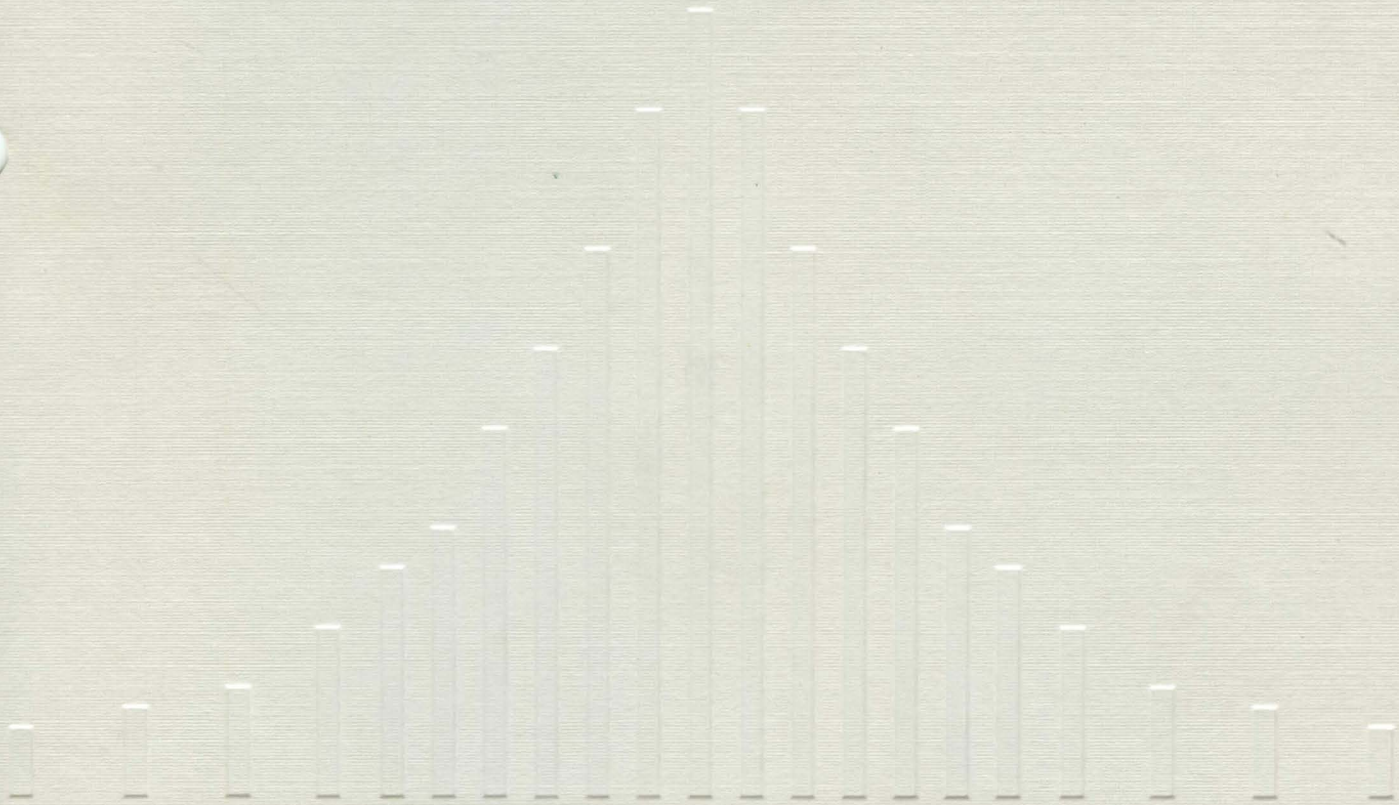
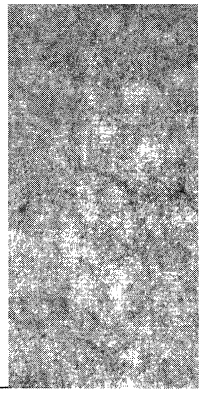
...the science of networking networks

Router Products Configuration and Reference Volume I



CISCO SYSTEMS



Router Products Configuration and Reference Volume I

Cisco Systems, Inc.
October 1991
Software Release 8.3

Corporate Headquarters:
Cisco Systems, Inc.
1525 O'Brien Drive
Menlo Park, California 94025 USA
1-415-326-1941
1-800-553-NETS
FAX 1-415-326-1989

Customer Order Number: DOC-R8.3
Cisco Document Assembly Number: 83-0017-01
Text Part Number: 78-0828-01

