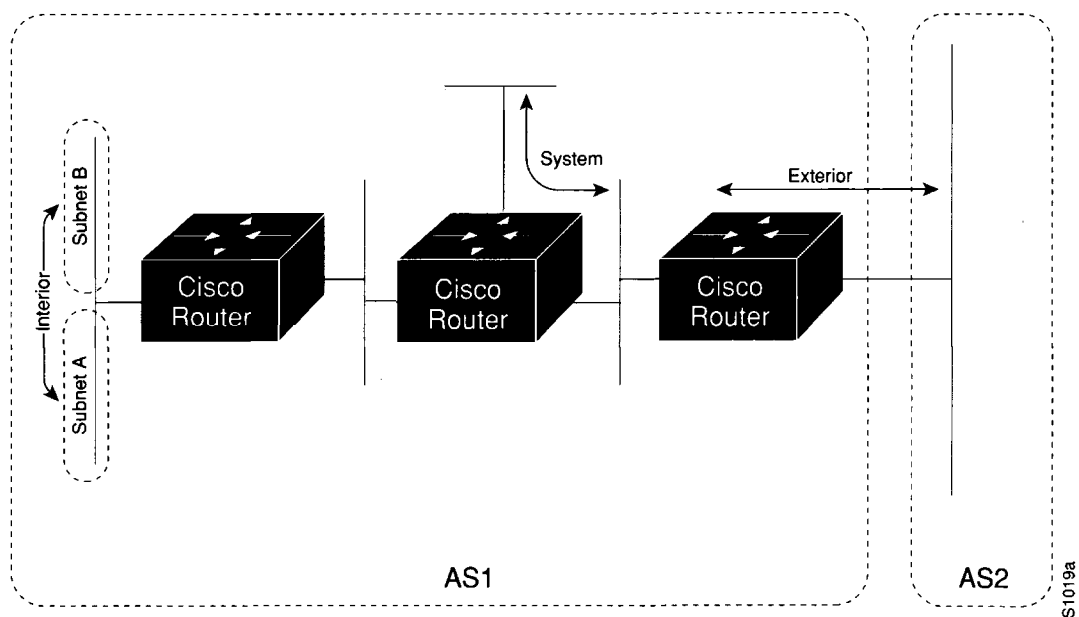


Figure 14-2 Interior, System, and Exterior Routes

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” later 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 14-3 illustrates the process of determining IGRP path feasibility.

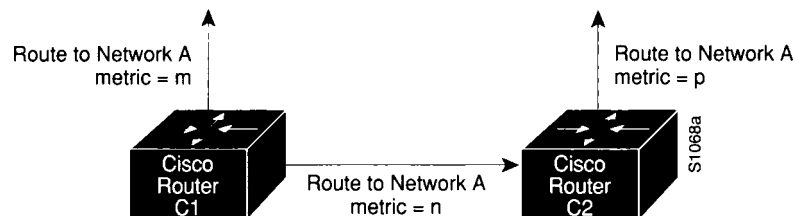


Figure 14-3 Determining IGRP Path Feasibility

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 .

1. If m is greater than p , then the first condition is met.
2. 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 MTU (Maximum Transmission Unit), 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 9,600 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
- Cisco support of OSPF
- Steps in configuring OSPF for Cisco routers and details about Cisco'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 Cisco's 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 Cisco's 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: Cisco 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 the outside of 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 router**—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 router**—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 router**—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 routers.

- **AS boundary router**—A router that exchanges routing information with routers belonging to other ASs. 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.

Figure 14-4 illustrates the various OSPF router types and their relationships to each other and to the overall OSPF environment.

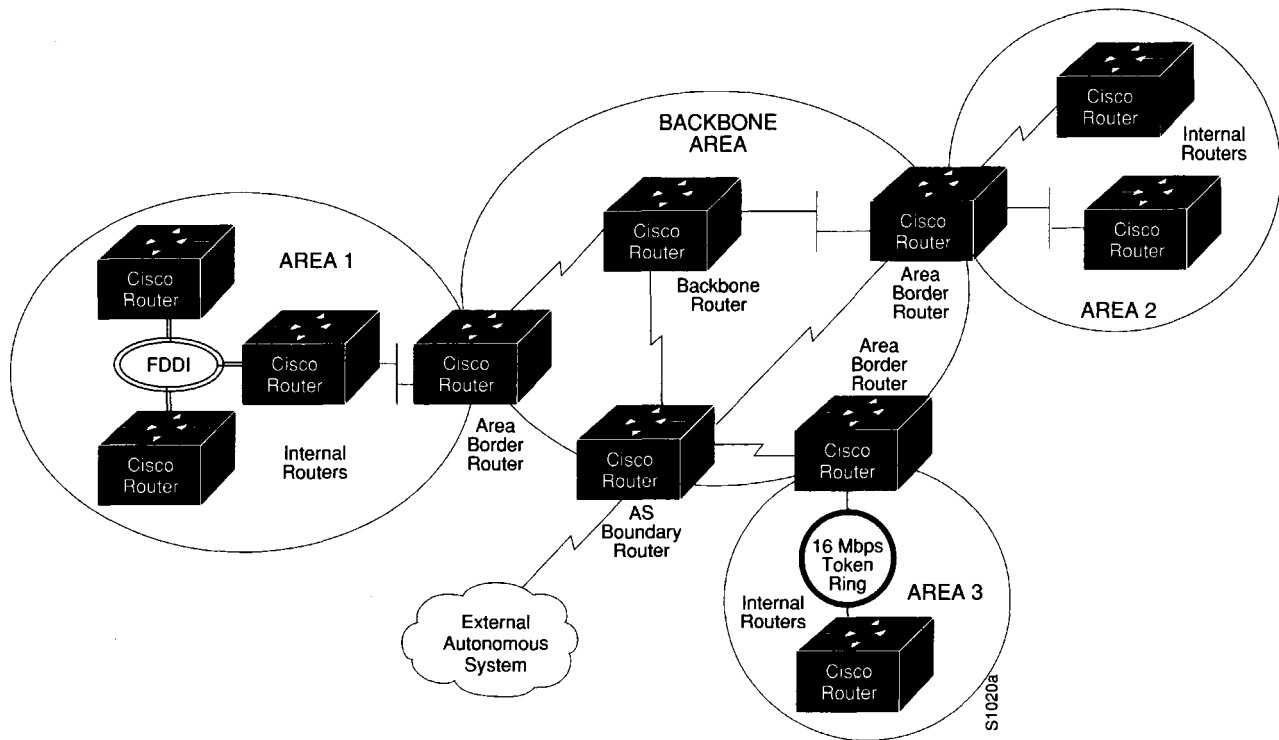


Figure 14-4 Overview of OSPF Router Types and Relationships

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 Cisco's implementation.

OSPF Physical Network Support

OSPF routing implementations support the following types of physical networks:

- **Point-to-point networks**—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**—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**—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 ASs 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*. The designated router 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.

Cisco's OSPF Implementation

The sections that follow detail the specific router and interface subcommands used to enable and configure the OSPF IP routing protocol on Cisco routers. Cisco's implementation conforms to the OSPF Version 2 specifications detailed in Internet RFC 1247. The list that follows outlines key features supported in Cisco's 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**—Cisco 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.

- Step 1:** Enable OSPF using the **router ospf** global command and **network** router subcommand.
- Step 2:** Specify route redistribution, as needed (addressed separately with redistribution discussion for all IP routing protocols).
- Step 3:** Set any OSPF area parameters, as needed.
- Step 4:** Set interface parameters, as needed.
- Step 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 131.109.10.0 0.0.0.255 area 3
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 Ethernet1 is in area 2:
interface Ethernet 1
ip address 131.108.1.5 255.255.255.0
!
! Interface Ethernet2 is in area 2:
interface Ethernet 2
ip address 131.108.2.5 255.255.255.0
!
! Interface Ethernet3 is in area 3:
interface Ethernet 3
ip address 131.109.10.5 255.255.255.0
!
! Interface Ethernet4 is in area 0:
interface Ethernet 4
ip address 131.109.1.1 255.255.255.0
!
! Interface Ethernet5 is in area 0:
interface Ethernet 5
ip address 10.1.0.1 255.255.0.0
```

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:

- Step 1:** The *wildcard-mask* is logically ORed with the interface IP address.
- Step 2:** The *wildcard-mask* is logically ORed with *address* in **network** command.
- Step 3:** The router compares the two resulting values.
- Step 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 Ethernet 1. OSPF is then enabled for that interface and Ethernet 1 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
- 4-Mbps Token Ring—Default cost is 25
- Ethernet—Default cost is 10
- 16-Mbps Token Ring—Default cost is 6
- FDDI—Default cost is 1

In general, the path cost is calculated as follows:

$$\frac{10^8}{\textit{Bandwidth}}$$

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 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
```

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 network's 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 DeadInterval 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 14-5); 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.

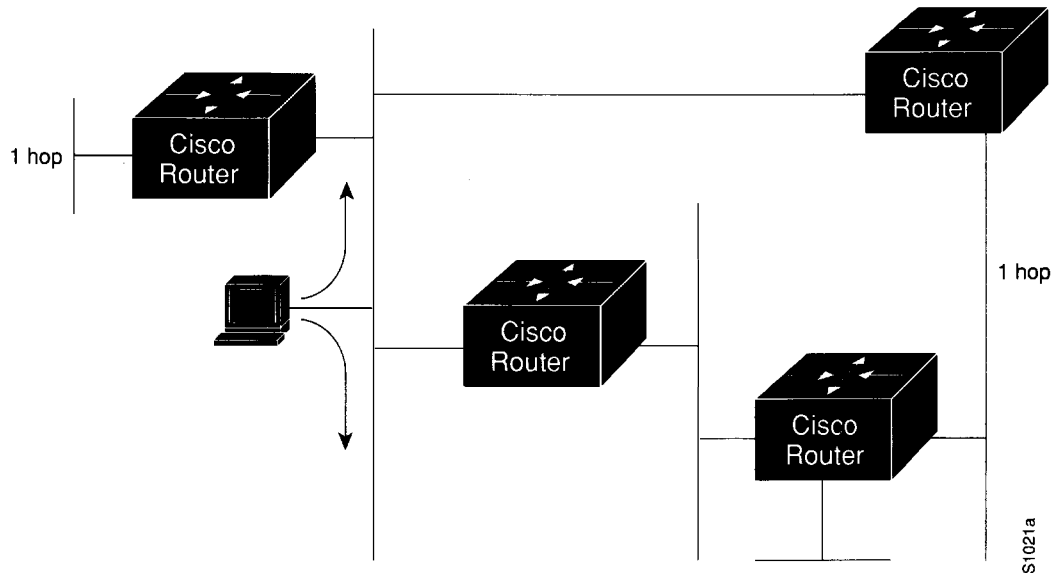


Figure 14-5 Hop Count in RIP

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 Cisco Systems 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 14-6). 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 Cisco Systems implementation uses a delay of 100 milliseconds for all routes, regardless of their actual delay characteristics.

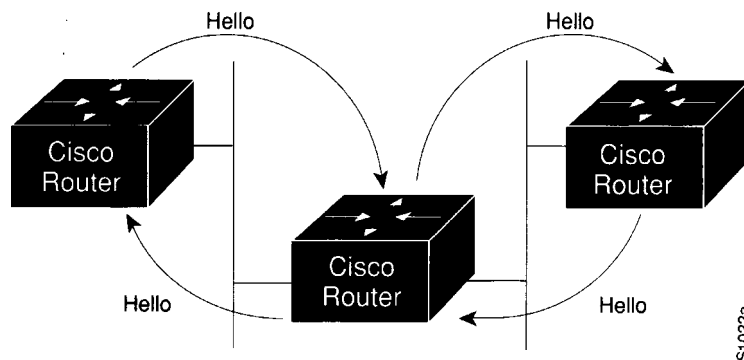


Figure 14-6 The Hello Protocol

Creating the Hello Routing Process

Create the routing process with the **router hello** 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 ASs) 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
no 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 13 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 IGP's), 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 ASs 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 14-7). In most cases, the BGP information should override IGP information.



Figure 14-7 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 ASs. 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 14-8, 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.

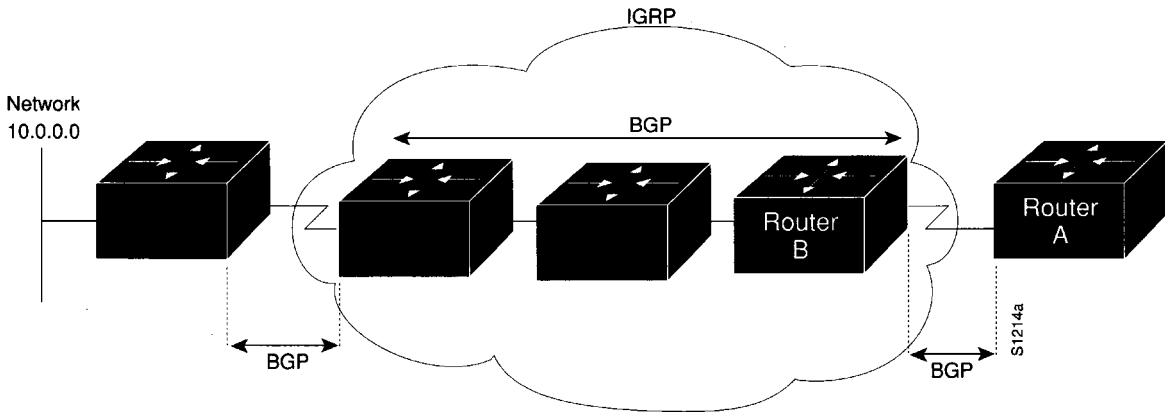


Figure 14-8 Illustration of Synchronization

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 14-9. 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.

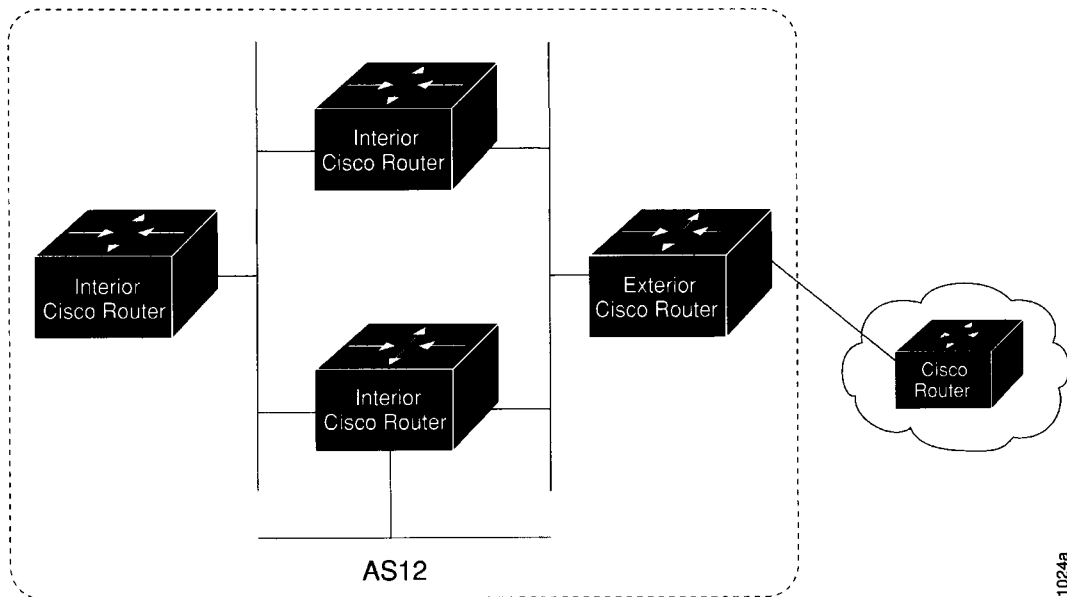


Figure 14-9 EGP and Interior and Exterior Routers

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” later 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 14-10. 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
```

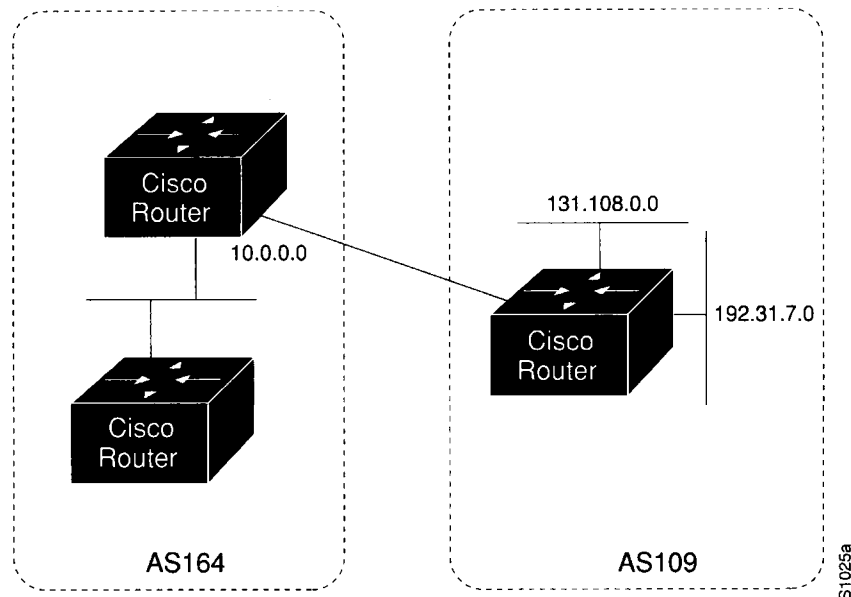



Figure 14-10 Router in AS 164 Peers with Router in AS 109

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” later 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 Cisco router and the EGP peer specified by the *address* argument. All networks reachable through that third-party router will be listed in the Cisco EGP updates as reachable via that router. Any other networks will be listed as reachable via the Cisco router. 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-metric** 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 Protocols Configuration Examples” later 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 ASs, 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 Cisco router about a specific network, and that router is the *best* path, the Cisco router will *not* inform the originating external router about that path). Cisco routers 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 14-11 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
```

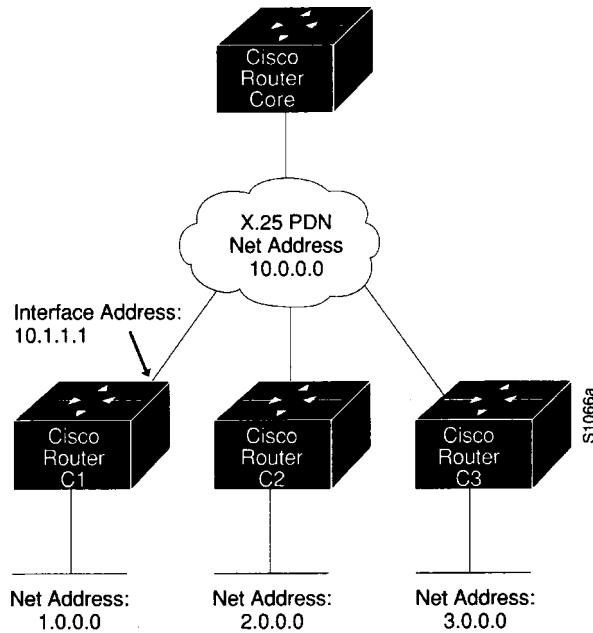


Figure 14-11 Core EGP Third-Party Update Configuration Example

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 1. Figure 14-12 shows this configuration.

```
router igrp 109
network 131.108.0.0
passive-interface ethernet 1
```

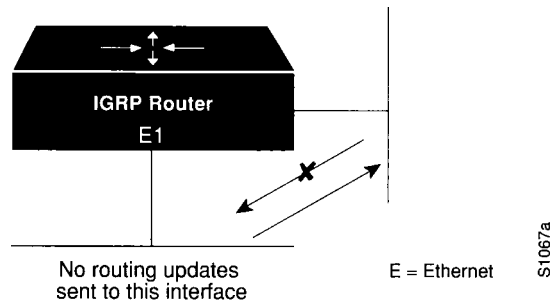


Figure 14-12 Filtering IGRP Updates

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 1. 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 1
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 Ethernet 1
ip address 131.108.2.1 255.255.255.0
interface Ethernet 2
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 Ethernet 2
```

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 13. 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” later 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 3. Figure 14-13 illustrates the effects of this command.

```
distribute-list 1 out ethernet 3
```

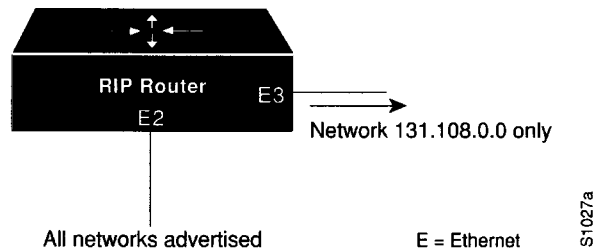


Figure 14-13 Filtering RIP Updates

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 13 of this manual. 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 a 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 14-1 shows the default administrative distance for various sources of routing information.

Table 14-1 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 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” later 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 14-2. 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 14-2. 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 14-2 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
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” later 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” earlier 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 14-14 shows this type of redistribution.

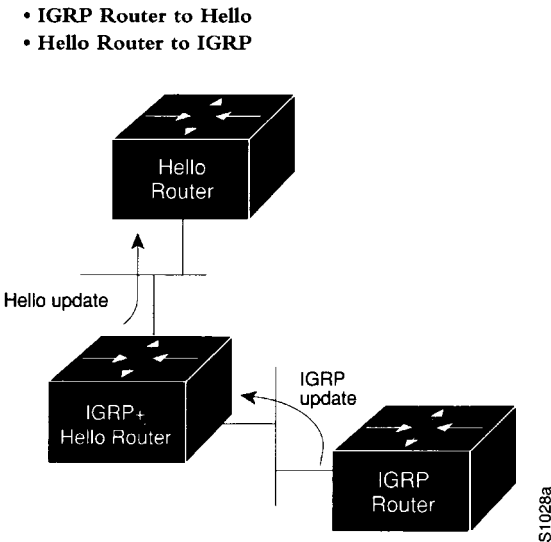


Figure 14-14 Assigning Metrics for Redistribution

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 routers 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 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 14-15. 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 14-15 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.

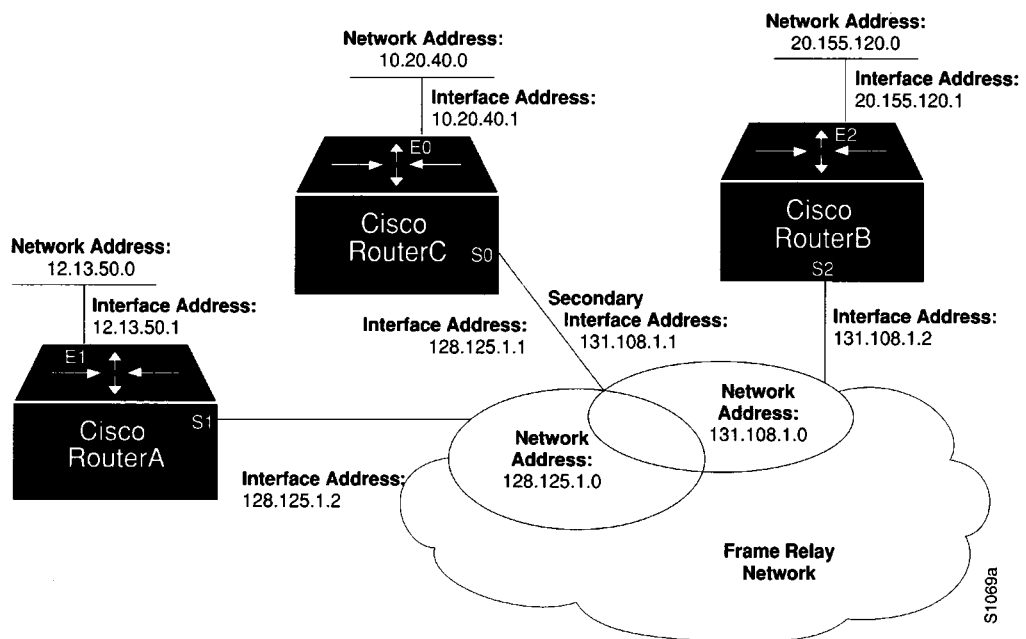


Figure 14-15 Disabled Split Horizon Example for Frame Relay Network

Example interface configuration for RouterA:

```
interface ethernet 1
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 2
ip address 20.155.120.1
!
interface serial 2
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:

$$\text{metric} = [K1 * \text{bandwidth} + (K2 * \text{bandwidth}) / (256 - \text{load}) + K3 * \text{delay}]$$

If *K5* does not equal 0, an additional operation is done:

$$\text{metric} = \text{metric} * [K5 / (\text{reliability} + K4)]$$

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 14-3 lists the default values used for several common media.

Table 14-3 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
```

Note: This command assumes that the entire AS is running Release 8.2(5) or more recent software, because the hop count is used to avoid information looping. Using it with earlier software will cause problems.

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, cable unterminated, and so on).

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. Unsupported, example GDP clients can be obtained from Cisco Systems.

GDP is not a standard; it is a protocol designed by Cisco Systems to meet the customers' needs. 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 14-16 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.

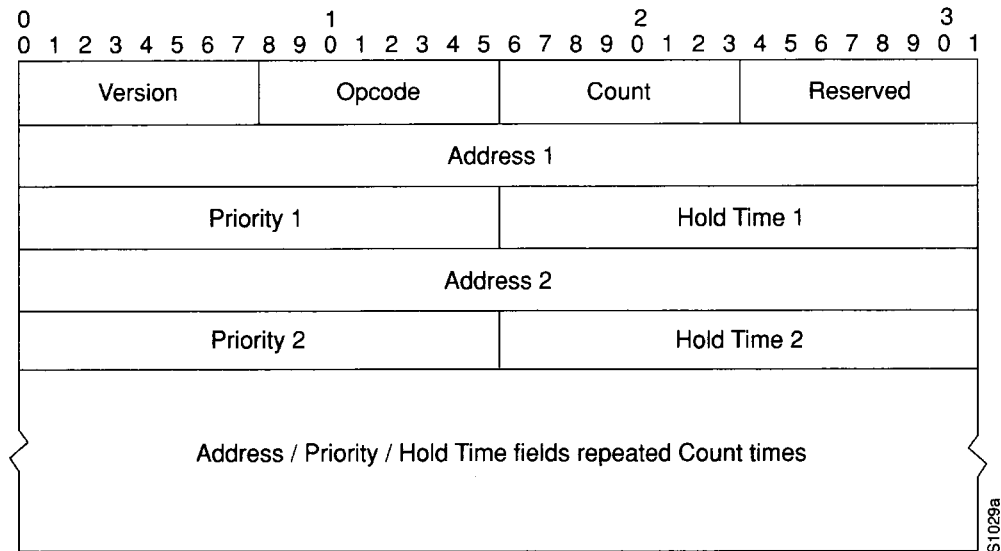


Figure 14-16 GDP Report Message Packet Format

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 1 with a report time of ten seconds, and priority and hold time set to their defaults (because none are specified).

```
ip gdp reporttime 10
```

ICMP Router Discovery Protocol

The ICMP Router Discover Protocol (IRDP) implemented in Cisco's router 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 Ethernet 0 and Ethernet 1 to area 0.0.0.0, and redistributes RIP into OSPF.

```

interface Ethernet0
ip address 130.93.1.1 255.255.255.0
ip ospf cost 1
!
interface Ethernet 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 14-17 provides a general network map that illustrates this example configuration.

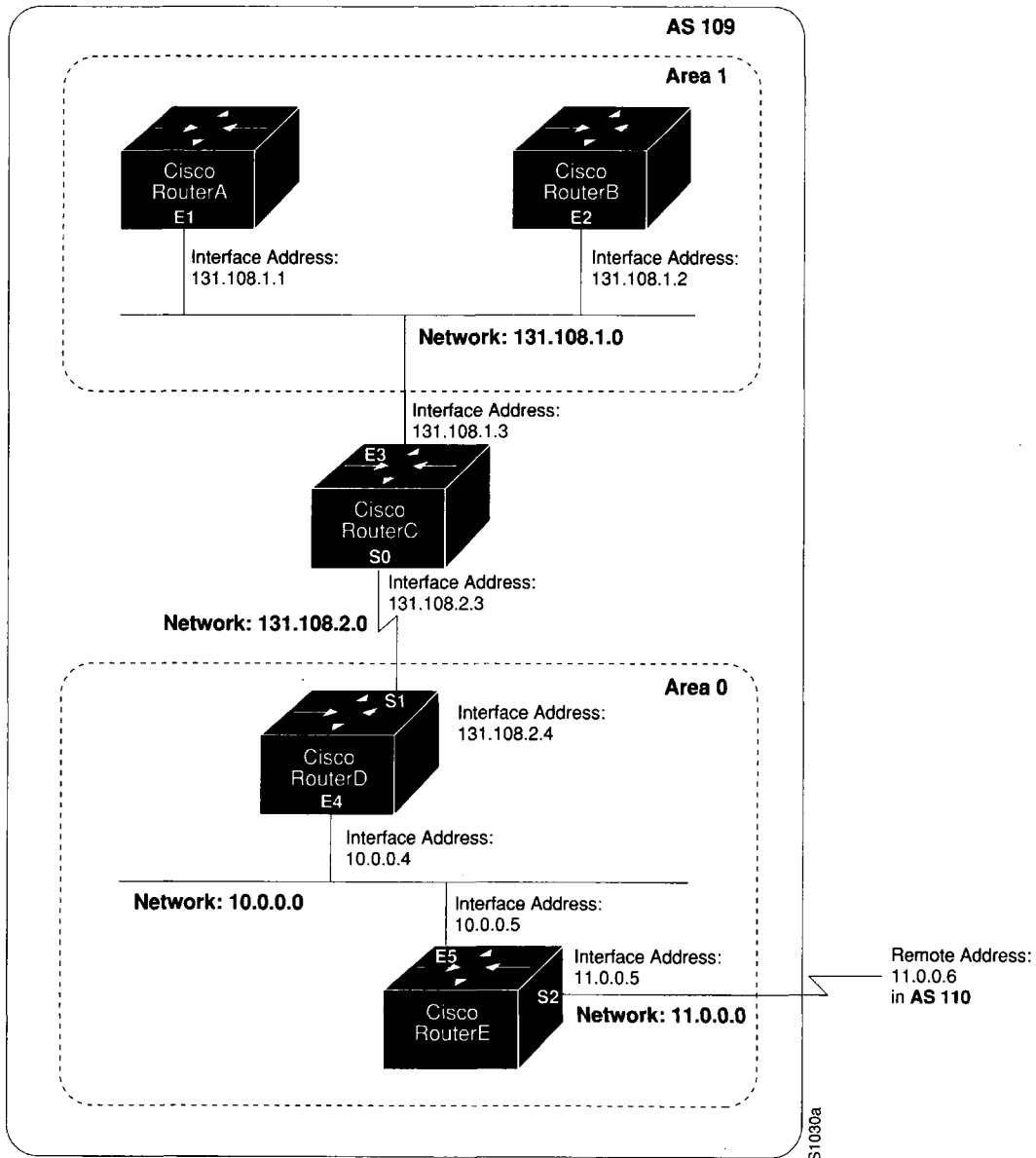


Figure 14-17 Sample OSPF Autonomous System Network Map

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 1
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 2
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 3
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 4
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 5
ip address 10.0.0.5 255.0.0.0

interface Serial 2
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

```

Example 3: Complex OSPF Configuration

The following example configuration accomplishes several tasks in setting up an area border router. These tasks can be split into two general categories:

- Basic OSPF configuration
- Route redistribution

The specific tasks outlined in this configuration are detailed briefly in the following descriptions. Figure 14-18 illustrates the network address ranges and area assignments for the interfaces.

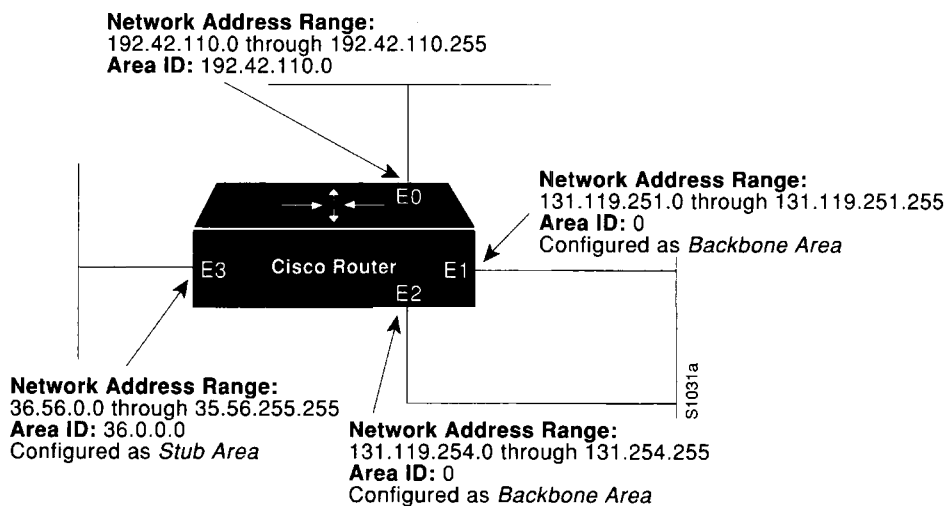


Figure 14-18 Interface and Area Specifications for OSPF Example Configuration

The basic configuration tasks in this example are as follows:

- Configure address ranges for interfaces Ethernet 0 through Ethernet 3.
- Enable OSPF on each interface.
- Set up an OSPF authentication password for each area and network.
- Assign link state metrics and other OSPF interface configuration options.
- Create a *stub area* with area id 36.0.0.0. (Note that the **authentication** and **stub** options of the **area** router subcommand are specified with separate subcommand entries, but can be merged into a single **area** subcommand.)
- Specify the backbone area (Area 0).

Configuration tasks associated with redistribution are as follows:

- Redistribute IGRP and RIP into OSPF with various options set (including **metric-type**, **metric**, **tag**, and **subnet**).
- Redistribute IGRP and OSPF into RIP.

The following is an example OSPF configuration command listing:

```
interface Ethernet0
ip address 192.42.110.201 255.255.255.0
ip ospf authentication-key abcdefgh
ip ospf cost 10
!
interface Ethernet1
ip address 131.119.251.201 255.255.255.0
ip ospf authentication-key ijklmnop
ip ospf cost 20
ip ospf retransmit-interval 10
ip ospf transmit-delay 2
ip ospf priority 4
!
interface Ethernet2
ip address 131.119.254.201 255.255.255.0
ip ospf authentication-key abcdefgh
ip ospf cost 10
!
interface Ethernet3
ip address 36.56.0.201 255.255.0.0
ip ospf authentication-key ijklmnop
ip ospf cost 20
ip ospf dead-interval 80
!
! OSPF on network 131.119
!
router ospf 201
network 36.0.0.0 0.255.255.255 area 36.0.0.0
network 192.42.110.0 0.0.0.255 area 192.42.110.0
network 131.119.0.0 0.0.255.255 area 0
area 0 authentication
area 36.0.0.0 stub
area 36.0.0.0 authentication
area 36.0.0.0 default-cost 20
```

```

area 192.42.110.0 authentication
area 36.0.0.0 range 36.0.0.0 255.255.0.0
area 192.42.110.0 range 192.42.110.0 255.255.255.0
area 0 range 131.119.251.0 255.255.255.0
area 0 range 131.119.254.0 255.255.255.0

redistribute igrp 200 metric-type 2 metric 1 tag 200 subnets
redistribute rip metric-type 2 metric 1 tag 200
!
! IGRP AS 200 on 131.119.0.0
!
router igrp 200
network 131.119.0.0
!
! RIP for 192.42.110
!
router rip
network 192.42.110.0
redistribute igrp 200 metric 1
redistribute ospf 201 metric 1

```

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 is 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  delrcwnd:  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 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 the description of **show ip tcp** in Chapter 13 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** command.

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 14-4 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 14-4 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	<ul style="list-style-type: none">• 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
  Redistributing: 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 192.42.110.200
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 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**, a 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 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:

```
gw1est> 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, ethernet0, or fddi1) or can be formed as the two-word *interface-type unit-number* specification (for example, serial 0, ethernet 1, or e 2).

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, ethernet0, or fddi1) or as the two-word *interface-type unit-number* specification (for example, serial 0, ethernet 1, or e 2).