CISCO SYSTEMS

Router Products Configuration and Reference

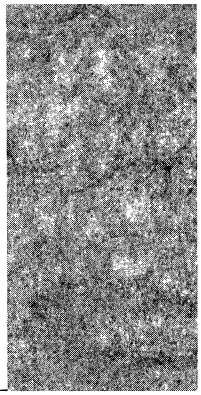
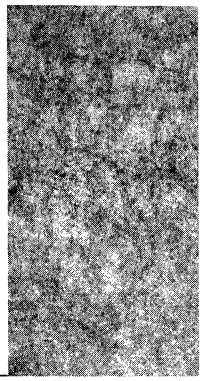
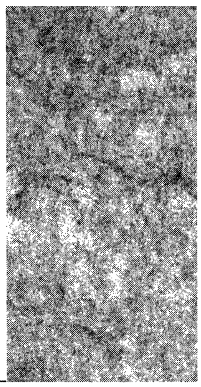


Table of Contents

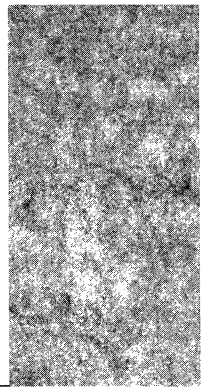
Chapter 10 Routing AppleTalk 10-1



List of Figures



List of Tables



About This Manual

This front matter section introduces the *Router Products Configuration and Reference* audience and scope, organization, use, and conventions.

Audience and Scope

This publication addresses the system administrator who will configure and maintain a Cisco gateway, router, or bridge running Release 8.3 and earlier software. (Software changes from previous releases are noted in the release note accompanying this manual.)

Document Organization and Use

The *Router Products Configuration and Reference* provides information about using and configuring the Cisco network servers—use and administration of the system interface, system configuration, and network configuration. This publication and a software release note are included with each Cisco network server system.

To match the multiprotocol, multirouting, and bridging capabilities of the Cisco network server product line, this publication has been divided into parts, contained in which are chapters describing information about a particular task, protocol, or command function. You will find dividers separating this publication into these parts:

Volume I

- **Part One: Product Overview,** contains an overview of the Cisco network server product line.
- **Part Two: System Configuration and Management,** contains chapters that provide system start-up procedures, information about the system command interpreter, and system configuration procedures. Begin your system configuration process in this part of the manual.

- **Part Three: Interface Configuration,** contains chapters that provide information and procedures for configuring the system interfaces. This information includes special configuration techniques, and procedures to examine the interfaces for correct operation. Become familiar with the information in this part of the manual, and refer to it while configuring your Cisco router/bridge for its particular task.

Volume II

- **Part Four: Routing Configuration,** contains chapters that describe how to configure each Cisco-supported routing protocol. These protocols include the AppleTalk, Apollo Domain, Banyan VINES, CHAOSnet protocols, Internet Protocol (IP), ISO Connectionless Network Services (CLNS), Novell IPX, Phase IV DECnet, PUP, and Xerox XNS (including Ungermann-Bass and 3Com). The chapters are arranged in alphabetical order, for ease of use.
- **Part Five: Bridging and Connectivity,** contains chapters that describe how to configure transparent and source-route bridging on the Cisco router/bridges, and how to configure the SDLC transport and serial-tunneling mechanisms in an IBM local area network.
- **Appendices,** which contain system error messages, a summary of the access list commands, a technical brief describing regular expressions to use for pattern-matching operations, a list of Ethernet type codes, an ASCII chart, and a list of references and recommended reading.
- **Glossary,** which contains definitions of terms and concepts described in this manual.
- **Index,** which provides page references to the topics and commands covered in this publication.

Document Conventions

The command descriptions use these conventions:

- Commands and keywords are in **boldface**.
- Variables for which you supply values are in *italics*.
- Elements in square brackets ([ ]) are optional.
- Alternative but required keywords are grouped in braces ({ }) and are separated by a vertical bar (|).
- A string is defined as a nonquoted set of characters. For example, when setting up a community string for SNMP to "public", do not use quotes around the string or the string will be set to "public".

The samples use these conventions:

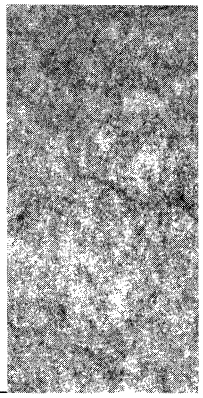
- Terminal sessions are printed in a `screen font`.
- Information you enter is in a **`boldface screen`** font.