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 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, Ethernet2
E   192.67.131.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
O E2 192.68.132.0 [160/5] via 131.119.254.6, 0:00:59, Ethernet2
O E2 130.130.0.0 [160/5] via 131.119.254.6, 0:00:59, Ethernet2
E   128.128.0.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
E   129.129.0.0 [200/129] via 131.119.254.240, 0:02:22, Ethernet2
E   192.65.129.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
E   131.131.0.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
E   192.75.139.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet2
E   192.16.208.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
E   192.84.148.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet2
E   192.31.223.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
E   192.44.236.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet2
E   140.141.0.0 [200/129] via 131.119.254.240, 0:02:22, Ethernet2
E   141.140.0.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet2
```

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 Serial2, 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 Cisco 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 about all neighbors. 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 R.IP 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.

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 number. The argument *local-AS* is the local autonomous system (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.

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] 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 Unit (MTU) of the route. The **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 routes those 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). This assumes that the entire AS is running Version 8.2(5) or more recent software since the hop count is used to avoid information looping. Using it with Version 7.1 or earlier software will cause problems.

[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 Type of Service (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**]

Control 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 non-broadcast, 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 DeadInterval 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 Cisco router 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 handle 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 back-door 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 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.

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 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 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.

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 ASs. 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. Cisco does not support Type of Service (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 disable 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.

[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.

Router Products Configuration and Reference

Index



Index

Symbols

- > (angle bracket)
 - as system prompt 2-8
- ? (question mark) command
 - use of 2-9

Numerics

- 3Com access lists 20-21
- 80D5
 - header, described A-1
 - header, illustrated A-2

A

AARP

- AppleTalk routing protocol 10-2
- retransmission count, specifying 10-22
- transmissions time interval, specifying 10-21

access control

- DECnet Phase IV routing 12-18
- NetBIOS filtering 22-55
- terminal 4-25
- using byte offset 22-58
 - configuration 22-58
- using station names 22-55
 - configuration 22-55

access expressions

- and MAC addresses 22-61
- and NetBIOS 22-61
- Boolean, combining access filters 22-61
- configuring 22-62
- designing 22-62
- for filtering on Token Ring 22-61
- optimizing 22-66

access groups

- Apollo Domain 9-6
- assigning 10-31
- configuring 12-18
- IP 13-28

access lists

- 3Com 20-21

and access expressions, altering 22-67

Apollo Domain 9-5

AppleTalk 10-27, 10-29

examples 10-49

assigning to DDR interfaces 6-70

BGP, defining 14-32

bridging

extended 21-28

byte offset 22-59

bytes filter 22-60

configuring IP 13-23

configuring standard 13-24

configuring vendor code 22-49

controlling traffic 17-10

DECnet 12-13, 12-14

connect initiate filtering 12-17

using to manage traffic 12-13

definition of 13-23

displaying IP 13-28

extended 12-14

extended XNS 20-16

filtering by protocol type 21-19, 22-45

filtering on outgoing messages 22-58

filtering outgoing information (IP) 14-45

IP 13-23, 13-24, 13-45

IP extended

configuring 13-25, 13-46

NetBIOS filtering 22-55

NetBIOS station name 22-56

Novell IPX 17-8, 17-9

numerical ranges, by protocol B-6

SAP filtering 17-14

SLIP 13-54

SNMP 4-32

station, specifying filter 22-57

summary by protocol B-1

used by expression

configuring 22-63

designing 22-63

vendor code 21-23

VINES 19-8

access-class command 4-45

access-class in command 13-27
 access-class out command 13-27
 access-list command 10-29, 12-14, 12-15, 13-24,
 13-25, 17-9, 17-14, 21-29
 access-list deny command 10-29, 12-13, 13-24, 17-8,
 20-15, 20-16, 21-19, 21-23, 22-45, 22-49
 access-list permit command 10-29, 13-24, 17-14,
 20-15, 20-16, 21-19, 21-23, 22-45, 22-49
 accounting
 See IP accounting
 accounting option 6-7
 active lines
 See line, displaying active
 active sessions
 See sessions, displaying active
 active system processes
 See system processes
 address mapping information
 displaying 12-28
 address mask
 IP 13-24, 14-53
 address prioritization
 SNA local LU 22-33
 Address Resolution Protocol
 See ARP
 Address Translation Gateway
 See ATG
 addresses
 addressing rules 15-6
 addressing structure 15-6
 Apollo Domain 9-2
 AppleTalk nonextended 10-5
 assigning IP 13-2
 CHAOSnet 11-1
 classes, formats 13-3
 DECnet 12-2
 destination, filtering 21-24, 22-50
 dynamic MacIP clients, specifying 10-25
 filtering multicast 21-18
 helper 13-44
 Internet broadcast 13-12
 Internet notation 13-2
 IP 13-2, 13-7, 14-53
 IS-IS NSAP 15-5
 ISO CLNS 15-4
 ISO-IGRP NSAP 15-5
 MAC, See MAC address
 mapping to multicast 8-70
 mappings 8-9, 8-14
 Novell IPX 17-1
 NSAP 15-5, 15-6, 16-2
 PUP 18-1
 host 18-2
 subnet 18-2
 PVC protocol 8-12
 resolution using ARP 13-8
 resolution using Probe 13-10
 resolution using Proxy ARP 13-10
 resolution, local 13-8
 secondary IP 13-7, 14-53
 SMDS 8-67, 8-75
 source, filtering 21-23, 22-49
 specifying SLIP 13-51
 static MacIP clients, specifying 10-26
 static map for individual, SMDS 8-69
 static name-to-address mappings 13-20
 subnet zero 13-7
 suppress called 8-32
 suppress calling 8-31
 VINES 19-2
 X.121 8-8, 8-19, 8-23, 8-29
 X.25 8-19, 8-23
 XNS 20-2
 adjustable routing timers 14-69
 administrative distance 14-50, 14-51, 14-52
 administrative filtering
 by protocol type 22-45
 by vendor code or address 22-49
 combining with Boolean access expressions
 22-61
 destination addresses 21-24, 22-50
 dynamically configured stations 21-18
 Ethernet address 21-16, 21-17
 Ethernet-encapsulated packets 21-20, 21-21
 IEEE 802.3-encapsulated packets 21-21, 21-22
 IEEE 802.5-encapsulated packets 22-47
 LAT service announcements 21-25
 MAC-layer address 21-16
 multicast addresses 21-18
 protocol type 21-19
 SNAP-encapsulated packets 21-20, 22-47
 source addresses 21-23
 source-route bridging 22-45
 statically configured stations 21-18

- vendor code 21-23
 - access lists 22-49
- AEP
 - AppleTalk 10-2
- AGS+ temperature readings
 - obtaining 5-8
- AGS+ voltage readings
 - obtaining 5-8
- alias
 - AppleTalk, NBP 10-6
 - ARP responses, IP 13-9
 - displaying MacIP 10-69
 - Internet address mappings, SLIP 13-67
 - MacIP address requirements 10-24, 10-25
- American National Standards Institute
 - See ANSI
- analyzer
 - connecting 4-42
- angle bracket (>)
 - as system prompt 2-8
- ANSI
 - contacting F-4
- ANSI frame relay
 - enabling 8-56
 - monitoring 8-63
- apollo access-group command 9-6
- apollo access-list deny command 9-6
- apollo access-list permit command 9-6
- Apollo Domain
 - access groups 9-6
 - access lists 9-5
 - configuring 9-5
 - addresses 9-2
 - assigning network number 9-3
 - Cisco's implementation 9-1
 - debugging the network 9-9
 - displaying ARP table 9-8
 - displaying interface parameters 9-7
 - displaying routes 9-7
 - displaying traffic 9-8
 - global configuration command summary 9-9
 - interface subcommand summary 9-10
 - monitoring the network 9-7
 - multiple paths 9-4
 - restrictions 9-1
 - routing
 - enabling 9-3
 - static routes 9-3
 - update timers 9-4
- apollo maximum-paths command 9-4
- apollo network command 9-3
- apollo route command 9-4
- apollo routing command 9-3
- apollo update-time command 9-4
- apple event-logging command 10-83
- Apple Networking Architecture (ANA) 10-3
- AppleTalk
 - access control methods 10-28
 - access list configuration examples 10-49
 - access lists
 - assigning 10-31
 - defining 10-29
 - assigning cable range for Phase II (extended) 10-13
 - assigning nonextended (Phase I) address 10-13
 - assigning zone names 10-14
 - checksum 10-20
 - configuration examples 10-38
 - configuration guidelines 10-12
 - configuring for SMDS 8-72
 - configuring IPTalk 10-16, 10-44
 - configuring over X.25 10-41
 - configuring routing updates 10-18
 - configuring SNMP 10-43
 - controlling names displayed 10-35
 - DDP protocol 10-2
 - debugging the network 10-82
 - description 10-1
 - discovery mode, setting 10-15
 - displaying AARP cache 10-63
 - displaying adjacent routes 10-62
 - displaying aliased MacIP clients 10-69
 - displaying directly connected routes 10-71
 - displaying fast-switching cache 10-63
 - displaying global information 10-64
 - displaying network routing table 10-73
 - displaying registered NBP services 10-70
 - displaying socket information 10-75
 - displaying specific interface information 10-65
 - displaying traffic 10-76
 - displaying zone information 10-78
 - dynamic address assignment 10-7
 - enabling routing 10-11, 10-13

- EtherTalk 2.0 10-5
- extended (Phase II) 10-4
- extended addresses 10-9
- extended routing over HDLC 10-43
- extended zones 10-10
- filtering incoming updates 10-31
- filtering outbound updates 10-32
- get-zone-list filters 10-33
- global configuration command summary 10-84
- interface subcommand summary 10-88
- interfaces supported 10-2
- IP encapsulation 10-17
- IPTalk configuration steps 10-44
- MacIP alias requirements 10-24
- MacIP defined 10-22
- mapping DDP to UDP 10-17
- monitoring the network 10-62
- NBP aliases 10-6
- NBP ping interface 10-79
- NBP routing protocol 10-6
- network number-based access control 10-28
- node numbers 10-7
- nonextended (Phase I) 10-4
- nonextended addresses 10-5
- nonextended routing example 10-39
- OSI reference model 10-3
- packet validity 10-18
- permitting partial zones 10-34
- Phase I 10-4
- Phase II 10-4
- ping command 10-78
- proxy network number 10-19
- requiring specific route zones 10-35
- RTMP routing table 10-8
- seed router 10-7
- setting NBP service types cached 10-36
- setting routing update timers 10-18
- setting NBP service-type lookup interval 10-38
- simple configuration, extended AppleTalk 10-42
- transition mode 10-40
 - example 10-40
- UDP port mapping 10-17
- UNIX host configuration for IPTalk 10-46
- well-known sockets 10-17
- ZIP routing protocol 10-7
- zone-based access control 10-28
- zones 10-5
- appletalk access-group command 10-31
- appletalk address command 10-13
- AppleTalk Address Resolution Protocol
 - See AARP
- appletalk arp interval command 10-21
- appletalk arp retransmit-count command 10-22
- appletalk cable-range command 10-14
- appletalk checksum command 10-20
- appletalk discovery command 10-15
- appletalk distribute-list command 10-32
- AppleTalk Echo Protocol
 - See AEP
- appletalk getzonelist-filter command 10-33
- appletalk iptalk command 10-16
- appletalk iptalk-baseport command 10-17
- appletalk lookup-type command 10-36
- appletalk macip dynamic command 10-25
- appletalk macip server command 10-24
- appletalk macip static command 10-26
- appletalk name-lookup interval command 10-38
- appletalk permit-partial-zones command 10-34
- appletalk proxy-npb command 10-19
- appletalk require-route-zones command 10-35
- appletalk routing command 10-13
- appletalk send-rtmp command 10-18, 10-19
- appletalk strict-rtmp command 10-18
- AppleTalk Transaction Protocol (ATP) 10-3
- appletalk zone command 10-14
- applique
 - configuring for DCE 7-2
 - internal loop to 7-14
- area
 - DECnet Phase IV 12-6
 - ISO CLNS 15-2
- area command 14-18, 14-19
- area virtual-link command 14-24
- ARP
 - Apollo Domain 9-8
 - CHAOSnet 11-2
 - debugging transactions 13-75
 - definition of 13-8
 - displaying AppleTalk 10-63
 - displaying PUP 18-3
 - displaying SLIP cache 13-67
 - IP responses alias 13-9
 - HP probe 13-10
 - proxy 13-10

- static cache entries 13-8
 - using 13-8
 - VINES 19-5, 19-8
- arp arpa command 13-9
- ARP cache
 - clearing dynamic entries 13-57
 - displaying contents of 13-8, 13-59, 13-67
 - removing entries 13-9
 - timeout 13-10
- arp command 13-9
 - SMDS option 8-72
- arp probe command 13-9, 13-10
- arp snap command 13-9
- arp timeout command 13-9
- ARP, enabling
 - SMDS 8-69
- AS
 - definition of 14-2
- AS number
 - BGP 14-28
 - EGP 14-39
 - gateway of last resort 14-8
 - use for IGRP 14-5
- ASCII character set E-1
- async-bootp command 13-55
- AT&T SMDS service
 - enabling 8-71
- ATG
 - command syntax 12-24
 - configuration examples 12-24
 - description 12-24
 - limitations 12-27
 - routing table 12-24
- automatic warning message
 - receipt 5-9
- autonomous switching
 - and source-route bridging 22-4
 - enabling IP 13-40
- autonomous system
 - See AS
- autonomous-system command 14-39
- auxiliary line
 - configuring 4-42
- auxiliary port
 - configuring CPU 4-42
 - disabling SLIP on 13-51
 - RS-232 4-42

B

- backup delay command 6-58
- backup interface command 6-57
- backup line 6-57
 - defining delay 6-58
- backup load command 6-58
- backup service, dial
 - See dial backup
- bandwidth
 - setting interface 7-11
- bandwidth command 7-11
- banner
 - disabling 4-45
 - enabling 4-45
 - setting system 4-2
 - suppressing display 4-45
- banner command 4-2
- banner displays
 - order 4-4
- banner exec command 4-3
- banner incoming command 4-3
- banner message
 - changeable banners 4-2
 - displaying 4-2
 - EXEC process 4-3
 - incoming on specific terminal line 4-3
 - message-of-the-day 4-2
 - on incoming connections 4-3
 - suppressing 4-45
- banner motd command 4-2
- Banyan VINES
 - See VINES
- baud rate
 - setting SLIP 13-53
- BFE
 - encryption 8-36
- bfe {enter|leave} command 8-38, 8-90
- BGP
 - access list, defining 14-32
 - adjusting timers 14-34
 - administrative distance, specifying 14-33
 - advertisements, filtering 14-31
 - backdoor route 14-36
 - basic neighbor specification 14-30
 - configuring 14-28
 - connections, clearing 14-35

-
- creating routing process 14-28
 - definition of 14-2
 - displaying list of neighbors 14-29
 - displaying list of networks 14-29
 - displaying neighbor statistics 14-85
 - displaying paths 14-87
 - displaying routes learned from neighbor 14-87
 - displaying routing table 14-84
 - displaying status of all BGP connections 14-88
 - external neighbors 14-29
 - interacting with IGP 14-35
 - internal neighbors 14-29
 - neighbor, accepting any 14-30
 - path attributes 14-37
 - redistribution 14-56
 - route advertisement, redistribution 14-77
 - route selection rules 14-36
 - route weights, setting 14-31
 - routes, filtering 14-32
 - routing table 14-84
 - support in router 1-2
 - synchronization between BGP and IGP 14-37
 - version number, specifying 14-33
- bit control
 - setting for FDDI 6-42
 - black holes
 - eliminating 14-2
 - Blacker Emergency Mode 8-36
 - encapsulation 8-36
 - monitoring 8-39
 - Boolean access expressions
 - combining with administrative filters 22-61
 - boot buffersize command 4-9
 - boot command 2-16, 4-7
 - boot commands
 - configuring multiple instances 4-9
 - boot file
 - configuration 4-7
 - obtaining over the network 4-7
 - specifying buffer size 4-9
 - boot host command 4-7
 - boot loading
 - See netbooting
 - boot network command 4-7
 - boot system command 4-7
 - boot system flash command 4-16
 - boot system rom command 4-8
 - booting from Flash Memory
 - automatically 4-16
 - manually 4-17
 - booting system software
 - using the Flash Memory card 4-9
 - BootP
 - definition of 13-11
 - displaying SLIP 13-69
 - SLIP extended requests 13-55
 - use in reverse address resolution 13-11
 - use in SLIP 13-47
 - Border Gateway Protocol
 - See BGP
 - BPDU
 - intervals between Hello 21-13
 - Break key
 - function 3-8
 - BRI
 - See ISDN BRI
 - bridge
 - displaying current configuration 22-91
 - external Novell 17-1
 - remote source-route with direct encapsulation 22-22
 - root 21-13
 - spanning tree 21-4
 - See also source-route bridging
 - bridge acquire command 21-18
 - bridge domain command 21-9
 - bridge flooding 21-10
 - bridge forward-time command 21-14
 - bridge group protocol command 21-8
 - bridge hello-time command 21-13
 - bridge lat-service-filtering command 21-26
 - bridge max-age command 21-14
 - bridge multicast-source command 21-18
 - bridge priority command 21-13
 - Bridge Protocol Data Units
 - See BPDU
 - bridge-group circuit command 21-30
 - bridge-group command 21-10
 - bridge-group input-address-list command 21-23
 - bridge-group input-lat-service-deny command 21-26
 - bridge-group input-lat-service-permit command 21-27
 - bridge-group input-lsap-list command 21-21

- bridge-group input-type-list command 21-20
 - bridge-group lat-compression command 21-35
 - bridge-group output-address-list command 21-24
 - bridge-group output-lat-service-deny command 21-27
 - bridge-group output-lat-service-permit command 21-28
 - bridge-group output-lsap-list command 21-22
 - bridge-group output-type-list command 21-21
 - bridge-group path-cost command 21-15
 - bridge-group priority command 21-16
 - bridging
 - between source-route and transparent bridge groups (SR/TLB) 22-36
 - controlling access to media 21-2
 - controlling the logical link 21-2
 - LLC 21-1
 - MAC addresses 21-1
 - on frame relay 21-30
 - on X.25 21-30
 - overview 21-1
 - source-route, See source-route bridging
 - transit 21-5
 - transparent, See transparent bridging
 - X.25 frames 8-35
 - broadcast
 - control using helper facilities 17-19
 - definition of 13-11
 - flooding Novell IPX 17-21
 - flooding of IP 13-15, 13-43
 - forwarding IP packets 13-13
 - forwarding Novell IPX 17-19, 17-21
 - helper facilities, Novell IPX 17-18
 - Internet addresses 13-12
 - SLIP 13-48
 - storms 13-16
 - TCP/IP 13-16
 - UDP 13-13
 - buffer size
 - use for netbooting 4-9
 - buffer sizing
 - dynamic 4-6
 - buffers
 - internal, message logging 4-38
 - management parameters 4-5
 - pool statistics 5-2
 - setting 4-5
 - setting size of 4-7
 - setting system 4-5
 - buffers command 4-5
 - buffers huge size command 4-6
 - bypass mode
 - and FDDI optical bypass switch 6-3
 - byte offset
 - assigning access list name 22-59
 - pattern matching 22-59
 - use in access control 22-58
 - bytes access list filter
 - on incoming messages 22-60
 - on outgoing messages, specifying 22-60
- ## C
- cable ranges
 - AppleTalk extended 10-9
 - cache
 - ARP 13-59
 - DECnet Phase IV 12-11
 - displaying Novell IPX entries 17-24
 - displaying RIF 22-90
 - displaying route 13-61
 - host name-and-address 13-57
 - call request
 - retransmission timer 8-28
 - virtual circuit 8-13
 - Call User Data
 - compressed TCP 8-35
 - defined 8-13
 - protocols and initial bytes 8-14
 - call user data field
 - virtual circuit 8-13
 - called addresses
 - suppressing X.25 8-31
 - translating X.25 8-19
 - calling address
 - suppressing X.25 8-31
 - CAP
 - AppleTalk communication package 10-16
 - IPTalk configuration 10-44
 - carrier wait time, specifying 6-64
 - cbus-flooding command argument 21-10
 - CFM
 - FDDI MAC-level connection 6-38

-
- challenge handshake access protocol
 - See CHAP
 - Channel Service Unit/Digital Service Unit
 - See CSU/DSU
 - CHAOSnet
 - addresses 11-1
 - ARP entries 11-2
 - Cisco's implementation 11-1
 - configuration 11-2
 - debugging 11-3
 - displaying statistics 11-3
 - monitoring 11-2
 - CHAP
 - dialer map 6-67, 6-68
 - enabling on interface 6-79
 - using 6-79
 - chap authentication command 6-79, 6-89
 - character padding
 - changing 3-9
 - setting 3-9, 4-49
 - character width
 - 8-bit support 3-10, 4-4, 4-44, 4-51
 - character widths
 - global configuration commands 4-4
 - line commands 4-44
 - setting for international character sets 4-44
 - user commands 3-10
 - checkpointed database
 - clearing 13-57
 - checksum
 - AppleTalk 10-20
 - ISO CLNS 15-13
 - circuit
 - simplex Ethernet 13-39
 - circuit group
 - use in load balancing 21-30
 - class of service 23-10
 - clear arp-cache command 13-9, 13-57
 - clear bridge command 21-39
 - clear clns cache command 15-18
 - clear clns neighbors command 15-19
 - clear clns route command 15-18
 - clear host command 13-57
 - clear interface bri 0 command 6-16
 - clear interface command 6-10, 6-21, 6-28, 6-47, 6-52
 - clear ip accounting command 13-57
 - clear ip bgp command 14-35
 - clear ip route command 13-58, 14-83
 - clear line command 3-4, 13-58
 - clear request
 - retransmission timer 8-29
 - clear rif-cache command 22-8, 22-90
 - clear vines neighbor command 19-10
 - clear vines route command 19-11
 - clear x25-vc command 8-42
 - client router
 - configuring 4-20
 - CLNS
 - configuring for SMDS 8-72
 - frame relay congestion 8-55
 - clns checksum command 15-13
 - clns cluster-alias command 15-18
 - clns configuration-time command 15-11
 - clns congestion-threshold command 15-14
 - clns enable command 15-9
 - clns erpdu-interval command 15-14
 - clns es-neighbor command 15-9, 15-11
 - clns holding-time command 15-11
 - clns host command 15-16