- Nonprinting characters are shown in angle brackets (<>).
- Information the system displays is in `screen font`, with default responses in square brackets ([ ]).

This publication also uses the following conventions:

Note: is a special paragraph that means *reader take note.* It ususally refers to helpful suggestions, the writer's assumptions, or reference to materials not contained in this manual.

Caution: is a special paragraph that means *reader be careful.* It means that you are capable of doing something that might result in equipment damage, or worse, that you might have to take something apart and start over again.



Service and Support

Cisco Systems provides a full range of support services to ensure that you get maximum network uptime with low life-cycle equipment cost.

Warranty Information

All Cisco Systems products are covered under a limited factory warranty. This warranty covers defects in the hardware, software, or firmware. Refer to the Cisco Systems *Customer Services Product Guide* for more information on Cisco's warranty policy, or contact Cisco Systems at 1-800-553-NETS or 1-415-326-1941.

Note: Warranty and other service agreements may differ for international customers. Contact your closest Cisco regional representative for more information.

Maintenance Agreements

Cisco Systems offers Comprehensive Maintenance Agreements throughout North America, which include on-site remedial services, software support, a 24-hour emergency hot line, overnight parts replacement, and an escalation procedure. Cisco also offers Software and Advanced Replacement Service under a SMARTnet agreement for customers who desire those services. Noncontract maintenance services are provided at current time-and-materials rates. For more information contact Customer Services at 1-800-553-NETS or 1-415-326-1941.

Our maintenance strategy is founded upon customer-initiated service requests to the Cisco Systems Technical Assistance Center (TAC). The TAC coordinates all customer services, including hardware and software telephone technical support, on-site service requirements, and module exchange and repair.

The TAC is available during normal business hours from 6:00 A.M. to 6:00 P.M. Pacific coast time, Monday through Friday excluding company holidays, at the TAC telephone number listed below. If you must return your Cisco equipment for repair or replacement, please contact the TAC or a Cisco regional representative for more information.

Customer Support

Hardware and software support specialists are available to help diagnose and solve customer problems. They will be able to isolate and solve your problem much faster if you are prepared with the information they need.

When you call have the following information ready:

- Chassis serial number
- Maintenance contract number
- Software version

You can display your software version level and your hardware configuration by using the **show hardware** command.

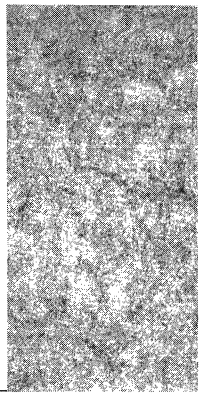
How to Contact Customer Support

For technical assistance:

TAC Telephone Number:	1-800-553-2447	**FAX:**	1-415-688-7878
	1-415-688-8209	**Email:**	`tac@cisco.com`

For sales and order information:

Phone:	1-800-553-NETS (6387)	**FAX:**	1-415-903-8080
	1-415-903-7208	**Email:**	`csrep@cisco.com`



Obtaining Additional Information

This front matter section describes how to obtain other Cisco publications.

Information About Networks in General

For a list of texts with specific information about routing, bridging, and the protocols supported by Cisco Systems, see Appendix F, "References and Recommended Reading."

Ordering Additional Cisco Publications

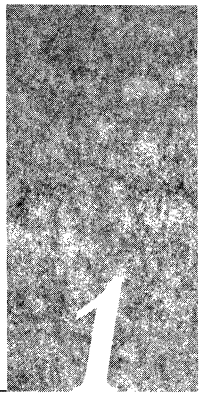
Cisco includes a *Router Products Getting Started* publication with router/bridge systems that covers basic system installation and configuration. These publications are also available:

- *Field Service Manual*—Covers hardware troubleshooting and replacement procedures for all Cisco hardware products. Order this publication from Cisco Systems Customer Services.
- *Modular Products Hardware Installation and Reference* (for users with Cisco's modular products), or *IGS Hardware Installation and Reference* (for users with the IGS product)—Covers device installation and start-up procedures, and hardware upgrade procedures for all Cisco products. One of these publications is included with each system, as appropriate.
- *Terminal Server Configuration and Reference*—Offers information about the use of the Cisco terminal server, to include use of the system interface, system configuration, and network and system administration. This publication and associated release notes and addendums are included with each terminal server.
- *Protocol Translator Configuration and Reference*—Offers information about the use of the Cisco Protocol Translator, to include use of the system interface, system configuration, and network and system administration. This publication and associated release notes and addendums are included with each protocol translator.