 - clns is-neighbor command 15-9, 15-12
 - clns mtu command 15-12
 - clns net command 15-17
 - clns packet-lifetime command 15-15
 - clns rdpdu-interval command 15-15
 - clns rdpdu-mask command 15-15
 - clns route command 16-3, 16-4
 - clns route-cache command 15-13
 - clns router isis command 16-11
 - clns router iso-igrp command 16-6
 - clns routing command 15-8
 - clns send-erpdu command 15-14
 - clns send-rdpdu command 15-15
 - clns want-erpdu command 15-16
- clock rate
 - configuring on serial interface 7-2
 - speeds 7-3
- clock signal configuration 6-9
- clockrate command 7-2
- Closed User Group
 - See CUG number
- CMNS
 - Cisco support defined 8-45
 - configuration examples 8-47
 - configuration steps, routing 8-46

- debugging 8-54
- enabling 8-46
- interface subcommand summary 8-78
- LLC2 support, See LLC2
- monitoring LLC2 parameters 8-52
- monitoring traffic activity 8-51
- specifying address mappings 8-46
- virtual circuit, displaying active 8-53
- cmns enable command 8-46
- CMT
 - FDDI 6-42
- cmt connect command 6-44
- cmt disconnect command 6-44
- CMT microcode disable 6-44
- Columbia AppleTalk Package
 - See CAP
- command collection mode
 - configuring 4-41
- command interpreter, EXEC 2-8
- command summary
 - Apollo Domain global configuration 9-9
 - Apollo Domain subcommands 9-10
 - AppleTalk global configuration 10-84
 - AppleTalk interface subcommands 10-88
 - CMNS subcommands 8-78
 - DECnet global configuration commands 12-32
 - DECnet interface subcommands 12-35
 - EXEC system management 5-16
 - EXEC terminal 3-12
 - frame relay subcommands 8-79
 - global configuration commands, interface 6-83
 - interface configuration subcommands, adjusting 7-20
 - interface support EXEC command 6-90
 - interface support subcommands 6-84
 - IP global configuration 13-76
 - IP interface subcommands 13-80
 - IP router subcommands 14-106
 - IP routing global configuration 14-104
 - IP routing subcommands 14-116
 - ISO CLNS global configurations 15-31, 16-26
 - ISO CLNS interface subcommands 15-32, 16-28
 - LAPB subcommands 8-80
 - line configuration subcommands 4-57
 - LLC2 interface subcommands 24-19
 - Novell IPX global configuration 17-28
 - Novell IPX interface subcommands 17-29
 - SDLC interface subcommands 24-21
 - SMDS subcommands 8-81
 - source-route bridging global configuration 22-96
 - source-route bridging subcommands 22-100
 - STUN global configuration 23-43
 - STUN interface subcommands 23-45
 - system configuration 4-49
 - transparent bridging global configuration 21-42
 - transparent bridging subcommands 21-44
 - VINES global configuration 19-15
 - VINES interface subcommands 19-16
 - X.25 global commands 8-77
 - X.25 interface subcommands 8-82
 - XNS global configuration 20-25
 - XNS interface subcommands 20-26
- commands
 - abbreviating 2-8
 - correcting entries 2-8
 - DECnet monitoring 12-27
 - levels 2-8
 - listing 2-8
 - negating 2-11
 - upper- and lowercase 2-8
- community string
 - default access 4-33
 - SNMP access 4-32
- concurrent routing protocols 14-2
- configuration
 - erasing system information 5-14
 - global system command summary 4-49
 - line subcommand summary 4-57
 - writing system information 5-14
- configuration commands
 - abbreviating 2-11
 - correcting entries 2-11
 - entering 2-11
 - global 2-11, 2-12
 - interface 2-11, 2-12
 - line subcommands 2-11, 2-12
 - negating 2-11
 - router 2-11, 2-13
- configuration file
 - auto load 2-15
 - autoloading 4-37
 - automatic execution of 2-14
 - changing filename 4-7

-
- changing host name 4-7
 - changing network name 4-7
 - creating 2-13
 - examples 2-11
 - failing to load 2-16
 - loading 2-16, 4-9
 - through TFTP 2-16
 - network 2-15
 - setting specifications 4-7
 - configuration methods
 - auto load 2-15
 - from console 2-13
 - from nonvolatile memory 2-14
 - from remote hosts 2-15
 - configuration mode
 - entering 2-10
 - configuration register
 - processor 2-17, 4-8
 - configuration script
 - system startup 2-4
 - configure command 2-10
 - configuring datagram transport 8-36
 - configuring DECnet
 - token ring 12-12
 - congestion threshold
 - ISO CLNS 15-13
 - connect command 3-1
 - connect initiate filtering 12-14
 - Connection Management
 - See CMT
 - Connection-Mode Network Service
 - See CMNS
 - Connection-Oriented Network Service
 - See CMNS
 - connections
 - aborting 3-4
 - clearing BGP 14-35
 - disconnecting 3-4
 - displaying active 3-2, 3-6
 - displaying TCP 3-6
 - establishing restrictions 4-45
 - restricting access 13-27
 - SLIP 13-48
 - Telnet, incoming 3-5, 4-45
 - Telnet, outgoing 4-45, 13-27
 - console
 - configuring from 2-13
 - displaying debug messages 3-8
 - line, configuring 4-41
 - logging messages to 4-39
 - controller
 - autonomous switching support 13-40
 - loopback test 7-16
 - status, displaying 6-5
 - controlling traffic with access lists 17-10
 - conversion
 - DECnet Phase IV to Phase V 12-19
 - host name-to-address 13-20
 - copy flash tftp command 4-17
 - copy tftp flash command 4-13
 - core gateway EGP process
 - defining 14-43
 - corrupted network numbers, Novell IPX
 - repairing 17-3
 - cost
 - assigning to DECnet Phase IV 12-5
 - link state, OSPF 14-20
 - maximum for intraarea routing 12-8
 - OSPF, displaying link 14-98
 - redistribution, OSPF metrics 14-60
 - unequal-cost load balancing, IGRP 14-6
 - counters
 - clearing 6-3
 - SMDS 8-76
 - CPU auxiliary port
 - configuring 4-42
 - CSC-1R 6-5, 6-26, 6-27
 - CSC-2R 6-5, 6-26, 6-27
 - CSC-ENVM
 - See ENVM card
 - CSC-FCI 6-33, 7-18
 - CSC-FCIT interface card 6-33
 - CSC-MC+
 - See Flash Memory Card
 - CSC-MCI
 - See MCI card
 - CSC-MEC
 - See MEC card
 - CSC-R 6-5, 6-26
 - CSC-R card 7-18
 - CSC-R16 6-5, 6-26, 6-27
 - CSC-R16M 6-5
 - CSC-SCI
 - See SCI card

CSU/DSU
loopback 7-13, 7-15

CTRL-^
See escape character
CUG number 8-10, 8-86

D

DAS

FDDI 6-33

data link connection identifier
See DLCI

data structures
resetting 3-4

datagram
accepting unlabeled 13-31
accepting with extended security option 13-32
prioritizing 13-33
priority queuing 7-4
security options 13-32

Datagram Delivery Protocol
See DDP

datagram transport
configuring on DDN 8-36

DCE
device testing with loopback 7-15
dial-on-demand routing 6-61
serial interface appliques 7-2
use in LAPB 8-2

DCE clear request retransmission timer
setting 8-29

DCE request retransmission timer
setting 8-28

DCE reset request retransmission timer
setting 8-29

dce-terminal-timing-enable command 6-9

DDN

configuring X.25 8-17
conversion scheme 8-40
encapsulation 8-36
use of EGP 14-38
use of HDH protocol 8-41
X.25 basic service 8-36
X.25 configuration subcommands 8-41
X.25 standard service 8-36

DDP

AppleTalk routing protocol 10-2

AppleTalk well-known sockets 10-17
IP encapsulation 10-17

DDR

access list configuration examples 6-72
assigning access lists 6-70
carrier wait-time 6-64
Cisco's implementation 6-61
configuring 6-60
debugging 6-77
dialer idle time 6-63
dialer rotary groups 6-69
monitoring routing 6-76
multiple destinations 6-66
setting interface idle time 6-63
specifying dialer enable time 6-64
specifying dialer type 6-62
specifying multiple destinations 6-67
specifying single DDR telephone number 6-65
specifying strings passed to DCE 6-65
specifying V.25 bis dialing 6-62
using access lists with 6-71
V.25 bis conformance 6-61

DDR implementation 6-61

DDR interface

assigning access lists 6-70
using access lists 6-71

debug ? command 5-11

debug all command 5-11

debug apollo-packet command 9-9

debug apollo-routing command 9-9

debug apple-arp command 10-83

debug apple-errors command 10-83

debug apple-event command 10-83

debug apple-nbp command 10-83

debug apple-packet command 10-83

debug apple-routing command 10-84

debug appletalk command 10-83

debug apple-zip command 10-84

debug arp command 8-76, 13-75

debug broadcast command 6-25

debug chaos-packet command 11-3

debug chaos-routing command 11-3

debug clns-esis-events command 15-30

debug clns-events command 15-31

debug clns-igrp-packets command 16-26

debug clns-packets command 15-31

debug clns-routing command 15-31

debug command 5-11

debug decnet-packets command 12-32

debug decnet-routing command 12-32

debug fddi-cmt-events command 6-41

debug fddi-smt-packets command 6-41

debug frame-relay events command 8-65

debug frame-relay-lmi command 8-65

debug frame-relay-packets command 8-65

debug hdh command 8-41

debug ip-bgp command 14-102

debug ip-bgp-events command 14-102

debug ip-bgp-updates command 14-103

debug ip-egp command 14-103

debug ip-egp-events command 14-103

debug ip-hello command 14-103

debug ip-icmp command 13-75

debug ip-igrp command 14-103

debug ip-ospf-adj command 14-103

debug ip-ospf-events command 14-103

debug ip-ospf-flood command 14-103

debug ip-ospf-packets command 14-103

debug ip-ospf-spf command 14-104

debug ip-packet command 13-75

debug ip-rip command 14-104

debug ip-routing command 13-75, 14-104

debug ip-tcp command 13-75, 14-104

debug ip-tcp-packet command 14-104

debug ip-udp command 13-75, 14-104

debug isis-adj-packet command 16-26

debug isis-spf-events command 16-26

debug isis-update-packets command 16-26

debug lapb command 8-5, 8-41

debug lat command 21-42

debug llc2-events command 24-10

debug llc2-packets command 24-10

debug llc2-test command 24-10

debug lnm-events command 22-83

debug lnm-llc command 22-83

debug lnm-mac command 22-83

debug local-ack command 22-33, 25-21

debug local-ack state command 22-33

debug messages

- displaying on terminal or console 3-8

debug mop command 3-11

debug novell-packet command 17-27

debug novell-routing command 17-27

debug novell-sap command 17-28

debug packet command 6-26

debug probe command 13-76

debug psn command 8-42

debug psn-events command 8-42

debug pup-packet command 18-4

debug pup-routing command 18-4

debug rif command 22-93

debug sdlc command 24-19

debug sdlc-local-ack command 23-20

debug sdllc command 25-21

debug serial-interface command 6-14, 6-19, 6-51, 6-77, 6-80, 8-76

debug serial-packet command 6-14, 6-19, 6-78

debug slip command 13-76

debug slip-event command 13-76

debug source-bridge command 22-94

debug source-event command 22-96

debug span command 21-41

debug stun command 23-43

debug stun-packet command 23-43

debug token-event command 6-32, 22-96

debug token-ring command 6-32, 22-96

debug vines-arp command 19-14

debug vines-echo command 19-14

debug vines-packet command 19-14

debug vines-routing command 19-14

debug vines-table command 19-14

debug x25 command 8-44, 8-54

debug x25-events command 8-44, 8-54

debug x25-vc command 8-44, 8-54

debug xns-packet command 20-25

debug xns-routing command 20-25

debugging

- Apollo Domain network 9-9
- AppleTalk 10-82
- ARP transactions 13-75
- CHAOSnet 11-3
- CMNS 8-54
- DECnet 12-31
- diagnostics 5-11
- dial-on-demand routing (DDR) 6-77
- Ethernet interface 6-25
- FDDI 6-41
- frame relay 8-65
- IP network 13-74
- IP routing 14-102
- IPSO 13-35

- ISO CLNS 15-31
 - network 15-30
 - routing protocols 16-26
- LAPB 8-5
- Novell IPX network 17-27
- PUP 18-4
- real-time support 5-11
- serial interface 6-14, 6-51
- SLIP 13-76
- SMDS 8-76
- source-route bridging 22-93
- STUN 23-43
- timestamping 5-11
- Token Ring interface 6-32
- transparent bridging 21-41
- use of trace command 5-14
- VINES network 19-14
- X.25 8-44
- XNS network 20-24
- DEC MOP
 - server 3-11
 - See also MOP
- DECnet
 - ATG limitations 12-27
 - Cisco implementation 12-1
 - cluster aliases, configuring 15-18
 - configuration examples 12-22
 - configuring on Token Ring 12-12
 - configuring routing 12-4
 - connect initiate filtering 12-14
 - debugging the network 12-31
 - defaults altering 12-10
 - designated router 12-22
 - specifying 12-22
 - displaying address mapping information 12-28
 - displaying routing table 12-28
 - displaying status 12-27
 - displaying traffic statistics 12-29
 - enabling routing 12-4
 - establishing routing 12-22
 - global interface command summary 12-32
 - interface subcommand summary 12-35
 - level 1, level 2 routing 12-22
 - maximum address space 12-22
 - monitoring commands 12-27
 - Phase IV, frame relay congestion 8-55
 - routing over frame relay 8-62
 - setting interfaces 12-22
 - decnet access-group command 12-18
 - decnet area-max command 12-7
 - decnet area-max-cost command 12-7
 - decnet area-max-hops command 12-8
 - decnet command 12-24
 - decnet conversion command 12-20
 - decnet cost command 12-5
 - decnet encapsulation command 12-13
 - DECnet encapsulation over Token Ring 12-13
 - decnet hello-timer command 12-11
 - decnet in-routing-filter command 12-18
 - decnet map command 12-24
 - decnet max-address command 12-7
 - decnet max-area command 12-7
 - decnet max-cost command 12-8
 - decnet max-hops command 12-9
 - decnet max-paths command 12-10
 - decnet max-visits command 12-9
 - decnet node-type command 12-6
 - decnet out-routing-filter command 12-19
 - decnet path-split-mode interim command 12-10
 - decnet path-split-mode normal command 12-10
 - DECnet Phase IV
 - access groups 12-18
 - access lists 12-13, 12-14
 - addresses 12-2
 - adjusting timers 12-11
 - altering defaults 12-10
 - assigning the cost 12-5
 - configuring maximum visits 12-9
 - configuring path selection 12-9
 - converting to DECnet Phase IV 12-19
 - disabling fast switching 12-11
 - equal cost paths 12-10
 - interarea routing 12-7
 - intra-area routing 12-8
 - maximum node address 12-7
 - on Token Ring 12-12
 - parameters 12-3
 - restrictions for using 12-1
 - route cache 12-11
 - routing 12-4
 - routing protocol 12-1
 - specifying area sizes 12-6
 - specifying designated router 12-12
 - specifying node 12-6

DECnet Phase IV/Phase V
 conversion 12-19
 conversion example 12-23
 designing network to support both 12-21

decnet route-cache command 12-11

decnet router-priority command 12-12

decnet routing command 12-4

decnet routing-timer command 12-11

default command 7-8

default network
 generating 14-54

default routes
 generating IP 14-54
 generating OSPF 14-61
 picking 14-55

default values for minor keywords 13-33

default-information allowed command 14-58

default-information originate command 14-43

default-information originate metric command 14-61

default-metric command 14-58

default-value exec-character-bits command 4-4, 4-51

default-value special-character-bits command
 4-4, 4-51

Defense Data Network
 See DDN

define module configurator command 3-11

delay
 backup line 6-58
 setting on interface 7-11

delay command 7-12

delimiting character
 in banner messages 4-2

description command 6-2

descriptive name
 adding to interface 6-2

designated routers
 IS-IS, specifying election 16-12
 OSPF 14-14
 specifying for DECnet Phase IV 12-12

destination addresses
 administrative filtering 21-24, 22-50

diagnostics
 enabling output 5-11

dial backup
 configuration examples 6-59
 line, configuring 6-57
 service, configuring 6-57

dialer enable-timeout command 6-64

dialer fast-idle command 6-63

dialer idle-timeout command 6-63

dialer in-band command 6-62

dialer map command 6-67, 6-86

dialer map commands 6-67, 6-86

dialer rotary groups 6-69

dialer string command 6-65

dialer type
 specifying 6-62

dialer wait-for-carrier-time command 6-64

dialer-group command 6-70

dialer-list command 6-71

dialer-rotary command 6-69, 6-86

dial-on-demand routing
 access-list examples 6-72
 configuring 6-60
 debugging 6-77
 See also DDR

Digital Equipment Corporation (DEC) 12-1

direct encapsulation
 remote source-route bridging with 22-22

disconnect command 3-4

displaying
 incoming message banner 4-3
 message-of-day 4-2

distance bgp command 14-33

distance command 14-50, 16-8

Distributed Computer Network project 14-27

distribute-list command 14-47, 14-49

distribution list
 definition of AppleTalk 10-32

DLCI
 defining for multicast 8-60
 mapping 8-57
 setting local 8-60

DNS
 dynamic name lookup 13-20
 using 2-16

document conventions xlvi

domain
 ISO CLNS 15-3

domain list
 defining 13-22
 establishing IP 13-45

Domain Name System
 See DNS

Domain Specific Part

See DSP

dotted-decimal address notation for IP 13-2

DSP

NSAP address 15-5

DTE

loopback to 7-15

use in LAPB 8-2

dte-invert-txc command 6-9

DTR

signal pulsing 7-2

Dual Attach Stations

See DAS

dual homing

FDDI 6-45

dynamic address assignment

AppleTalk 10-7

dynamic buffer sizing 4-6

dynamic entries

clearing from ARP cache 13-57

dynamic name lookup

configuring 13-20

E

early-token-release command 6-27

configuring 6-27

Echo message

ICMP 13-19

EGP

adjusting timers 14-41

backup router 14-42, 14-76

configuring third-party support 14-42

core gateway, defining 14-43

creating routing process 14-39

default route, generating 14-43

definition of 14-2, 14-38

displaying statistics 14-88

neighbor, accepting any 14-44

network to advertise 14-39

redistribution 14-56

specifying autonomous system number 14-39

specifying list of neighbors 14-39

support in router 1-2

third-party support 14-76

enable command 2-8, 2-10, 4-22

enable last-resort command 4-28

enable password command 4-22

enable use-tacacs command 4-27

enable-password command 2-8

enabling 22-4

encapsulation

AppleTalk IPtalk 10-16

DDN X.25 8-36

DECnet 12-13

Ethernet interface 6-21

FDDI 6-34

frame relay 8-56

HDH protocol 8-41

HDLC 6-10

HSSI 6-47

LAPB 8-2

PPP 6-78

serial interface 6-9

serial, See direct encapsulation

SMDS 8-68

STUN 23-7

Token Ring interface 6-28

UltraNet 6-52

VINES 19-6

X.25 8-7

XNS 20-2, 20-6

encapsulation bfex25 command 8-37

encapsulation command 6-9, 6-15, 6-21, 8-2, 8-36

encapsulation frame-relay command 8-56

encapsulation hdh command 8-41

encapsulation hdlc command 6-10

encapsulation ppp command 6-78

encapsulation sdlc-primary 24-21

encapsulation sdlc-primary command 24-11

encapsulation sdlc-secondary command 24-11

encapsulation smds command 8-68

encapsulation stun command 23-7

encapsulation x25 command 8-7

encapsulation x25-dce command 8-8

encrypting passwords 4-24

encryption

BFE 8-36

End System-Intermediate System

See ES-IS

ENVM card

automatic shutdown message 5-9

environmental shutdown 5-10

environmental warning messages 5-9

-
- last measured environmental value 5-10
 - equal cost paths
 - configuring 12-10
 - ERPDU
 - configuring support 15-14
 - configuring to send 15-14
 - determining interval 15-14
 - ISO CLNS 15-14
 - error count reset frequency
 - setting 7-12
 - error logging conditions
 - displaying syslog 5-7
 - error messages
 - duplicate IP addresses 13-7
 - ICMP 13-35
 - logging 4-38
 - redirecting 4-38
 - setting levels 4-39
 - error protocol data unit
 - See ERPDU
 - errors
 - frame-copied 22-69
 - error-threshold command 7-12
 - ES
 - ISO CLNS 15-11, 15-26
 - static configuration 15-11
 - escape character
 - Break key 4-47
 - setting system 4-46
 - escape sequence
 - Telnet connection 3-2
 - escape-character command 3-2, 4-46
 - ES-IS
 - ISO CLNS 15-1, 15-11, 15-24
 - Ethernet
 - administrative filtering on 21-17
 - configuring loopback server 7-18
 - filtering encapsulated packets 21-20, 21-21
 - media type command 6-20
 - SDLLC support 25-9
 - squelch command 6-20
 - transparent bridging 21-36
 - Ethernet cards
 - loopback on 7-17
 - Ethernet interface
 - cards 6-20
 - clearing 6-21
 - configuring 6-20
 - debugging 6-25
 - encapsulation methods 6-21
 - loopback on 7-17
 - maintaining 6-21
 - monitoring 6-22
 - Novell IPX encapsulation 17-4
 - resetting hardware logic 6-21
 - show interfaces field descriptions 6-22
 - special routing techniques 13-39
 - specifying 6-20
 - support 6-20
 - Ethernet to Internet
 - example 13-26
 - ethernet-transit-oui command 22-42
 - EXEC
 - system management, command summary 5-16
 - EXEC commands
 - displaying 2-8
 - entering 2-10
 - help 2-8
 - multiple screen output 2-8
 - syntax 2-8
 - terminal command summary 3-12
 - using command interpreter 2-8
 - exec-banner command 4-45
 - exec-character-bits command 4-44, 4-58
 - exec-timeout command 4-48
 - exit command 3-3
 - explorer packets
 - configuring 22-12
 - spanning 22-12
 - extended access lists
 - configuring IP 13-25
 - configuring XNS 20-16
 - DECnet Phase IV 12-14
 - Novell IPX 17-9
 - extended networks
 - using secondary addresses 14-53
 - extended TACACS mode
 - enabling 4-28
 - Exterior Gateway Protocol
 - See EGP
 - exterior gateway protocols
 - See EGP
 - See also exterior routing protocols

exterior routing protocols

configuring 14-4

See also EGP

external bridge

Cisco router as 17-1

Novell IPX 17-1

F

Fast Sequenced Transport

performance considerations 22-22

remote source-route bridging with 22-19

setting up peer name 22-19

fast switching

and source-route bridging 22-4

AppleTalk invalidation conditions 10-63

clearing IP cache 13-57

DECnet Phase IV 12-11

disabling IP 13-40

disabling ISO CLNS packet 15-13

disabling Novell packet 17-3

displaying cache for AppleTalk 10-63

displaying enabled ISO CLNS 15-24

displaying enabled Novell packet 17-24

enabling IP 13-40

enabling Novell packet 17-3

ISO CLNS packet 15-13

Novell IPX 17-22

SMDS

configuring 8-73

XNS 20-5

See also switching

FDDI

bit specifications 6-43

bridging configurations 6-34

controlling CMT microcode 6-44

controlling transmission time 6-42

CSC-FCI card 6-33

debugging 6-41

determining bandwidth 6-41

disconnecting 6-44

dual homing 6-45

encapsulation methods 6-34

encapsulation mode compatibility 6-34

FDDI Station Management (SMT) 6-46

frame contents 6-33

loopback on CSC-FCI card 7-18

monitoring 6-34

ring scheduling 6-41

setting bit control 6-42

setting SMT queue size 6-45

show interfaces field descriptions 6-35

signal bits 6-42

special commands 6-41

specifying 6-33

starting 6-44

stopping 6-44

support 6-33

transit bridging 21-5

FDDI CMT microcode control 6-44

fddi cmt-plus 6-44

fddi cmt-signal-bits command 6-42

fddi encapsulate 6-34

fddi token-rotation-time command 6-41

fddi valid-transmission-time command 6-41

fddi-tl-min-time command 6-42

FECN 8-55

Fiber Distributed Data Interface

See FDDI

fiber-optic cable

FDDI designations for 6-44

file loading

configuration 2-15, 4-9

filenames

changing default 4-7

changing host configuration 4-7

changing network configuration 4-7

filter

administrative for source-route bridging 22-45

administrative for transparent bridging 21-16

bytes access list 22-60

DECnet Phase IV routing 12-18

destination addresses 21-24, 22-50

IEEE 802.3-encapsulated packets 21-21, 21-22

IEEE 802.5-encapsulated packets 22-47

incoming information 14-49

interface updates 14-45

IP accounting 13-36

LAT service announcements 21-25

network updates 14-47

Novell IPX packets 17-8

Novell IPX, access lists 17-8

Novell IPX, establishing input filters 17-12

Novell IPX, establishing output filters 17-13

-
- Novell IPX, establishing router filters 17-13
 - Novell IPX, outgoing traffic 17-9
 - Novell IPX, routing updates 17-12
 - Novell IPX, SAP filters 17-13, 17-15
 - Novell IPX, SAP input filter
 - example 17-15
 - Novell IPX, SAP output filter
 - example 17-17
 - outgoing information (IP) 14-45
 - point-to-point updates 14-48
 - received updates 14-49
 - routing information 14-45
 - source addresses 21-23, 22-49
 - sources of routing information 14-50
 - station access list 22-57
- filtering
- administrative, See administrative filtering
 - DECnet connect initiate 12-14
 - examples 12-17
 - IEEE 802 frames 22-47
 - SNAP frames 22-47
- Finger protocol 4-37
- flapping
- routing problems 14-69
- Flash Memory
- automatically booting from 4-16
 - booting system software 4-9
 - configuring 4-11
 - copying image 4-17
 - copying TFTP image 4-13
 - displaying statistics 4-11
 - fault-tolerant strategy 4-18
 - manually booting from 4-17
 - saving configuration first 4-11, 4-13
 - security precautions 4-10
 - statistics 4-11
 - displaying 5-10
 - verifying installation 5-10
 - storing system software 4-9
- Flash Memory card
- also known as CSC-MC+ 4-9
 - using 2-18
 - verifying installation 4-11
- Flash Memory statistics
- verifying installation 4-11
- Flash ROM statistics
- displaying 4-11
 - verifying installation 4-11
- flooding
- in XNS, definition 20-11
- flow control modulus
- setting 8-30
- forward
- broadcast packets 13-13
 - delay interval 21-14
 - Novell IPX broadcast to address 17-19
- forward delay interval 21-14
- Forward Explicit Congestion Notification
- See FECN
- forwarding database
- viewing entries in 21-39
- frame
- setting parameters 8-4
- frame conversion
- between source-route and transparent bridging 22-44
 - used in SR/TLB A-1
 - used in SRT A-1
- frame relay
- configuring for ISO CLNS 8-58
 - FECN 8-55
- frame relay
- configuring 8-54
 - examples 8-61
 - congestion information 8-55
 - debugging 8-65
 - disabling split horizon 14-64
 - displaying global statistics 8-64
 - displaying map entries 8-64
 - enabling ANSI LMI 8-56
 - encapsulation, specifying 8-56
 - hardware configuration 8-55
 - IP split horizon, default 8-55
 - keepalive timer 8-57
 - local DLCI 8-60
 - map entries 8-64
 - monitoring 8-62
 - multicast DLCI 8-60
 - netbooting over 8-61
 - short status messages 8-59
 - static routing example 8-61
 - subcommand summary 8-79

- frame size
 - maximum source-route bridge 22-16, 22-20, 22-23, 22-26
- frame-copied errors
 - Token Ring 22-69
- frame-relay bridging
 - configuration examples 21-34
 - configuring 21-33
 - with multicasts 21-34
 - with no multicasts 21-33
- frame-relay keepalive command 8-57
- frame-relay lmi-type command 8-56
- frame-relay local-dci command 8-60
- frame-relay map bridge command 8-59, 21-33
- frame-relay map clns command 8-58
- frame-relay map command 8-57
- frame-relay multicast-dlci command 8-60
- frame-relay short-status command 8-59
- FST
 - See Fast Sequenced Transport
- Fuzzball gateways 14-27

G

- gateway
 - definition of 14-1
 - last resort 14-8
- Gateway Discovery Protocol
 - See GDP
- GDP
 - changing parameters 14-72
 - commands 14-72
 - description of 14-70
 - messages 14-70
 - query message 14-70
 - report message 14-70
- global broadcast address 13-4
- global command summary
 - SDLLC 25-21
- global configuration command summary
 - Apollo Domain 9-9
 - AppleTalk 10-84
 - DECnet 12-32
 - IP 13-76
 - IP routing 14-104
 - ISO CLNS 15-31, 16-26
 - Novell IPX 17-28

- packet-switched software 8-77
- source-route bridging 22-96
- STUN 23-43
- transparent bridging 21-42
- VINES 19-15
- X.25 8-77
- XNS 20-25
- global configuration commands 2-12
 - entering 2-11
 - username name 4-30
- global system configuration
 - command summary 4-49
- global system parameters
 - configuring 4-1
- GOSIP
 - NSAP format 15-7

H

- hdh command 8-41
- hdh message command 8-41
- hdh packet command 8-41
- HDH protocol, using 8-41
- HDLC
 - connected to Cisco routers 23-4
 - serial encapsulation method 6-10
 - using TCP transport mechanism 23-4
- HDLC Distant Host (HDH) protocol
 - See HDH protocol
- header
 - Internet, configuring options 13-19
 - ISO CLNS options 15-16
- header compression
 - TCP 13-41, 13-65
- Hello
 - DPDU interval 21-13
 - packets, Net/One 20-8
 - packets, Ungermann-Bass 20-8
 - specifying, ISO CLNS 15-11
- Hello Protocol
 - configuring 14-27
 - creating the routing process 14-28
 - definition of 14-2
 - description (OSPF) 14-13
 - displaying list of networks 14-28
 - metric transformations 14-56

- redistribution 14-56
 - example 14-75
- help command 2-8
- helper address
 - control broadcasts 17-19
 - IP 13-44
 - Novell IPX 17-19
 - PUP 18-3
- helper list
 - control broadcasts 17-19
 - defining for Novell IPX 17-18
- helping
 - in XNS, configuring 20-13
 - in XNS, definition 20-11
- High-Speed Serial Interface
 - See HSSI
- hold queue 7-10
 - SLIP 13-53
- holddown
 - disabling 14-67
- hold-queue command 7-10
- hop
 - definition 13-5
- hop count
 - DECnet Phase IV 12-8
 - RIP 14-26
 - setting for IGRP 14-67
 - setting for interarea routing 12-8
- host
 - displaying statistics 13-60
 - on a network segment 13-45
 - writing configuration to 5-14
- host configuration file 4-7
 - changing name 4-7
- host name
 - assigning 4-1
 - setting 4-1, 4-25
- hostname command 2-8, 4-1
- host-name-and-address cache
 - clearing entries 13-57
- host-name-to-address
 - conversion 13-20
- HP Advancenet
 - See HP Probe
- HP hosts
 - on network segment 13-45
- HP Probe

- address resolution 13-10
- proxy requests 13-21
- HSCI card
 - loopback test 7-16
- HSSI
 - clearing 6-47
 - configuring internal loop on 7-14
 - debugging 6-51
 - encapsulation methods 6-47
 - loopback on 7-14
 - loopback, externally requested 7-16
 - maintaining 6-47
 - monitoring 6-48
 - specifying 6-47
 - support 6-46
- hssi external-loopback-request command 7-16

I

- IBM
 - 3174, frame-copied errors 22-69
 - 8209 bridges, and SR/TLB routers 22-42
 - network, extending with STUN 23-27
 - PC/3270 emulation, and source-route bridging 22-68
 - using SR/TLB in LLC2 environments 22-43
- ICMP 14-2
 - configuring 13-16
 - customizing services 13-44
 - error messages 13-35
 - redirect messages 13-17
 - subnet masks 13-7
 - unreachable messages 13-17
- ICMP Router Discovery Protocol
 - See IRDP values
- ICP
 - VINES 19-8
- idle time
 - setting 6-63
- IDP
 - NSAP address 15-5
- IEEE 802 frames on output
 - filtering 22-47
- IEEE 802.2
 - LLC encapsulation 6-21
- IEEE 802.3
 - encapsulation 6-21

-
- IEEE 802.5
 - administrative filtering 22-47
 - committee 22-2
 - Token Ring media 6-26
 - ignore authority field
 - security 13-31
 - ignore VC timer
 - configuring 8-27
 - IGP 14-2
 - IGRP
 - configuring 14-5
 - creating the routing process 14-5
 - definition of 14-2
 - determining route feasibility 14-7
 - displaying list of networks 14-6
 - exterior routing 14-5
 - metric adjustments 14-66
 - metric information 14-8
 - redistribution 14-57, 14-75
 - routes 14-5
 - setting hop count 14-67
 - support in router 1-2
 - system routes 14-5
 - unequal-cost load balancing 14-6
 - update broadcasts 14-8
 - variance command 14-7
 - implicit masks 13-24
 - incoming information
 - filtering 14-49
 - Initial Domain Part
 - See IDP
 - in-routing filter
 - configuring 12-18
 - interarea routing
 - maximum route cost, DECnet Phase IV 12-7
 - setting hop count 12-8
 - interdomain
 - routing, description 15-3
 - interface
 - adding descriptive name 6-2
 - addresses, multiple 14-53
 - addresses, secondary 14-53
 - AppleTalk subcommand summary 10-88
 - assigning path costs 21-15
 - assigning priority group 7-9
 - assigning queuing priority 7-7
 - assigning to spanning-tree group 21-10
 - clearing counters 6-3
 - configuration subcommand summary 7-20
 - configuring IS-IS 16-11
 - configuring STUN 23-7
 - displaying AppleTalk-specific information 10-65
 - displaying controller status 6-5
 - displaying information about 6-4
 - displaying Novell IPX parameters 17-25
 - displaying settings for VINES 19-11
 - displaying statistics 6-7
 - displaying system configuration 6-4
 - displaying Token Ring statistics 22-93
 - error count frequency, setting 7-12
 - Ethernet, See Ethernet interface
 - forwarding STUN frames 23-9
 - global command summary, interface support 6-83
 - hold queues 7-10
 - HSSI, See HSSI
 - IP, See IP interface
 - ISO CLNS-specific 15-23
 - loopback on Ethernet 7-17
 - loopback to DTE 7-15
 - monitoring, frame relay 8-62
 - null 6-82
 - parameters and statistics, displaying 8-42
 - placing in a STUN group 23-8
 - priority queuing 7-4
 - restarting 6-3
 - restricting access to 13-28
 - serial processing on IP 13-38
 - serial, See serial interface
 - setting bandwidth on 7-11
 - setting delay value 7-11
 - setting IP addresses 13-7
 - setting priority for bridging 21-16
 - settings, overriding 8-33
 - shutting down 6-3
 - specifying 6-1
 - subcommand summary, interface support 6-84
 - testing 5-15
 - Token Ring, See Token Ring and Token Ring interface
 - UltraNet, See UltraNet interface
 - unit numbers 6-2
 - usability status 14-52
 - usable, definition of 14-52

-
- VINES access list 19-9
 - X.25 8-42
 - interface access
 - controlling 13-28
 - interface cards
 - CSC-HSA 6-47
 - CSC-HSCI 6-47, 6-52
 - CSC-ULA 6-52
 - MCI 6-20
 - MEC 6-20
 - interface command 6-2, 7-7
 - interface configuration
 - global command summary 7-19
 - interface dialer command 6-69
 - interface ethernet command 6-20
 - interface fddi command 6-33
 - interface hssi command 6-47
 - interface serial command 6-8
 - interface subcommand summary
 - SDLLC 25-21
 - interface tokenring command 6-26, 12-12
 - interface ultranet command 6-52
 - interface, global command summary
 - interface characteristics 7-19
 - interface, subcommand summary
 - Apollo Domain 9-10
 - CMNS 8-78
 - DECnet 12-35
 - frame relay 8-79
 - interface characteristics 7-20
 - IP 13-80
 - IP routing 14-116
 - ISO CLNS 15-32, 16-28
 - LAPB 8-80
 - Novell IPX 17-29
 - packet-switched software 8-78
 - SMDS 8-81
 - source-route bridging 22-100
 - STUN 23-45
 - transparent bridging 21-44
 - VINES 19-16
 - X.25 8-82
 - XNS 20-26
 - interfaces
 - adjusting characteristics 7-3
 - interior gateway protocols
 - See IGP
 - See also interior routing protocols
 - Interior Gateway Routing Protocol
 - See IGRP
 - See also IGP
 - interior route
 - IGRP 14-5
 - interior routing protocols
 - configuring 14-4
 - See also IGP
 - Intermediate System-Intermediate System
 - See IS-IS
 - internal buffer
 - message logging 4-38
 - internal loop
 - on HSSI applique 7-14
 - international characters
 - setting default widths 4-4
 - setting widths at EXEC level 3-10
 - Internet
 - configuring header options 13-19
 - Internet address
 - allowable 13-4
 - assigning 13-2
 - assigning to SLIP 13-51
 - broadcast addresses 13-11, 13-12
 - Class A 13-3
 - Class B 13-3
 - Class C 13-4
 - Class D 13-4
 - Class E 13-4
 - classes of 13-3
 - conventions 13-4
 - displaying mapped SLIP 13-67
 - dotted decimal 13-2
 - finding 13-11
 - multiple 14-52, 14-53
 - notation 13-2
 - obtaining 13-3
 - resolution using ARP 13-8
 - restricting access 13-27
 - secondary 14-53
 - setting 13-7
 - SLIP 13-52
 - special 13-4
 - Internet addresses 13-2
 - Internet Control Message Protocol
 - See ICMP

-
- Internet Control Protocol
 - See ICP
 - Internet routing protocols
 - supported 14-1
 - Interprocess Communications
 - See IPC
 - interval
 - forward delay 21-14
 - Hello BPDUs 21-13
 - Maximum idle 21-14
 - intraarea routing
 - maximum route cost, DECnet Phase IV
 - 12-7, 12-8
 - setting hop count 12-9
 - invert serial clock signal 6-9
 - IP
 - assigning addresses 13-2
 - broadcast flooding 13-15, 13-43
 - broadcast forwarding 13-13
 - configuring for SMDS 8-72
 - displaying SLIP ARP cache 13-67
 - establishing domains 13-45
 - global configuration command summary 13-76
 - implementation overview 13-1
 - ping command 13-70
 - special configurations 13-38
 - split horizon, enabling and disabling 14-64
 - static routing redistribution 14-74
 - trace command 13-71
 - ip access-group command 13-28
 - IP accounting
 - clearing checkpointed database 13-57
 - configuring 13-36
 - controlling transit records 13-37
 - disabling on outbound transit traffic 13-36
 - displaying 13-59
 - enabling on outbound transit traffic 13-36
 - maximum entries 13-36
 - specifying filters 13-36
 - threshold 13-36
 - transit records 13-37
 - ip accounting command 13-36
 - ip accounting-list command 13-36
 - ip accounting-threshold command 13-36
 - ip accounting-transits command 13-37
 - IP address
 - See Internet address
 - ip address command 13-7, 14-53
 - ip as-path access-list command 14-32
 - ip broadcast-address command 13-12
 - ip configuration examples 13-42
 - ip default-network command 14-54, 14-55
 - ip directed-broadcast command 13-12
 - ip domain-list command 13-22
 - ip domain-lookup command 13-21
 - ip domain-name command 13-21
 - ip forward-protocol nd command 13-14
 - ip forward-protocol spanning-tree command 13-15
 - ip forward-protocol udp command 13-14
 - ip gdp command 14-72
 - ip gdp holdtime command 14-72
 - ip gdp priority command 14-72
 - ip gdp reporttime command 14-72
 - ip helper-address command 13-13
 - ip host command 13-20
 - ip hp-host command 13-22
 - IP IEN-116 name server
 - specifying 13-21
 - IP interface
 - disabling processing 13-7
 - displaying statistics 13-62
 - multilevel security 13-30
 - multiple addresses 13-7
 - restricting access 13-28
 - secondary addresses 13-7
 - setting addresses 13-7
 - setting default metrics 14-58
 - subcommand summary 13-80
 - suppressing updates on 14-45
 - unnumbered 13-38
 - using subnet zero address 13-7
 - ip ipname-lookup command 13-21
 - ip irdp command 14-73
 - ip mask-reply command 13-17
 - ip mtu command 13-18
 - ip name-server command 13-20
 - IP network
 - debugging 13-74
 - displaying IP show commands 13-58
 - maintaining 13-57
 - monitoring 13-58
 - ip ospf authentication-key command 14-22
 - ip ospf cost command 14-20
 - ip ospf dead-interval command 14-22

-
- ip ospf hello-interval command 14-21
 - ip ospf priority command 14-21
 - ip ospf retransmit-interval command 14-20
 - ip ospf transmit-delay command 14-21
 - ip probe proxy command 13-22
 - ip processing
 - on serial interface 13-38
 - ip proxy-arp command 13-10
 - ip redirects command 13-17
 - ip route command 14-54, 14-63
 - ip route-cache cbus command 13-41
 - ip route-cache command 13-40, 13-41
 - IP routing
 - adjustable timers 14-69
 - and bridging 21-10
 - configuration examples 13-42
 - configuring 13-1
 - debugging 14-102
 - disabling 21-10
 - displaying routing table 14-99
 - enabling 13-2
 - global configuration command summary 14-104
 - interface subcommands 14-116
 - keepalive timers 14-68
 - maintaining operations 14-83
 - monitoring operations 14-83
 - outgoing information, filtering 14-45
 - over simplex Ethernet interface 13-39
 - subcommand summary 14-106
 - ip routing command 13-2
 - IP routing processes
 - maximum number 14-3
 - IP routing protocols
 - configuration 14-74
 - displaying parameters, status 14-89
 - IP routing table
 - displaying 13-63
 - ip security add command 13-32
 - ip security command 13-30
 - ip security dedicated command 13-30
 - ip security extended-allowed command 13-32
 - ip security first command 13-33
 - ip security ignore-authorities command 13-31
 - ip security implicit-labelling command 13-31
 - ip security multilevel command 13-30
 - IP Security Option
 - See IPSO
 - ip security strip command 13-32
 - ip show commands
 - displaying 13-58
 - ip source-route command 13-38
 - IP split horizon
 - frame relay default 8-55
 - SMDS default 8-66
 - X.25 default 8-6
 - ip split-horizon command 14-64
 - ip subnet-zero command 13-7
 - ip tcp compression-connections command 13-42
 - ip tcp header-compression command 13-41
 - ip udp command 7-6
 - ip unnumbered command 13-38
 - ip unreachable command 13-17
 - IPC
 - VINES 19-8
 - IPSO
 - configuring 13-28
 - debugging 13-35
 - default keyword values table 13-33
 - definitions 13-29
 - disabling 13-30
 - minor keywords 13-33
 - security actions table 13-35
 - setting security classifications 13-30
 - IPSO configuration
 - examples 13-33
 - IRDP values 14-74
 - ISDN
 - dialer map 6-67
 - encapsulation 6-15
 - maintaining 6-16
 - ISDN BRI
 - debugging 6-19
 - encapsulation 6-15
 - maintaining 6-16
 - monitoring 6-16
 - show interfaces field descriptions 6-16
 - specifying 6-15
 - IS-IS
 - areas, multihoming 15-7
 - database, displaying 16-24
 - designated router election 16-12
 - enabling on an interface 16-11
 - enabling routing 16-9
 - Level 1 routing table, displaying 16-22

- link state metrics, configuring 16-11
 - network entity titles, configuring 16-9
 - password, configuring 16-12
 - redistributing routes 16-10
 - router support, specifying level 16-10
 - routing examples 16-13
 - routing, configuring 16-8
 - specifying desired adjacency 16-12
 - support in router 1-2
 - isis circuit-type command 16-12
 - isis metric command 16-11
 - isis password command 16-12
 - isis priority command 16-12
 - IS-IS routing
 - configuration 16-11, 16-19
 - examples 16-19, 16-20
 - ISO CLNS
 - addresses 15-4
 - areas 15-2
 - basic static routing example 16-13
 - Cisco's implementation 15-1
 - clearing cache 15-18
 - configuring 15-4, 16-2
 - configuring checksums 15-13
 - configuring ISO-IGRP 16-4
 - configuring over X.25 15-9
 - configuring overlapping areas 16-17
 - configuring performance parameters 15-12
 - configuring static routing 16-2
 - congestion threshold 15-13
 - debugging routing protocols 16-26
 - debugging the network 15-30
 - DECnet cluster aliases, configuring 15-18
 - disabling ERPDU 15-14
 - disabling fast switching 15-13
 - displaying ES neighbors 15-26
 - displaying general information 15-19
 - displaying IS neighbors 15-25
 - displaying protocol-specific information 16-22
 - displaying redirect information 15-22
 - displaying routes 16-20
 - displaying routing cache 15-20
 - displaying specific interfaces 15-23
 - displaying traffic 15-21
 - domain 15-3
 - dynamic interdomain routing 16-18
 - dynamic routing in overlapping areas 16-17
 - dynamic routing within a domain 16-16
 - enabling CLNS routing 15-8
 - enabling routing 15-8
 - ES static configuration 15-11
 - ES-IS parameters 15-11
 - fast switching 15-13
 - global configuration command summary
 - 15-31, 16-26
 - header options 15-16
 - IGRP support 15-2
 - interdomain routing example 16-15
 - interface subcommand summary 15-32, 16-28
 - intradomain static routing 16-13
 - local source packet parameters 15-15
 - maintaining the network 15-18
 - monitoring the network 15-19, 16-20
 - network architecture 15-2
 - ping command 15-27
 - protocols supported 15-1
 - redistributing static routes 16-7
 - routing in more than one area 16-16
 - routing protocols supported 16-1
 - routing, configuring 16-2
 - specifying Hello messages 15-11
 - static routing 15-2
 - static routing example 16-13
 - switching 15-2
 - trace command 15-28
 - tracing routes 15-29
 - ISO CLNS router subcommand
 - summary 16-27
 - ISO-IGRP
 - domain, redistributing routes into 16-6
 - metrics, configuring 16-6
 - network entity titles, configuring 16-5
 - routing examples 16-13
 - specifying router level 16-6
 - support in router 1-2
 - is-type command 16-10
- ## K
- keepalive command 14-68
 - keepalive timer
 - frame relay 8-57
 - IP routing 14-68

keywords

- retransmission timer 8-28
- SVC range limit 8-25

L

LAN Network Manager

- and Cisco routers 22-72
- changing reporting times 22-74
- Cisco's implementation 22-70
- Configuration Report Server 22-74
- configuring 22-75
- configuring the router 22-73
- debugging 22-83
- displaying bridge information 22-78
- displaying configuration information 22-78
- displaying station-specific information 22-81
- displaying Token Ring information 22-79
- monitoring 22-78
- Ring Error Monitor 22-74
- Ring Parameter Server 22-74

LAPB

- configuring 8-1
- debugging 8-5
- displaying statistics 8-4
- encapsulation 8-2
- encapsulation sample 8-2
- encapsulation with a single protocol 8-2
- encapsulation with multiple protocols 8-2
- interface subcommand summary 8-80
- logging transactions 8-41
- monitoring 8-4
- setting Level 2 parameters 8-3
- setting retransmission timer 8-3
- troubleshooting 8-4
- using leased serial line 8-1

lapb k command 8-4

lapb n1 command 8-4

lapb n2 command 8-4

lapb protocol command 8-2

lapb t1 command 8-3

last resort login feature

- enabling privileged mode 4-28
- setting 4-27, 4-28

LAT

- compression 21-35
- group codes 21-26

LAT service announcements

- administrative filtering 21-25
- deny conditions for LAT group codes 21-26
- group code service filtering 21-26
- permit conditions for LAT group codes 21-27

leased-line circuits

- X.25 8-3

length command 4-48

Level 2 switching (bridging) 6-26

Level 3

- X.25, monitoring 8-42
- X.25, retransmission timer 8-27

Level 3 switching

- routing 6-26

line

- backup, See dial backup
- clearing 3-4
- configuration, starting 4-41
- configuration, subcommand summary 4-57
- displaying active 3-7
- displaying SLIP status 13-68
- enabling SLIP mode 13-51
- loopback 7-15
- numbers, decimal 4-37
- numbers, octal 4-37
- numbers, specifying terminal 4-42
- password, See password
- password, specifying 4-23
- restricting access 13-27
- routing protocol, status 14-52

line access

- controlling 13-27

line aux command 4-42

line command 2-12, 4-41

link level

- restart 8-33

link state metrics

- configuring IS-IS 16-11

listing connections 3-2

LLC

- definition of 21-1

LLC2

- asynchronous balanced mode 24-2
- Cisco's implementation 24-3
- CMNS support 24-1
- command summary 24-19
- comparison with SDLC 24-2

- configuring 24-3
- debugging 24-10
- monitoring 24-8
- overview 24-2
- parameters, tuning for network performance 25-2
- station addressing 24-2
- T1 timer 24-6
- table of parameters 24-4
- llc2 ack-delay-time command 24-4
- llc2 ack-max command 24-4
- llc2 idle-time command 24-5
- llc2 keyword command 24-3
- llc2 local-window command 24-5
- llc2 n2 command 24-5
- llc2 t1-time command 24-6
- llc2 tbusy-time command 24-6
- llc2 tpf-time command 24-7
- llc2 trej-time command 24-6
- llc2 xid-neg-val-time command 24-7
- llc2 xid-retry-time command 24-7
- lnm alternate command 22-73
- lnm crs command 22-74
- lnm loss-threshold command 22-74
- lnm password command 22-73
- lnm rem command 22-74
- lnm rps command 22-74
- lnm snmp-only command 22-73
- lnm softerr command 22-75
- load balancing 14-2
 - fast switching 13-40
 - IGRP, unequal cost 14-6
 - over serial lines 21-30
- locaddr-priority command 22-35, 23-22
- locaddr-priority-list command 22-34, 23-22
- Local Acknowledgment
 - configuring for STUN interfaces 23-17
 - configuring LLC2 parameters 22-30
 - configuring SDLC parameters 23-17
 - debugging 23-20
 - displaying statistics 23-19
 - for LLC2
 - advantages of enabling 22-28
 - and NetBIOS timers 22-29
 - configuring for Token Ring interfaces 22-30
 - debugging 22-33
 - displaying statistics 22-32, 25-18
 - local-ack keyword with source-bridge remote-peer command 22-16
 - on per-remote-peer basis 22-30
 - on per-ring basis 22-31
 - overhead issues 22-29
 - overview 22-27
 - setting up 22-30
 - solving the T1 timer problem 22-28
 - source-bridge remote-peer command 22-16
 - when to use 22-3
- for SDLLC
 - displaying statistics 25-18
 - on per-STUN-peer basis 23-18
 - performance tuning for LLC2 25-2
 - performance tuning for SDLC 25-3
 - SDLC, advantages of enabling 23-17
 - SDLC, when to use 23-5
- SDLLC
 - debugging 22-33
 - displaying statistics 22-32
 - setting up 23-18
- local acknowledgment 23-15
- Local Area Transport
 - See LAT
- local source-route bridging
 - configuring 22-11
 - configuring explorer packets 22-12
 - configuring multiport source-bridges 22-13
 - configuring ring groups 22-13
 - enabling 22-11
- local termination
 - See local acknowledgment
- location command 4-47
- logging
 - displaying syslog 5-7
 - logging buffered command 4-38
 - logging command 4-40
 - logging console command 4-39
 - logging monitor command 4-39
 - logging on command 4-38
 - logging trap command 4-40
- Logical Link Control
 - See LLC
- login
 - limiting attempts 4-25
- login command 4-23
- login tacacs command 4-23

- loop circuit command 7-18
- loopback
 - DTE interface 7-15
 - Ethernet server support 7-18
 - external 7-16
 - HSCI card ribbon cable 7-16
 - HSSI externally requested 7-16
 - line 7-15
 - on CSC-FCI FDDI card 7-18
 - on HSSI 7-14
 - on MCI Ethernet card 7-17
 - on MCI serial card 7-17, 7-18
 - on MEC Ethernet card 7-17
 - on SCI serial card 7-17, 7-18
 - on serial interface 7-14
 - on Token Ring card 7-18
 - on ULA applique 7-17
 - on VMS system 7-18
 - remote CSU/DSU 7-15
 - restore interface to normal operation 7-14
 - test, description of 7-13
 - UltraNet connection 7-17
- loopback applique command 7-14
- loopback command 7-17, 7-18
- loopback dte command 7-15
- loopback line command 7-15
- loopback remote command 7-15
- loops
 - preventing bridge datagram 21-30
 - routing 14-55
- lost password
 - recovering from 4-23

M

- MAC address
 - administrative filtering by 21-16
 - and bridging 21-1
- MAC layer
 - and source-route bridging 22-1
- mac-address command 22-69
- MacIP
 - addresses and aliasing 10-24
 - clients, monitoring 10-69
 - configuration notes 10-27
 - configuration steps 10-24
 - defined 10-22

- exceptions to RFC specification 10-23
- servers
 - enabling 10-24
 - monitoring 10-66
 - specifying dynamic client addresses 10-25
 - specifying static client addresses 10-26
 - status, displaying 10-66
 - traffic, monitoring 10-69
- Maintenance Operation Protocol
 - See MOP
- Management Information Base
 - See MIB
- map
 - displaying address 8-14
 - displaying DECnet address information 12-28
 - DLCI 8-57
 - dynamic Internet-to-X.121 address 8-39
 - frame relay 8-57, 8-64
 - network-protocol-to-X.121-address 8-9, 8-14
 - protocol-to-virtual-circuit 8-13
 - PUP-to-IP 18-2
 - SMDS multicast address 8-70
 - SMDS static 8-69
 - static name-to-address 13-20
 - VINES name-to-address 19-7
- mapping
 - between address and DLCI 8-57
 - to multicast address, SMDS 8-70
- mask reply
 - requesting a reply 13-17
 - setting ICMP 13-17
- mask request
 - requesting a reply 13-17
- masks
 - implicit 13-24
- masks, displaying network 14-102
- maximum packet size
 - See MTU
- maximum route cost
 - specifying for interarea routing 12-8
 - specifying for intra-area routing 12-7
- Maximum Transmission Unit
 - See MTU
- maximum visits
 - configuring 12-9
- M-bit
 - use in X.25 8-30

-
- MCI card
 - Apollo Domain restrictions 9-1
 - loopback on Ethernet 7-17
 - loopback on serial 7-17, 7-18
 - pulsing DTR signal on 7-2
 - serial interface 6-8
 - MEC card
 - loopback on Ethernet 7-17
 - Media Access Control
 - See MAC address
 - media-type command 6-20
 - memory
 - displaying system statistics 5-3
 - testing 5-15
 - memory utilization
 - displaying 5-5
 - message logging
 - enabling 4-38
 - keywords and levels 4-39
 - to a UNIX syslog server 4-40
 - to another monitor 4-39
 - to internal buffer 4-38
 - to the console 4-39
 - message queue length
 - SNMP server 4-33
 - message-of-the-day banner
 - displaying 4-2
 - messages
 - destination unreachable 13-19
 - displaying on terminal or console 3-8
 - Echo, ICMP 13-19
 - GDP Query 14-70
 - GDP Report 14-70
 - host unreachable 13-17
 - ICMP 13-16, 13-35
 - Internet broadcast 13-12
 - protocol unreachable 13-17
 - redirect, ICMP 13-17
 - short status, frame relay 8-59
 - unreachable 13-17
 - metric holddown command 14-67
 - metric maximum-hops command 14-67
 - metric weights command 14-66
 - metrics
 - adjusting 14-49
 - assigning for redistribution 14-59
 - automatic translations 14-56
 - IGRP 14-8, 14-66
 - setting default 14-58
 - transformation table 14-56
 - translations supported 14-56
 - MIB
 - FDDI support 4-31, 6-33
 - RFCs 4-31
 - source-route bridging support 4-31, 22-3
 - support 4-31
 - Token Ring support 4-31, 6-26, 22-3
 - microwave communications
 - simplex Ethernet 13-39
 - modem
 - configuring line as dedicated SLIP 13-51
 - interactive mode, SLIP 13-52
 - monitor
 - logging messages to 4-39
 - monitoring FDDI 6-34
 - MOP, DEC server 3-11
 - more bit
 - use in X.25 8-30
 - more data bit 8-44
 - More prompt
 - use in multiple screen output 2-8
 - MTU
 - definition 13-18
 - IGRP 14-8
 - path discovery 13-18
 - mtu command 7-12
 - MTU size
 - adjusting media MTU 7-12
 - default media MTU table 7-13
 - specifying, IP 13-18
 - specifying, ISO CLNS 15-12
 - multibus memory
 - testing 5-15
 - multicast
 - SMDS address mapping 8-70
 - multihoming
 - IS-IS areas 15-7
 - multilink SDLC transmission groups 23-24
 - multiple novell helper-address 17-19
 - multiple paths
 - Apollo Domain 9-4
 - multipoint
 - support in SDLC 24-11
 - support in SDLLC 25-8

Multiport Communications Interface card

See MCI card

multiport source-route bridges

configuring 22-13

multiring command 22-7

N

Name Binding Protocol

See NBP

name cache, NetBIOS 22-53

name-connection command 3-3

National Science Foundation Network

See NSFnet

NBP

AppleTalk routing protocol 10-2, 10-6

name registration, displaying 10-70

ping command, help subcommand 10-80

ping command, lookup subcommand 10-81

ping command, parms subcommand 10-80

ping command, poll subcommand 10-81

ping command, zones subcommand 10-82

ping interface, AppleTalk 10-79

NCP

DEC MOP 3-11

DECnet Phase IV parameters 12-3

neighbor

BGP 14-29

displaying ES, IS 15-24, 15-25

displaying table of VINES 19-12

EGP 14-39

ISO CLNS 15-12

neighbor any command (BGP) 14-30

neighbor any command (EGP) 14-44

neighbor any third-party command 14-44

neighbor command 14-30, 14-31, 14-32, 14-33,
14-39, 14-42, 14-48

neighbor interface command 14-23

NET

configuring 16-9

ISO CLNS addresses 15-4

static address for router 15-17

net command 16-5, 16-9

Net/One

See UB Net/One

NetBIOS

access control filtering 22-55

access control using byte offset 22-58

access control using station names 22-55

access filters, using 22-66

assigning station access list name 22-56

error recovery 22-29

name caching

creating static entries 22-52

debugging 22-54

enabling 22-51

maintaining 22-54

netbios access-list bytes deny command 22-59

netbios access-list bytes permit command 22-59

netbios access-list host deny command 22-56

netbios access-list host permit command 22-56

netbios enable-name-cache command 22-53

netbios input-access-filter bytes command 22-60

netbios input-access-filter host command 22-57

NetBIOS name cache 22-53

NetBIOS name caching 22-50

netbios output-access-filter bytes command 22-60

netbios output-access-filter host command 22-58

netbooting

description 2-17

loading system images 2-17

over frame relay 8-61

over X.25 8-34

specifying buffer size 4-9

using nonvolatile memory 2-17, 4-8

network addresses

resolution 13-8

network architecture

ISO CLNS 15-2

network backdoor command 14-36

network command 14-6, 14-16, 14-26, 14-28,
14-29, 14-39

network configuration file

changing name 4-7

copying to a remote host 2-15

Network Control Program

See NCP

Network Entity Title

See NET

Network Information Center

See NIC

network numbers

Apollo Domain 9-3

repairing on Novell IPX 17-3

-
- network protocol
 - AppleTalk 10-1
 - X.25 8-2
 - network server
 - assigning host name 4-1
 - changing host name 4-1
 - priority queuing 7-4
 - Network Service Access Points
 - See NSAP
 - network services
 - tailoring use of 4-37
 - network supporting Phase IV, Phase V
 - designing 12-21
 - network-protocol-to-X.121
 - display address mapping 8-40
 - NFS
 - port number 7-6
 - NIC
 - assigning Internet addresses 13-2
 - IP address assignment 13-2
 - RFC maintenance 13-2
 - no 17-5
 - no ip routing command 21-10
 - no slip command 13-51
 - no snmp-server command 4-32
 - node numbers, area sizes
 - specifying 12-6
 - node type
 - specifying 12-6
 - nonbroadcast networks
 - configuring OSPF 14-23
 - non-SDLC serial tunneling
 - configuring steps 23-5
 - nonvolatile memory
 - checksum protection 2-14
 - clearing contents of 2-14
 - erasing configuration from 5-14
 - executing commands automatically at startup 2-14
 - password checking 4-23
 - use in netbooting 2-17, 4-8
 - writing configuration file to 2-14, 5-14
 - notification
 - pending output 4-48
 - setting 4-48
 - notify command 4-48
 - novell access-group command 17-9
 - novell encapsulation command 17-4
 - novell helper-address command 17-19
 - novell helper-list command 17-18
 - novell input-network-filter command 17-12
 - novell input-sap-filter command 17-15
 - Novell IPX
 - addresses 17-1
 - broadcast helper facilities 17-18
 - Cisco's implementation 17-1
 - configuration example 17-24
 - configuration restrictions 17-2
 - configuring 17-2
 - access lists 17-8
 - for SMDS 8-72
 - corrupted numbers, repairing 17-3
 - debugging the network 17-27
 - displaying cache entries 17-24
 - displaying interface parameters 17-25
 - displaying routing table 17-25
 - displaying servers 17-25
 - displaying traffic 17-26
 - enabling fast switching 17-22
 - enabling on interface 17-3
 - enabling routing 17-2
 - encapsulation 17-4
 - extended access lists 17-9
 - filtering outgoing traffic 17-9
 - filtering packets 17-8
 - filtering routing updates 17-12
 - filtering SAP messages 17-8
 - filtering traffic 17-9
 - flooding of broadcasts on 17-18
 - global configuration command summary 17-28
 - helper address 17-19
 - helper facilities 17-18
 - input filters 17-12
 - interface subcommand summary 17-29
 - maximum paths 17-6
 - monitoring the network 17-24
 - multiple helper address 17-21
 - output filters 17-13
 - ping command 17-27
 - repairing network numbers 17-3
 - restricting SAP updates 17-22
 - routing filters 17-13
 - routing over frame relay 8-62
 - routing protocol 17-1
 - SAP filters 17-13, 17-15, 17-17

- SAP update delays 17-23
- specifying servers 17-19
- standard access lists 17-8
- static routes 17-4
- update timers 17-7
- novell maximum-paths command 17-6
- novell network command 17-3
- novell output-network-filter command 17-13
- novell output-sap-delay command 17-23
- novell output-sap-filter command 17-15
- novell route command 17-4
- novell route-cache command 17-22
- novell router-filter command 17-13
- novell router-sap-filter command 17-15
- novell routing command 17-3
- novell sap-interval command 17-22
- novell source-network-update command 17-3
- novell update-time command 17-7
- NSAP
 - addressing rules 15-6
 - field formats 15-5
 - GOSIP format 15-7
 - ISO CLNS addresses 15-4
 - shortcut command 15-16
 - static address assignments 15-17
 - static interface addresses 15-17
- NSFnet
 - use of EGP 14-2, 14-38
- null interface
 - configuring 6-82

O

- offset-list command 14-49
- Open Shortest Path First
 - See OSPF
- Open Systems Interconnection
 - See OSI
- operating system
 - reloading 2-18
- optical bypass switch
 - bypass mode 6-3
- optional password verification 4-30
- order of banner displays 4-4
- Organization Unique Identifier
 - See OUI
- OSI

- bridging 21-1
- reference model, AppleTalk 10-3
- OSPF
 - adjacency, defined 14-13
 - advertised addresses, consolidating 14-19
 - advertised Hello interval, setting 14-21
 - area authentication, setting 14-18
 - area border router, defined 14-10
 - area parameters, configuring 14-18
 - AS boundary router, defined 14-11
 - assigning area ids 14-16
 - authentication key, specifying 14-22
 - backbone router, defined 14-10
 - backbones 14-10
 - Cisco implementation summary 14-14
 - configuration steps summarized 14-15
 - database, displaying 14-92
 - default AS boundary router routes 14-61
 - definition 14-2
 - description 14-9
 - designated routers, defined 14-14
 - enabling routing 14-15
 - example, assigning area ids 14-16
 - external routing 14-13
 - Hello protocol 14-13
 - interarea routing 14-12
 - interface parameters, displaying 14-98
 - interface-specific parameters, configuring 14-19
 - internal router, defined 14-10
 - intra-area routing 14-12
 - IP subnetting support 14-12
 - link state updates, setting transmission time 14-21
 - link state, setting retransmission interval 14-20
 - neighbor information, displaying 14-99
 - neighbor routers 14-13
 - nonbroadcast networks, configuring for 14-23
 - path cost, specifying 14-20
 - physical network support 14-11
 - redistributing routes into OSPF 14-59
 - redistributing routes into other domains 14-62
 - route redistribution 14-78
 - router classifications 14-10
 - router dead interval, setting 14-22
 - router priority, setting 14-21
 - routing conventions 14-11
 - routing processes, displaying 14-90

- routing protocol
 - areas 14-9
 - domain 14-9
 - overview 14-8
- routing table, displaying 14-99
- stub areas, defined 14-13
- support in router 1-2
- virtual links
 - creating 14-24
 - defined 14-14
- OUI
 - compatibility between Cisco routers and IBM 8209 bridges 22-42
- out-routing filter
 - configuring 12-18
- P**
- packet acknowledgment
 - X.25 8-31
- packet filtering
 - establishing size 4-34
- packet hold queue
 - defining X.25 8-32
- packet internet groper
 - See ping
- packet size
 - maximum IP packet size 13-18
 - maximum ISO CLNS packet size 15-12
 - setting maximum 7-12
 - setting, adjusting 13-18
 - X.25, specifying output 8-30
- packet switch nodes
 - See PSN
- packet-level restarts
 - forcing X.25 8-33
- packets
 - administrative filtering 21-20
 - controlling size of SLIP 13-53
 - explorer, configuring 22-12
 - filtering Ethernet-encapsulated 21-20, 21-21
 - filtering IEEE 802.3-encapsulated 21-21, 21-22
 - filtering IEEE 802.5-encapsulated 22-47
 - filtering Novell IPX 17-8
 - filtering SNAP-encapsulated 21-20, 21-21, 22-47
 - ISO CLNS 15-15, 15-28
 - network carrier, X.25 8-33
- padding
 - character setting 4-49
- padding command 4-49
- parallel router 14-53
- PARC Universal Protocol
 - See PUP
- passive-interface command 14-45
- password
 - assigning 4-23
 - configuring IS-IS 16-12
 - for privileged level access 2-8
 - line, establishing 4-43
 - optional 4-30
 - privileged-level 4-22
 - recovering from lost 4-23
 - specifying 4-23
- password command 4-23, 4-43
- password encryption 4-24
- passwords
 - establishing 4-18
- path
 - attributes, BGP 14-37
 - costs, assigning 21-15
 - discovery, MTU 13-18
 - selection, configuring for DECnet Phase IV 12-9
- pattern matching
 - X.25 regular expression D-1
- PC/3270 emulation
 - and source-route bridging 22-68
- PCM
 - FDDI 6-42
- PDU
 - error, See ERPDU
 - ISO CLNS 15-14
 - redirect, See RDPDU
- peer
 - definition 22-15, 22-19
 - See also remote peer
- peer bridges
 - listing 22-16, 22-20
- pending output
 - terminal notification of 4-48
- permissions
 - access list 4-32, 13-24
- Phase I AppleTalk
 - See AppleTalk nonextended

-
- Phase II AppleTalk
 - See AppleTalk extended
 - Phase IV, Phase V
 - DECnet, designing network to support 12-21
 - Physical Connection Management
 - See PCM
 - ping
 - aborting session 5-13
 - definition of 5-13
 - function 13-19
 - IP interface 13-70
 - ISO CLNS 15-27
 - Novell IPX 17-27
 - PUP 18-3
 - specifying Internet header options 13-20
 - test characters table 15-27
 - testing connectivity 5-13
 - use on AppleTalk 10-78
 - VINES 19-13
 - X.25 support over multiple serial lines 8-10
 - ping command 13-19, 13-70, 15-27, 17-27, 19-13
 - point-to-point protocol
 - See PPP
 - point-to-point updates
 - filtering 14-48
 - Poor Man's Routing
 - on DECnet 12-27
 - port
 - prioritizing 7-6
 - PPP
 - configuring 6-78
 - ppp authentication chap command 6-79, 6-89
 - preferred routes
 - specifying 16-8
 - priority list
 - definition 7-5
 - priority queuing
 - assigning to an interface 7-9
 - by serial link address 7-9
 - priority queuing
 - assigning default 7-8
 - assigning to a protocol 7-5
 - assigning to an interface 7-7
 - by interface type 7-5
 - definition 7-4
 - group 7-9
 - maximum packets 7-8
 - monitoring 7-9
 - types of 7-4
 - priority-group command 7-9, 23-11
 - priority-list command 7-5, 7-7, 7-8, 23-10
 - with SNA local LU address 22-35, 23-12, 23-22
 - privileged mode last resort login
 - enabling 4-28
 - privileged-level commands
 - definition 2-8
 - TACACS 4-22, 4-27
 - Probe
 - Hewlett-Packard proxy support 13-21
 - processes
 - See system processes
 - processor
 - configuring auxiliary port 4-42
 - processor configuration register 2-17, 4-8
 - protocol analyzer
 - connecting a 4-42
 - Protocol Data Unit
 - See PDU
 - protocol traffic
 - displaying statistics 13-64
 - protocols
 - Apollo Domain 9-1
 - BGP, configuring 14-28
 - counted per packet 6-7
 - displaying configured 5-8
 - EGP, configuring 14-38
 - ES-IS 15-3
 - exterior IP routing 14-4
 - Finger 4-37
 - GDP, configuring 14-70
 - Hello, configuring 14-27
 - IGRP, configuring 14-5
 - interior IP routing 14-4
 - IS-IS 16-8
 - multiple routing 14-3
 - OSPF 14-8
 - PPP 6-78
 - RIP, configuring 14-25
 - spanning tree, defining 21-8
 - specific configuring 8-71
 - STUN 23-6, 23-39
 - supported by X.25 8-6
 - XNS 20-1

protocol-to-virtual-circuit
 mapping 8-13
 protocol-to-X.121
 address mapping 8-40
 proxy
 ARP, address resolution 13-10
 assigning number to AppleTalk 10-19
 explorers 22-26
 polling for STUN 23-37, 23-38
 Probe, Hewlett-Packard support 13-21
 proxy explorer with NetBIOS 22-54
 PSN
 logging transactions 8-42
 pulse-time command 7-2
 PUP
 addresses 18-1
 configuring routing 18-1
 debugging the network 18-4
 definition 18-1
 displaying ARP entries 18-3
 displaying routing entries 18-3
 displaying statistics 18-3
 displaying traffic 18-3
 enabling routing 18-2
 helper address 18-3
 monitoring the network 18-3
 ping command 18-3
 services 18-3
 pup address command 18-2
 pup helper-address command 18-3
 pup map command 18-2
 pup routing command 18-1
 PVC
 serial interfaces 8-21
 TCP connection 8-22
 X.25 Network 8-13
 PVCs
 configuring switched 8-21
 highest incoming circuit number 8-25
 highest outgoing circuit number 8-26
 highest two-way circuit number 8-26
 lowest incoming circuit number 8-25
 lowest outgoing circuit number 8-26
 lowest two-way circuit number 8-26
 setting 8-12

Q

query message
 GDP 14-70
 question mark (?) command 2-9
 queue
 controlling hold 7-10
 queue length
 SNMP server 4-33
 queue-limit command 7-8
 queuing, priority
 See priority queuing
 quit command 3-3

R

range limit keywords for SVCs 8-25
 RARP
 defined 13-8
 RDPDU
 configuring for sending, ISO CLNS 15-15
 disabling address mask 15-15
 interval to disable 15-15
 redirect information
 displaying ISO CLNS 15-22
 redirect messages
 generating 13-17
 Redirect Protocol Data Unit
 See RDPDU
 redistribute command 14-57, 14-59, 16-6, 16-11
 redistribute ospf command 14-62
 redistribute static command 16-7
 redistribution
 assigning metrics for 14-59
 BGP 14-56
 EGP 14-56
 Hello 14-56, 14-75
 IGRP 14-56, 14-75
 IS-IS 16-10
 ISO-IGRP 16-6
 OSPF, routes from 14-62
 OSPF, routes into 14-59
 RIP 14-75
 routing information 14-55
 routing table
 sample entries 16-3
 static routing 14-74

-
- redundant links
 - configuring 23-32
 - reload command 2-18
 - remote console function
 - MOP 3-11
 - remote CSU/DSU
 - loopback 7-15
 - remote monitoring
 - using an analyzer 4-42
 - remote peer
 - enabling COS 22-17
 - enabling LLC2 Local Acknowledgment with 22-30
 - specifying forced RSRB protocol version number 22-16, 22-20, 22-23
 - specifying frame size 22-16, 22-20, 22-23
 - remote source-route bridging
 - combining transport methods 22-24
 - configuring 22-15
 - configuring with TCP 22-15
 - enabling class of service 22-17
 - example configuration 22-85
 - example configuration with Local Acknowledgment 22-87
 - largest frame 22-26
 - listing peer bridges 22-16
 - proxy explorers 22-26
 - simple reliability 22-85
 - size of backup queue 22-19
 - with direct encapsulation 22-22
 - with FST encapsulation 22-19
 - report message
 - GDP 14-70
 - reset request
 - retransmission timer 8-29
 - restart
 - packet-level 8-33
 - retransmission timer request 8-28
 - resume command 3-3
 - retransmission timer
 - call request 8-28
 - clear request 8-29
 - keywords and defaults 8-28
 - reset request 8-29
 - restart request 8-28
 - setting for LAPB 8-3
 - X.25 Level 3 8-27
 - retransmit count
 - TACACS 4-26
 - retries
 - controlling 4-26
 - reverse address resolution
 - using BootP 13-11
 - using RARP 13-11
 - Reverse Address Resolution Protocol
 - See RARP
 - reverse ARP
 - See RARP
 - reverse charge calls
 - accepting X.25 8-32
 - configuring X.25 8-10
 - RIF
 - clearing the cache 22-90
 - configuring explorer packets 22-12
 - configuring static entry 22-9
 - displaying the cache 22-90
 - enabling use 22-7
 - establishing ring groups 22-9
 - for routed protocols 22-7
 - time-out interval, determining 22-8
 - use in source-route bridging 22-2, 22-7
 - rif command 22-9
 - rif timeout command 22-8
 - ring
 - scheduling FDDI 6-41
 - ring group
 - configuring 22-13
 - definition 22-13
 - example 22-14
 - ring speed
 - configuring for IGS/TR 6-27
 - ring table 22-95
 - ring-speed command 6-27
 - RIP
 - configuring 14-25
 - creating the routing process 14-26
 - definition 14-2
 - displaying list of networks 14-26
 - hop count 14-26
 - metric transformations 14-56
 - redistribution example 14-75
 - support in router 1-2
 - root bridge
 - selecting 21-13

- rotary groups
 - See dialer rotary groups
- route cache
 - displaying 13-61
- router
 - AppleTalk seed 10-7
 - assigning host name 4-1
 - capabilities 1-1
 - changing host name 4-1
 - communicating through X.25 network 8-8
 - designated for DECnet Phase IV 12-12
 - multiple protocol support 1-1
 - parallel 14-53
 - priority queuing 7-4
 - running SLIP on 13-46
 - SLIP connections 13-48
- router bgp command 14-28
- router chaos command 11-2
- router egp 0 command 14-43
- router egp command 14-39
- router hello command 14-28
- router igrp command 14-6
- router isis command 16-9
- router iso-igrp command 16-4
- router ospf command 14-16
- router rip command 14-26
- routes
 - clearing dynamic IP 14-83
 - clearing IP 13-58
 - direct connect 14-52
 - generating default 14-54
 - IGRP 14-5
 - ISO CLNS 15-29, 16-20
 - overriding static 14-54
 - removing static 14-83
- routing
 - configuring VINES 19-1
 - DECnet Phase IV 12-4
 - definition of 14-1
 - filtering information 14-45
 - IS-IS 16-8
 - ISO CLNS 15-3, 15-8, 16-2
 - ISO-IGRP 16-4
 - loops 14-55
 - on subnets 13-5
 - PUP 18-1
 - special configuration techniques 14-63
 - XNS 20-3
 - See also protocols
- routing cache
 - displaying ISO CLNS 15-20
- routing information
 - filtering sources of 14-50
 - passing among different protocols 14-57
 - redistributing 14-55
- Routing Information Field
 - See RIF
- routing multiple protocols 22-85
- routing processes, starting
 - BGP 14-29
 - BGP (back door network) 14-36
 - EGP 14-40
 - Hello 14-28
 - IGRP 14-6
 - OSPF 14-16
 - RIP 14-27
- routing protocols
 - BGP 1-2, 14-2, 14-28
 - CLNP 15-1
 - CLNS 15-2
 - concurrent 14-2
 - configuration overview 14-4
 - displaying parameters and status 14-89
 - EGP 1-2, 14-2, 14-38
 - ES-IS 15-1
 - exterior 14-2, 14-4
 - Hello 14-2, 14-27
 - IGRP 1-2, 14-2, 14-5
 - interior 14-2, 14-4
 - IS-IS 15-2, 16-8
 - ISO CLNS 15-1
 - ISO-IGRP 1-2, 16-4
 - metric translations 14-56
 - multiple 14-3
 - OSPF 1-2, 14-2
 - overriding static routes 14-54
 - RIP 1-2, 14-2, 14-25
 - support for native 1-3
 - Ungermann-Bass Net/One 20-9
- routing table
 - Apollo Domain 9-7
 - AppleTalk 10-73
 - ATG 12-24
 - BGP 14-35, 14-84

- CHAOSnet 11-2
- DECnet Phase IV 12-3, 12-10
- default network in IP 14-55
- displaying 12-28, 13-63, 14-99
- dynamic IP 14-2, 14-54
- dynamic ISO CLNS 15-3
- interface routes in IP 13-7
- IP 13-63, 14-51, 14-99, 14-102
- ISO CLNS 15-31
- log of events on VINES 19-14
- main IP 14-35
- matching entries in X.25 D-1
- Novell IPX 17-12, 17-25
- PUP and IP 18-2
- removing entries from IP 13-58, 14-8, 14-83
- static IP 14-2, 14-54, 14-63
- static ISO CLNS 15-3
- VINES 19-3, 19-10, 19-12, 19-14
- X.25 8-19
- XNS 20-18, 20-23
- Routing Table Maintenance Protocol
 - See RTMP
- Routing Table Protocol
 - See RTP
- routing updates
 - controlling list of networks 20-18
 - filtering Novell IPX 17-12
 - filtering XNS 20-17
- routing weights 14-31
- RPC
 - port number 7-6
- RS-232 auxiliary port
 - configuring 4-42
- RTMP
 - AppleTalk routing protocol 10-2, 10-8
- RTP
 - VINES 19-3

S

- SAP
 - access lists for filtering 17-14
 - building filters 17-13
 - configuring filters, Novell IPX 17-15
 - filtering messages on Novell IPX 17-8
 - input filter example 17-15
 - Novell IPX 17-1
 - output filter example 17-17
 - restricting updates 17-22
 - static configuring 17-5
 - update delays 17-23
 - use in bridging 21-2
- scheduler-interval command 7-3
- SCI card
 - loopback on serial 7-17, 7-18
 - serial interface 6-8
- screen
 - setting length 4-48
- SDLC
 - affecting output buffering 24-14
 - choosing the transport protocol 23-7
 - Cisco's implementation 24-11
 - comparison with LLC2 24-2
 - configuring 24-11
 - configuring local acknowledgment 23-17
 - controlling frame size 24-14
 - controlling the protocol 24-13
 - controlling window size 24-14
 - debugging 24-19
 - frame format 23-39
 - local acknowledgment 23-2, 23-15
 - local address priority configuration 23-22
 - monitoring 24-17
 - multiple-link transmission groups 23-24
 - multipoint support 24-11
 - normal response mode 24-2
 - overview 24-2
 - parameters, tuning for network performance
 - 25-3
 - polling secondary stations 24-15
 - retry counts, controlling 24-13
 - station addressing 24-2
 - STUN support 23-2
 - T1 timer 24-14
 - table of parameters 24-13
 - timers, controlling 24-13
 - using STUN 23-35
- sdlc address command 24-11
- sdlc command 24-12
- SDLC command summary 24-21
- sdlc fair-poll-timer command 24-15, 24-16
- sdlc frmr-disable command 24-13
- sdlc holdq command 24-15
- sdlc k command 24-14

- sdhc n1 command 24-14
- sdhc n2 command 24-14
- sdhc poll-limit-value command 24-15, 24-16
- sdhc poll-pause-timer command 24-15, 24-16
- sdhc t1 command 24-13
- SDLC transport function
 - configuring steps 23-4
- SDLLC
 - configuring 25-3
 - debugging 25-21
 - Ethernet support 25-9
 - function 25-1
 - global command summary 25-21
 - interface subcommand summary 25-21
 - monitoring 25-18
 - multipoint support 25-8
 - on serial interfaces, configuring 25-7
 - optional serial interface commands 25-10
 - specifying optional serial interface commands 25-10
- sdllc ring-largest-frame command 25-10
- sdllc sdhc-largest-frame command 25-10
- sdllc traddr command 25-7
- sdllc xid command 25-12
- SDSU
 - SMDS 8-66
- secondary address
 - subnetting 13-6
 - use in frame relay and SMDS 14-65
 - use in networking subnets 13-43
 - using 14-53
- security
 - accepting unlabeled datagrams 13-31
 - access lists 13-24
 - classification range 13-30
 - dedicated 13-30
 - establishing 4-22
 - ICMP error messages 13-35
 - modifying levels 13-31
 - multilevel 13-30
 - password encryption 4-24
 - setting classifications 13-30
- security option
 - adding by default 13-32
 - extended 13-32
 - prioritizing 13-33
 - removing by default 13-32
- security precautions
 - with Flash Memory card 4-10
- seed router
 - AppleTalk 10-7
- separated subnets
 - creating network from 13-43
- Sequence Packets Protocol
 - See SPP
- Serial
 - dte-invert-txc command 6-9
- serial encapsulation
 - See direct encapsulation
- serial interface
 - adjusting characteristics 7-1
 - backup, see dial backup
 - clearing 6-10
 - clock rate 7-2
 - configuring 6-8, 6-47
 - configuring IP 13-42
 - configuring STUN 23-7
 - DCE appliques 7-2
 - debugging 6-14, 6-51
 - DTR signal pulsing 7-2
 - encapsulation methods 6-9
 - HSSI 6-46
 - IP processing on 13-38
 - LAT compression 21-35
 - leased serial line, using LAPB 8-1
 - load balancing 21-30
 - loopback on 7-14, 7-18
 - loopback test on 7-17
 - maintaining 6-10
 - monitoring 6-11
 - parallel 21-30
 - show interfaces field descriptions 6-12
 - specifying 6-8
 - support 6-8
 - transmit delay 7-1
 - unnumbered IP 13-38
- serial interface cards
 - loopback on 7-18
- serial interface commands
 - specifying 25-10
- Serial Line Internet Protocol
 - See SLIP
- Serial Tunnel
 - See STUN

Serial-port Communications Interface card
 See SCI card

Service Access Points
 and IEEE 802 standard 21-2

Service Advertisement Protocol
 See SAP

service command 4-37

service config command 2-16

service config memory command 2-15

service password-encryption command 4-24

service timestamps command 5-11, 5-16

service, dial backup
 See dial backup

services
 network, tailoring use of 4-37

sessions
 displaying active 3-6
 exiting 3-4

setup command 2-1

setup command facility
 capabilities 2-2
 configuring protocols 2-2
 prompts displayed by 2-4
 using 2-2

short status messages
 frame relay, requesting 8-59

shortcut command
 NSAP 15-16

show ? command 5-1

show access-lists command 13-24

show apollo arp command 9-8

show apollo interface command 9-7

show apollo route command 9-7

show apollo traffic command 9-8

show apple adjacent-routes command 10-62

show apple arp command 10-63

show apple cache command 10-63

show apple interface command 10-65

show apple neighbor command 10-71

show appletalk access-lists command 10-62

show appletalk global command 10-64

show appletalk macip-clients command 10-69

show appletalk macip-servers command 10-66

show appletalk name-cache command 10-70

show appletalk nbp command 10-70

show appletalk route command 10-73

show appletalk socket command 10-75

show appletalk traffic command 10-69, 10-76

show appletalk zone command 10-78

show arp command 13-8, 13-59

show async-bootp command 13-69

show bridge command 21-39

show buffers command 4-5, 5-2

show chaos-arp command 11-2

show clns cache command 15-20

show clns command 15-19

show clns es-neighbors command 15-26

show clns interface command 15-23

show clns is-neighbors command 15-25

show clns neighbors command 15-24

show clns protocol command 16-22

show clns redirects command 15-22

show clns route command 16-20

show clns traffic command 15-21

show cmns command 8-51

show command 5-1

show configuration command 2-14, 5-7

show controller mci command 6-8

show controller serial command 6-8

show controllers command 6-5

show controllers field descriptions 6-6

show controllers token command 22-93

show debugging command 5-11

show decnet interface command 12-27

show decnet map command 12-28

show decnet route command 12-28

show decnet traffic command 12-29

show dialer 6-76

show dialer interface command 6-76

show environment command 5-8

show environment last command 5-10

show flash all command 4-12, 5-10

show flash command 4-12, 5-10

show frame-relay map command 8-64

show frame-relay traffic command 8-64

show hosts command 13-60

show imp-hosts command 8-41

show interface bri 0 command 6-16

show interface serial command 25-19

show interfaces accounting command 6-7

show interfaces command 8-4, 8-42, 8-63,
 22-93, 24-17

show ip accounting command 13-59

show ip aliases command 13-67

show ip arp command 13-67
 show ip bgp command 14-84
 show ip bgp neighbors command 14-85, 14-87
 show ip bgp paths command 14-87
 show ip bgp summary command 14-88
 show ip cache command 13-61
 show ip egp command 14-88
 show ip interface command 13-62
 show ip irdp command 14-74
 show ip ospf command 14-90, 14-92
 show ip ospf interface command 14-98
 show ip ospf neighbor command 14-99
 show ip protocols command 14-89
 show ip route command 11-2, 13-63, 14-55, 14-99
 show ip tcp header-compression command 13-65
 show ip traffic command 11-3, 13-64
 show isis database command 16-24
 show isis routes command 16-22
 show line command 13-68
 show llc2 command 24-8, 25-18
 show lnm bridge command 22-78
 show lnm config command 22-78
 show lnm interface command 22-79
 show lnm ring command 22-81
 show lnm station command 22-81
 show local-ack command 22-32, 25-19
 show logging command 4-38, 4-41, 5-7
 show memory command 5-3
 show novell cache command 17-24
 show novell interface command 17-7, 17-25
 show novell route command 17-25
 show novell servers command 17-25
 show novell traffic command 17-26
 show processes command 5-4
 show processes memory command 5-5
 show protocol command 5-8
 show pup arp command 18-3
 show pup router command 18-3
 show pup traffic command 18-3
 show rif command 22-8, 22-90
 show sessions command 3-6
 show slip command 13-68
 show smds addresses command 8-75
 show smds map command 8-75
 show smds traffic command 8-76
 show source-bridge command 22-91
 show span command 21-40
 show stacks command 5-6
 show stun command 23-19, 23-41
 show stun sdlc command 23-42
 show tcp command 3-6
 show terminal command 3-7, 3-9
 show users all command 4-42
 show users command 3-7
 show version command 4-21, 6-4
 show vines host command 19-11
 show vines interface command 19-11
 show vines neighbors command 19-12
 show vines route command 19-12
 show vines traffic command 19-13
 show x25 map command 8-8, 8-40
 show x25 remote-reds command 8-39
 show x25 route command 8-19
 show x25 vc command 8-42
 show xns cache command 20-22
 show xns interface command 20-22
 show xns route command 20-6, 20-23
 show xns traffic command 20-23
 shutdown
 interface 6-3
 SNMP system 4-36
 shutdown command 6-3
 signal bits
 FDDI 6-42
 signaling phase
 FDDI CMT 6-42
 signals
 pulsing DTR 7-2
 Simple Network Management Protocol
 See SNMP
 simplex circuit
 definition of 13-39
 simplex Ethernet interfaces
 configuring IP 13-39
 single DDR telephone number
 specifying 6-65
 SLIP
 access lists 13-54
 address, specifying 13-51
 bootp parameters, displaying 13-69
 configured lines, displaying status 13-68
 configuring 13-50, 13-57
 connections to router 13-48
 controlling packet size 13-53

-
- debugging 13-76
 - description 13-46, 13-47
 - disabling 13-58
 - on auxiliary port 13-51
 - displaying IP ARP cache 13-67
 - displaying line status 13-68
 - displaying mapped Internet address 13-67
 - dynamic address assignment 13-49, 13-52
 - extended BootP requests 13-55
 - extended bootp requests, specifying 13-55
 - hold queue 13-53
 - IP address assignment 13-51
 - maintaining 13-58
 - modem line 13-51
 - modem line in interactive mode 13-52
 - monitoring 13-67
 - MTU size of internet packets 13-53
 - on dedicated line 13-49
 - on permanent line 13-49
 - setting baud rate 13-53
 - treatment of broadcasts 13-48
 - using dedicated line 13-49
 - using default address 13-50, 13-52
 - using dynamic addresses 13-49
 - using permanent address 13-49
 - slip access-class command 13-54
 - slip address command 13-51
 - slip address dynamic command 13-52
 - slip address subcommand 13-51
 - slip command 13-48
 - slip dedicated command 13-51
 - slip default command 13-48
 - slip hold-queue command 13-53
 - slip interactive command 13-52
 - slip mtu command 13-53
 - SMDS
 - arp command 8-72
 - ARP, enabling 8-69
 - assigning addresses 8-67
 - AT&T version 8-71
 - configuration examples 8-74
 - configuring 8-65, 8-66
 - counters, displaying 8-76
 - debugging 8-76
 - disabling split horizon 14-64
 - displaying addresses 8-75
 - displaying counters 8-76
 - DS1 8-66
 - enabling 8-68
 - encapsulation 8-68
 - hardware requirements 8-66
 - interface subcommand summary 8-81
 - IP fast switching 8-73
 - IP split horizon, default 8-66
 - mapped addresses, displaying 8-75
 - monitoring 8-75
 - multiprotocol configuration 8-74
 - protocols supported 8-72
 - protocol-specific configuration 8-71
 - remote peer configuration 8-74
 - SDSU 8-66
 - special SMDS 8-66
 - specifying address 8-68
 - static map 8-69
 - subcommand summary 8-81
 - using addresses 8-67
 - smds address command 8-68
 - smds d15-mode command 8-71
 - smds enable-arp command 8-69
 - smds multicast command 8-70
 - smds statio-map command 8-69
 - SMT message queue size
 - setting 6-46
 - SMTP
 - port number 7-6
 - smt-queue-threshold command 6-46
 - SNA
 - and SDLLC 25-1
 - error recovery 22-29
 - local LU address prioritization 22-33
 - stations, translating between Token Ring and serial 24-15
 - STUN support 23-2
 - SNA local LU address priorities 22-34
 - SNAP
 - filtering encapsulated packets 21-20, 21-21, 22-47
 - frames on output, filtering 22-48
 - SNMP
 - configuring 4-31
 - configuring for AppleTalk routing 10-43
 - port number 7-6
 - system shutdown 4-36

-
- SNMP server
 - community string 4-32
 - configuring 4-32
 - defining access list 4-32
 - message queue length 4-33
 - packet filtering 4-34
 - setting community access 4-32
 - TRAP messages 4-34
 - snmp-server access-list command 4-32
 - snmp-server community command 4-32
 - snmp-server host command 4-34
 - snmp-server packetsize command 4-34
 - snmp-server queue length command 4-33
 - snmp-server system-shutdown command 4-36
 - snmp-server trap-authentication command 4-35
 - snmp-server trap-timeout command 4-35
 - SNPA
 - masks 15-15
 - software
 - displaying version level 2-3
 - loading over the network 2-17
 - source addresses
 - administrative filtering 21-23, 22-49
 - source bridge fst-peername command 22-20
 - source-bridge command 22-11
 - source-bridge cos-enable command 22-17, 23-10
 - source-bridge enable-80d5 command 22-43
 - source-bridge input-address-list command 22-50
 - source-bridge input-lsap-list command 22-47
 - source-bridge input-type-list command 22-47
 - source-bridge largest-frame command 22-26
 - source-bridge max-hops command 22-13
 - source-bridge old-sna command 22-68
 - source-bridge output-address-list command 22-50
 - source-bridge output-lsap-list command 22-47
 - source-bridge output-type-list command 22-48
 - source-bridge passthrough command 22-31
 - source-bridge proxy-explorer command 22-26
 - source-bridge proxy-netbios-only command 22-54
 - source-bridge remote-peer command with direct encapsulation 22-23
 - source-bridge remote-peer command with fst encapsulation 22-20
 - source-bridge remote-peer command with tcp encapsulation 22-16
 - source-bridge ring-group command 22-13
 - source-bridge route-cache command 22-5
 - source-bridge sap-80d5 command 22-44
 - source-bridge spanning command 22-12
 - source-bridge tcp-queue-max command 22-19
 - source-bridge transparent command 22-38
 - source-route autonomous-switching cache
 - enabling 22-4
 - source-route bridge groups
 - bridging to transparent bridge groups 22-36
 - source-route bridging
 - administrative filtering 22-45
 - and SNA 22-1, 22-42
 - assigning a RIF 22-10
 - Cisco's implementation 22-2
 - configuration examples 22-83
 - configuration steps 22-3
 - configuring 22-3
 - configuring explorer packets 22-12
 - configuring IP 13-38
 - debugging 22-93
 - definition 22-1, 22-2
 - determining the RIF 22-5
 - displaying current configuration 22-91
 - displaying RIF cache 22-90
 - dual port configuration 22-12
 - frame conversion between source-route and transparent bridging 22-44
 - global configuration command summary 22-96
 - IBM PC/3270 emulation 22-68
 - interface subcommand summary 22-100
 - interoperability 22-68
 - largest frame 22-26
 - limiting hops 22-13
 - LLC2 Local Acknowledgment function 22-27
 - local, See local source-route bridging
 - maintaining 22-90
 - monitoring 22-90
 - NetBIOS access control 22-55
 - NetBIOS protocol 22-1
 - overview 22-1
 - remote, See remote source-route bridging
 - RIF 22-7
 - RIF timeout interval 22-8
 - source-route fast-switching cache
 - disabling 22-4
 - enabling 22-4
 - Source-Route Translational Bridging
 - See SR/TLB

-
- source-route transparent bridging
 - See SRT
 - spanning explorer packets
 - enabling 22-12
 - spanning tree
 - algorithm 21-4
 - assigning interface to a group 21-10
 - assigning path costs 21-15
 - bridge 21-4
 - bridging and routing IP 21-10
 - broadcast flooding 13-15
 - interface priority, setting 21-16
 - known topology, displaying 21-40
 - load balancing 21-30
 - multiple domains, establishing 21-9
 - parameters, adjusting 21-13
 - parameters, adjusting Hello BPDU interval 21-13
 - parameters, defining forward delay interval 21-14
 - parameters, defining maximum idle intervals 21-14
 - parameters, electing the root bridge 21-13
 - protocols, defining 21-8
 - special-character-bits command 4-44, 4-59
 - speed command 13-53
 - speeds
 - clock rate 7-3
 - split horizon
 - enabling and disabling for IP 14-64
 - SPP
 - VINES 19-8
 - squelch command 6-20
 - SR/TLB
 - 0x80d5 processing 22-43
 - compatibility with IBM 8209 bridges 22-42
 - configuring 22-38
 - description 22-36
 - frame conversions, illustration of A-1
 - in IBM LLC2 environments 22-43
 - mixing IBM 8209 bridges and Cisco routers 22-42
 - overview 22-37
 - routers, in the same network with IBM 8209 bridges 22-42
 - using in IBM LLC2 environments 22-43
 - when to use 22-3, 22-40
 - SRT
 - compared with SR/TLB 21-6
 - configuration overview 21-5
 - frame conversions, illustration of A-1
 - hardware supporting 21-5
 - stack utilization
 - displaying 5-6
 - standard access lists
 - configuring IP 13-24
 - DECnet Phase IV 12-13
 - Novell IPX 17-8
 - XNS 20-15
 - standards
 - obtaining technical F-4
 - startup sequence
 - See setup command facility
 - static map, defining
 - SMDS 8-69
 - static name-to-address mappings 13-20
 - static RIF entry
 - configuring 22-9
 - static routes
 - Apollo Domain 9-3
 - configuring IP 14-63
 - Novell IPX 17-4
 - overriding with dynamic protocols 14-54
 - redistributing ISO CLNS 16-7
 - redistribution 14-74
 - static routing, ISO CLNS
 - interdomain 16-15
 - intradomain 16-13
 - station names
 - use in NetBIOS access control 22-55
 - statistics 4-11
 - statistics
 - accounting 13-36
 - buffer pool 5-2
 - displaying for IP interface 13-62
 - displaying for network interfaces 6-7
 - displaying host 13-60
 - displaying protocol traffic 13-64
 - system memory 5-3
 - storing system software
 - using the Flash Memory card 4-9
 - stub areas
 - creating 14-18
 - OSPF support 14-13

STUN

- choosing the basic protocol 23-7
 - configuration examples 23-25
 - configuration overview 23-4
 - configuring non-SDLC 23-5
 - configuring notes, tips 23-5
 - configuring on the interface 23-7
 - debugging 23-43
 - defining protocols 23-6, 23-39
 - defining your own protocols 23-39
 - description 23-1
 - displaying current status 23-41
 - displaying proxy states 23-42
 - enabling 23-6
 - enabling COS 23-10
 - enabling proxy polling 23-37
 - encapsulation 23-7
 - forwarding frames 23-8
 - global configuration command summary 23-43
 - interface in a group 23-8
 - interface subcommand summary 23-45
 - monitoring 23-41
 - primary side pass-through interval 23-38
 - proxy poll interval 23-38
 - proxy polling 23-37
 - serial link address priorities 23-10
 - traffic prioritization 23-31
 - use in IBM networks 23-25
 - use in SDLC environments 23-35
- stun group command 23-8
- STUN interface
- configuring SDLC Local Acknowledgment 23-17
- STUN peer
- enabling SDLC Local Acknowledgment with 23-17
- stun peer-name command 23-6
- stun poll-interval command 23-38
- stun primary-pass-through command 23-38
- STUN priority port numbers 23-13
- stun protocol-group command 23-6
- stun proxy-poll address discovery command 23-37
- stun proxy-poll address modulus command 23-37
- stun route address command 23-9, 23-18
- stun route all interface serial command 23-9
- stun route all tcp command 23-8
- stun schema format command 23-40
- stun schema length command 23-40

stun schema offset command 23-40

subnet

- default routes 14-55
- networking from separate 13-43
- routing on 13-5
- using address zero 13-7

subnet masks 13-6

- definition of 13-6
- table of 13-6
- using ICMP 13-7

subnet zero 13-7

subnetting

- definition of 13-5
- routing 13-5

SVC

- clearing 8-26, 8-42
- displaying active 8-43
- displaying CMNS information 8-53
- range limits 8-25
- X.25 8-26

Switched Multimegabit Data Services

See SMDS

switched PVCs

- configuring 8-21

switched virtual circuit

See SVC

switching

- configuring X.25 8-17
 - decisions by BGP routing table 14-35
 - fast packet 13-40
 - high-speed IP cache 13-40
 - ISO CLNS header options and packet 15-16
 - ISO CLNS packets 15-12, 15-20
 - PVC on X.25 8-21
 - X.25 local 8-6
 - X.25 remote 8-6
- switching operations
- changing priorities 7-3
 - system process scheduler 7-3
- sx25 ltc command 8-25
- synchronization command 14-37
- ## Synchronous Data Link Control
- See SDLC
- syslog daemon 4-41
- syslog messages
- levels of 4-40

-
- syslog server
 - limiting messages to 4-40
 - syslog server, UNIX
 - logging messages to 4-40
 - sysstat command 4-47
 - system
 - autonomous 14-2
 - monitoring processes 5-1
 - testing 5-15
 - writing configuration information 5-14
 - system banner message
 - See banner message
 - system buffers
 - changing size of 4-7
 - setting 4-5
 - See also buffers
 - system configuration
 - copying to memory 5-14
 - displaying 5-7
 - displaying interface information 6-4
 - erasing information 5-14
 - writing information 5-14
 - writing to a host 5-14
 - writing to nonvolatile memory 5-14
 - system configuration dialog
 - first-time system startup 2-4
 - system error messages
 - See error messages
 - system escape character
 - setting 4-46
 - system file names
 - changing 4-7
 - system host name
 - assigning 2-8
 - system images
 - obtaining through TFTP 2-16
 - system management
 - command summary 5-16
 - system memory
 - displaying statistics 5-3
 - testing 5-15
 - system processes
 - changing priorities 7-3
 - displaying active 5-4
 - monitoring 5-1
 - system prompt
 - EXEC 2-8
 - privileged level 2-8
 - user level 2-8
 - using “More” with multiple screens 2-8
 - system setup
 - using the setup command facility 2-1
 - system shutdown
 - SNMP 4-36
 - system software
 - booting with Flash Memory card 2-18
 - storing with Flash Memory card 2-18
 - system startup
 - configuration script 2-4
 - first-time 2-3
 - system timeout interval
 - See timeout interval
 - Systems Network Architecture
 - See SNA
- ## T
- T1 timer
 - LAPB 8-3
 - LLC2 24-6
 - SDLC 24-14
 - TACACS
 - accounting 4-28
 - configuring 4-28
 - controlling retries 4-26
 - definition 4-22
 - establishing privileged-level 4-27
 - extended mode 4-28
 - last resort login 4-27
 - limiting login attempts 4-25
 - login authentication 4-29
 - login notification 4-29
 - optional password verification 4-30
 - privileged mode 4-27
 - server not responding 4-27
 - setting server host name 4-25
 - timeout interval 4-26
 - tacacs-server attempts command 4-26
 - tacacs-server authenticate command 4-29
 - tacacs-server extended command 4-28
 - tacacs-server host command 4-25
 - tacacs-server last-resort command 4-27
 - tacacs-server notify command 4-29
 - tacacs-server optional-passwords command 4-30

-
- tacacs-server retransmit command 4-26
 - tacacs-server timeout command 4-26
 - TCP
 - common services 7-6
 - header compression 13-41, 13-65
 - remote source-route bridging with 22-15
 - TCP connections
 - displaying status 3-6
 - TCP header compression, X.25 8-35
 - TCP transport mechanism
 - configuring HDLC 23-4
 - TCP/IP
 - running X.25 over 8-18
 - Telnet
 - port number 7-6
 - table of special commands 3-5
 - telnet command 3-1
 - Telnet connections
 - creating 3-1
 - disconnecting 3-4
 - displaying active 3-6
 - ending a session 3-3
 - escape sequence 3-2
 - incoming 3-5, 4-45
 - leaving 3-2
 - listing 3-2
 - multiple 3-2
 - naming 3-3
 - options 3-5
 - outgoing 4-45
 - restricting access 13-27
 - resuming 3-3
 - switching between 3-2
 - terminal
 - changing the screen length 3-8
 - changing the screen width 3-7
 - changing the terminal escape character 3-8
 - character padding 3-9
 - configuring from 2-13
 - displaying debug messages 3-8
 - establishing input notification 3-9
 - location setting 4-47
 - parameters, changing 3-7
 - parameters, display of active 3-7
 - parameters, listing commands 3-7
 - setting screen length 4-48
 - writing configuration to 5-14
 - terminal ? command 3-7
 - terminal access control
 - establishing 4-25
 - Terminal Access Control Access Control System
 - See TACACS
 - terminal command 2-14
 - terminal emulation
 - IBM PC/3270 22-68
 - terminal escape-character command 3-8
 - terminal exec-character-bits command 3-10, 3-13
 - terminal length command 3-8
 - terminal line
 - configuring group of 4-42
 - terminal location
 - setting 4-47
 - terminal monitor command 3-8, 4-40
 - terminal notify command 3-9
 - terminal padding command 3-9
 - terminal special-character-bits 3-10, 3-14
 - terminal width command 3-7
 - terminating processes 3-4
 - terminology
 - ISO routing 15-2
 - test interfaces command 5-15
 - test memory command 5-15
 - test sbe command 5-15
 - testing the system 5-15
 - Texas Instruments
 - Token Ring MAC firmware problem 22-68
 - TFTP
 - autoloading configuration files 4-37
 - copying image to Flash Memory 4-13
 - monitoring transactions 2-18
 - port number 7-6, 22-35
 - server, configuring 4-36
 - server, loading files from 4-9
 - use in configuration files 2-16
 - using to load configuration files 2-16
 - using to load system images 2-16
 - TFTP server
 - configuring 4-36
 - copying Flash Memory image 4-17
 - tftp-server system command 4-36
 - third-party mechanism
 - EGP 14-42
 - THT
 - FDDI 6-41

-
- timeout interval
 - ARP 13-10
 - setting EXEC 4-47
 - system default 4-48
 - system setting 4-47
 - TACACS 4-26
 - timers
 - adjustable routing 14-69
 - adjusting BGP 14-34
 - adjusting EGP 14-41
 - adjusting XNS 20-6
 - Apollo Domain 9-4
 - DECnet Phase IV 12-11
 - ignore VC 8-27
 - keepalive 8-57, 14-68
 - Novell IPX update 17-7
 - retransmission, See retransmission timer
 - token holding 6-41
 - token rotation 6-41
 - transmission valid 6-41
 - timers basic command 14-69
 - timers bgp command 14-34
 - timers egp command 14-41
 - timing signal configuration 6-9
 - token holding timer
 - See THT
 - Token Ring
 - and frame-copied errors 22-69
 - and TI MAC firmware problem 22-68
 - basic configuration example 22-83
 - configuring DECnet Phase IV on 12-12
 - configuring XNS over 20-6
 - definition of ring 22-5
 - displaying interface statistics 22-93
 - extended LAN 22-2
 - frame format 22-2
 - IBM 8209 bridges and SR/TLB 22-42
 - maintaining source-route bridging 22-90
 - NetBIOS access control filtering 22-55
 - remote source-route bridging on 22-15
 - source bridge only example configuration 22-84
 - source bridge, basic example configuration 22-84
 - source-route bridging 22-1
 - Token Ring interface
 - clearing 6-28
 - configuring 6-26
 - configuring LLC2 Local Acknowledgment 22-30
 - debug messages 6-32
 - debugging 6-32
 - displaying information 22-93
 - early token release 6-27
 - encapsulation methods 6-28
 - Level 3 and Level 2 switching 6-26
 - maintaining 6-28
 - monitoring 6-28
 - ring speed, configuring for IGS/TR 6-27
 - show interfaces field descriptions 6-29
 - specifying 6-26
 - status messages 6-32
 - support 6-26
 - token rotation timer
 - See TRT
 - trace
 - common problems 13-72
 - definition 13-71
 - description 13-72, 15-28
 - extended test 5-14
 - IP 13-71
 - ISO CLNS 15-28
 - terminating 5-14, 15-29
 - test characters table 15-29
 - tracing IP routes 13-72
 - use in debugging 5-14
 - VINES 19-13
 - trace command 5-14, 15-28
 - traffic
 - AppleTalk 10-76
 - ISO CLNS 15-21
 - Novell IPX 17-9, 17-26
 - PUP 18-3
 - STUN 23-31
 - VINES 19-13
 - traffic load threshold
 - default 6-58
 - defining 6-58
 - traffic statistics
 - displaying 12-29
 - transient ring error 6-41
 - transit bridging
 - on FDDI 21-5
 - on UltraNet 21-5
 - transit records
 - IP accounting 13-37

-
- transition states
 - FDDI 6-36
 - translating called addresses, X.25 8-19
 - translational bridging
 - compatibility with IBM 8209 bridges 22-42
 - on FDDI interface 6-34
 - See also SR/TLB
 - translations
 - supported metric 14-56
 - transmission timer
 - FDDI 6-41, 6-42
 - transmission valid timer
 - See TVX
 - transmit delay
 - serial interface 7-1
 - transmit-interface command 13-39
 - transmitter-delay command 7-1, 7-2
 - transparent bridging
 - administrative filtering 21-16
 - configuring 21-8
 - debugging 21-41
 - displaying known spanning tree topology 21-40
 - Ethernet 21-36
 - frame conversion between source-route 22-44
 - global configuration command summary 21-42
 - interface restrictions 21-11
 - interface subcommand summary 21-44
 - IP 21-10
 - load balancing 21-30
 - maintaining 21-39
 - monitoring the network 21-39
 - on FDDI interface 6-34
 - removing static entries 21-39
 - sample configurations 21-36
 - setting priority 21-13
 - spanning tree algorithm 21-4
 - spanning tree parameters, adjusting 21-13
 - special configurations 21-30
 - troubleshooting 21-39
 - viewing forwarding database 21-39
 - X.25 21-31
 - transport, datagram
 - See datagram transport
 - TRAP host
 - message length queue 4-33
 - TRAP message
 - authentication 4-35
 - resending 4-35
 - specifying recipient 4-34
 - timeout 4-35
 - Trivial File Transfer Protocol
 - See TFTP
 - troubleshooting
 - network operations 5-11
 - TRT
 - FDDI 6-41
 - tunneling
 - enabling for X.25 8-18
 - TVX
 - FDDI 6-41
 - Type of Service (TOS) field 8-41
- ## U
- UB Net/One
 - and differences between XNS 20-7
 - configuring routing 20-8
 - configuring XNS 20-7
 - on Cisco router 20-7
 - routing protocol 20-9
 - UB routing
 - configuring 20-21
 - UB XNS
 - See UB Net/One
 - UDP
 - broadcast forwarding 13-13
 - broadcasts 13-13
 - common services 7-6, 22-35
 - port prioritizing 7-6
 - unreachable messages
 - generating 13-17
 - use in RIP 14-25
 - ULA applique
 - loopback on 7-17
 - UltraNet
 - encapsulation method 6-52
 - transit bridging 21-5
 - ultranet address command 6-56
 - UltraNet interface
 - clearing 6-52
 - encapsulation methods 6-52
 - loopback 7-17
 - maintaining 6-52
 - monitoring 6-52

- show interfaces field descriptions 6-53
- specifying 6-52
- static address assignment 6-56
- support 6-51
- unit numbers, interface 6-2
- UNIX syslog server
 - logging messages to 4-40
- update broadcast
 - IGRP 14-8
- update timers
 - Novell IPX 17-7
- User Datagram Protocol
 - See UDP
- user name authentication 4-30
- user-level commands
 - definition 2-8
- username command 4-30
- username name password secret command 6-80

V

- V.25 bis dialing
 - Cisco support 6-61
 - support 6-62
- vacant banner
 - turning on/off 4-46
- vacant-message command 4-46
- variance command 14-7
- VC ignore timer
 - configuring 8-27
- vendor code
 - administrative filtering 21-23, 22-49
 - configuring access lists 22-49
 - establishing access lists 21-23
- VINES
 - access lists 19-8
 - addresses 19-2
 - ARP 19-5, 19-8
 - clearing neighbor address 19-10
 - clearing routing table 19-11
 - configuring routing 19-3
 - configuring serverless network 19-5
 - debugging the network 19-14
 - displaying interface settings 19-11
 - displaying neighbor table 19-12
 - displaying routing table 19-11, 19-12
 - displaying traffic 19-13

- encapsulation 19-6
- global configuration command summary 19-15
- ICP 19-8
- interface subcommand summary 19-16
- IPC 19-8
- maintaining the network 19-10
- monitoring the network 19-11
- name-to-address map 19-7
- ping command 19-13
- routing table 19-3
- RTP 19-3, 19-8
- SPP 19-8
- trace command 19-13
- vines 19-8
 - vines access-group command 19-9
 - vines access-list deny command 19-8
 - vines access-list permit command 19-8
 - vines arp-enable command 19-5
 - vines encapsulation command 19-6
 - vines host command 19-7
 - vines metric command 19-3
 - vines redirect-interval command 19-4
 - vines routing command 19-3
 - vines serverless command 19-6
- virtual address request and reply
 - HP Probe 13-10
- virtual circuit
 - clearing 8-26
 - clearing X.25 8-42
 - displaying active 8-43, 8-53
 - maintaining 8-26
- virtual circuit channel sequence
 - range limit keywords 8-25
- virtual circuit ranges
 - configuring 8-24
 - X.25 8-24
- virtual links 14-14
- Virtual Network System
 - See VINES
- virtual ring
 - assigning 22-38
 - definition 22-2, 22-13
 - example 22-39
 - using with LAN Network Manager 22-75, 22-78
- virtual terminal line
 - configuring 4-41

configuring group of 4-42
VMS system
loopback 7-18

W

where command 3-2
window modulus 8-30
window sizes 8-31
write erase command 2-13, 2-14, 5-14
write memory command 2-13, 2-14, 5-14
write network command 2-13, 2-15, 4-9, 5-14
write service command 2-15
write terminal command 2-13, 4-9, 5-14

X

X.121 address 8-19, 8-23
DDN conversion table 8-40
display address mapping 8-14, 8-40
mapping 8-9
setting 8-23
updating 8-29
X.25
address mapping
displaying 8-14
issues 8-8
setting 8-9
bridging on 8-35
clearing virtual circuits 8-42
command summary 8-77
communicating via routers 8-8
configuration example 8-16
configuring 8-5, 8-10
configuring ISO CLNS over 15-9
configuring transparent bridging 21-31
connecting networks via TCP/IP network 8-6
constructing routing table 8-19
datagram transport 8-6
DDN configuration subcommands 8-41
debugging 8-44
displaying interface parameters 8-42
displaying interface statistics 8-42
enabling switching 8-18
encapsulation methods 8-7
forwarding calls 8-18
frame parameters 8-4

global command summary 8-77
header compression, TCP 8-35
interface subcommand summary 8-80, 8-82
international code use G-1
IP split horizon, default 8-6
maintaining 8-34, 8-42
multiple network protocols 8-2
netbooting over 8-34
remote switching 8-6
routing through LAN 8-17
running on TCP/IP 8-18
setting interface address 8-23
setting packet sizes 8-30
single network protocol 8-2
subcommand summary 8-82
support 8-6
supported protocols 8-6
switch 8-6
switching subsystem 8-18
TCP header compression 8-35
translating addresses 8-19
X.25 configuration
switching 8-22
X.25 Level 2
configuring parameters 8-3
restart 8-33
X.25 Level 3
monitoring 8-42
retransmission timer 8-27
setting parameters 8-23
X.25 parameters
displaying interface 8-42
Level 2, LAPB 8-3
Level 3 8-23
setting frame 8-4
setting per-call 8-33
x25 accept-reverse command 8-32
x25 address command 8-23
x25 bfe-decision command 8-38, 8-77
x25 bfe-emergency command 8-38, 8-77
x25 default command 8-14
x25 facility command 8-33
x25 hic command 8-25
x25 hoc command 8-26
x25 hold-queue command 8-32
x25 hold-vc-timer command 8-27
x25 htc command 8-26

-
- x25 idle command 8-26
 - x25 ip-precedence command 8-41
 - x25 ips command 8-30
 - x25 lic command 8-25
 - x25 linkrestart command 8-33
 - x25 loc command 8-26
 - x25 map bridge broadcast command 21-31
 - x25 map bridge command 8-35, 8-86, 21-31
 - x25 map cmns command 8-46
 - x25 map command 8-9
 - x25 map compressed tcp command 8-35
 - x25 modulo command 8-30
 - x25 nvc command 8-27
 - x25 ops command 8-30
 - x25 pvc command 8-12, 8-21
 - x25 remote-red command 8-37, 8-77
 - x25 route command 8-19
 - x25 routing command 8-18
 - x25 rpoa name command 8-33
 - x25 suppress-called-address command 8-32
 - x25 suppress-calling-address command 8-31
 - x25 t10 command 8-28
 - x25 t11 command 8-28
 - x25 t12 command 8-29
 - x25 t13 command 8-29
 - x25 t20 command 8-28
 - x25 t21 command 8-28
 - x25 t22 command 8-29
 - x25 t23 command 8-29
 - x25 th command 8-31
 - x25 use-source-address command 8-29
 - x25 win command 8-31
 - x25 wout command 8-31
 - X3T9.5
 - specification 6-41
 - Xerox Network Systems
 - See XNS
 - Xerox PARC Universal Protocol
 - See PUP
 - XNS
 - access lists, configuring 20-15
 - adding filters to routing table 20-18
 - addresses 20-2
 - adjusting timers 20-6
 - broadcasts, handling 20-10
 - cache entries, displaying 20-22
 - configuration examples 20-19
 - configuring for SMDS 8-72
 - configuring over Token Ring 20-6
 - configuring routing 20-3
 - debugging the network 20-24
 - displaying cache entries 20-22
 - displaying interface parameters 20-22
 - displaying routing table 20-23
 - displaying traffic statistics 20-23
 - dynamic routing 20-4
 - enabling fast switching 20-5
 - enabling routing 20-3
 - encapsulation 20-2, 20-6
 - extended access lists 20-16
 - filtering packets 20-17
 - filtering routing updates 20-17
 - filters, configuring 20-15
 - flooding, overview 20-10
 - forwarding specific protocols 20-14
 - global configuration command summary 20-25
 - helping, overview 20-10
 - input filters 20-18
 - interface subcommand summary 20-26
 - managing throughput 20-4
 - monitoring the network 20-22
 - multiprotocol routing, configuring 20-20
 - output filters 20-18
 - router filters 20-19
 - routing process, creating 20-19
 - routing protocol 20-1
 - setting multiple paths 20-5
 - setting timers 20-20
 - standard access lists 20-15
 - static routing 20-4
 - xns access-group command 20-17
 - xns encapsulation command 20-6
 - xns flood broadcast allnets command 20-12
 - xns flood broadcast net-zero command 20-12
 - xns flood specific allnets command 20-12
 - xns forward-protocol command 20-12
 - xns hear-rip command 20-8
 - xns helper-address command 20-11
 - xns input-network-filter command 20-18
 - xns maximum-paths command 20-5
 - xns network command 20-4
 - xns output-network-filter command 20-18
 - xns route command 20-4
 - xns route-cache command 20-5

xns router-filter command 20-19
xns routing command 20-3
xns ub-emulation command 20-8
xns ub-routing command 20-9
xns update-time command 20-6
XNS, configuring for SMDS 8-72

Z

zero, subnet

See subnet zero

ZIP

AppleTalk routing protocol 10-3, 10-7

Zone Information Protocol

See ZIP

zones

AppleTalk 10-5

AppleTalk extended 10-10

assigning AppleTalk 10-14

Corporate Headquarters

Cisco Systems, Inc.
P.O. Box 3075
1525 O'Brien Drive
Menlo Park, CA 94026
USA
Tel: 415 326-1941
800 553-NETS (6387)
Fax: 415 326-1989

European Headquarters

Cisco Systems Europe, s.a.r.l.
BP 706 Evotic
16 avenue du Quebec
91961 Les Ulis Cedex
France
Tel: 33 1 6092 2000
Fax: 33 1 6928 8326

European Offices

Belgium

Tel: 32 2 643 2626
Fax: 32 2 643 2627

Germany

Tel: 49 89 3215 070
Fax: 49 89 3215 0710

Italy

Tel: 39 2 62 726 43
Fax: 39 2 62 729 13

Spain

Tel: 34 1 57 203 60
Fax: 34 1 57 071 99

Sweden

Tel: 46 8 19 62 05
Fax: 46 8 19 04 24

Switzerland

Tel: 41 55 95 60 44
Fax: 41 55 95 64 14

United Kingdom

Tel: 44 494 464944
Fax: 44 494 465300

Intercontinental Headquarters

(Latin America and Asia-Pacific)

Cisco Systems, Inc.
1525 O'Brien Drive
P.O. Box 3075
Menlo Park, CA 94026
USA

Tel: 415 326-1941
Fax: 415 688-4646

Regional Offices

Cisco Systems Australia Pty., Ltd.

Tel: 61 2 957 4944
Fax: 61 2 957 4077

Cisco Systems Canada Limited

Tel: 416 506-1500
Fax: 416 506-1506

Cisco Systems Hong Kong, Ltd.

Tel: 852 529 3534
Fax: 852 520 2676

Cisco Systems de México, S.A. de C.V.

Tel: 525 254 0880
Fax: 525 531 9659

Cisco Systems New Zealand

Tel: 9 649 358 3776
Fax: 9 649 358 4442

Japanese Headquarters

Nihon Cisco Systems K.K.
Shiba Excellent Building, 5F
2-1-13 Hamamatsucho,
MinatoKuTokyo 105, Japan
Tel: 81 3 5472 3571
Fax: 81 3 5472 3577

Cisco Systems has over 50 sales offices worldwide. Call 415 326-1941 to contact your local account representative or, in North America, call 800 553-NETS (6387)