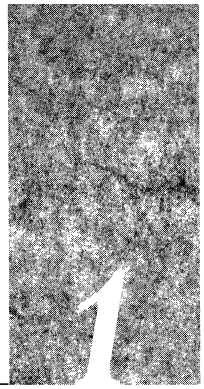
Current price information for all Cisco publications are included in the *Cisco Systems North American Price List*. Consult this guide or contact your sales representative to order these publications. Publications can be ordered directly from Cisco at:

Cisco Systems, Inc.
1525 O'Brien Drive
Menlo Park, California 94025 USA
1-800-553-NETS or 1-415-326-1941

Part 1
Product Overview



Chapter 1
Cisco Product Line Overview



Chapter 1
Cisco Product Line Overview

This chapter introduces Cisco's product line and provides the following information:

- Cisco's internetworking philosophy
- Terminology and examples of network configuration
- Media and protocols supported by Cisco routers
- Management and security software features
- Hardware configurations and connection options

Internetworking and Cisco

Cisco Systems designs and produces communications products to build multiprotocol, multimedia, multivendor networks. With Cisco Systems products, network managers can interconnect terminals, workstations, computer systems, and different networks to create networks of networks.

Products from Cisco Systems solve the problems of complex internetworks by supporting multiprotocol, multimedia, multivendor networks. Even organizations that do not yet require a complex wide area network (WAN) can connect existing equipment with Cisco Systems products to form a network of any size. With Cisco Systems products, network growth plans need not depend on a single topology, protocol, transmission medium, or vendor.

The key device for building a trouble-free WAN is a Cisco Systems network server. A Cisco Systems network server is a dynamic router that provides a wide variety of network management capabilities in addition to the multiprotocol, multimedia, multivendor internetworking already described.

Just as in interconnecting local area networks (LANs), Cisco Systems network servers help build WANs that achieve the interoperability and connectivity large organizations need. Network servers from Cisco Systems support computers and networking equipment from all vendors, using all available media and virtually any protocol.

A WAN that uses Cisco Systems network servers is both more cost-effective and more efficient. Cisco Systems network servers help organizations take advantage of all their existing computers and networking equipment. The dynamic routing of the network servers ensures that network traffic reaches the intended addresses promptly, regardless of different protocols, equipment from different vendors, and ongoing network configuration changes.

Terminology

A WAN interconnects LANs, host computers, and public and private networks into a communications network that spans a large geographic area. After organizations have become accustomed to the benefits of LANs, they often want to extend their networking capabilities by internetworking different LANs or connecting remote sites to large networks.

An organization may already own hundreds of microcomputers and mainframes manufactured by different vendors, as well as several different kinds of LANs such as Ethernet and Token Ring. Depending on its networking needs, the organization may consider using the following devices to extend its network:

- Terminal servers connect terminals, modems, and microcomputers over serial lines to LANs or WANs. They provide network access to terminals, printers, and computers that have no built-in network support.
- Repeaters connect two network segments that use the same medium, simply regenerating the transmission signal between them. Repeaters protect each network segment from electrical failures of other segments. However, repeaters cannot protect the network from data-related errors.
- Bridges connect two network segments that use the same medium. Bridges protect the resulting network from electrical problems and data-related errors, but not from problems related to higher-level protocols.
- Routers can interconnect networks over longer distances and over certain different media. Because routers support hierarchical network structures, they offer protection from network errors as well as electrical and data-related problems.
- The term *gateway* originally referred to an IP routing device, and was used as part of the naming schemes in early Cisco products. You will find the term gateway used as part of the Cisco software (for example, the Interior Gateway Routing Protocol); however, throughout this manual, we use the term *router* to refer to the device configured for routing, and the term *router/bridge* to refer to the device configured for bridging. The term *network server* refers to Cisco's full line of routing and bridging devices.

Capabilities of the Router

Complex internetworks have grown past the point where they can depend on equipment from a single vendor. Virtually all organizations connecting LANs and creating WANs today have major commitments to hardware and software from many different vendors. Therefore, current and future internetworking requires products that support multiprotocol, multimedia, and multivendor networks.

Routers from Cisco Systems help LANs and WANs achieve interoperability and connectivity. They operate with equipment from all vendors over most available media. This section describes Cisco-supported protocols and media, as well as the capabilities of Cisco Systems routers for routing, network management, and network security.

Support for Multiple Protocols

Large organizations need the flexibility that multiprotocol networks give them to communicate with diverse hardware and software from many vendors. Cisco Systems routers support many networking protocols, as well as several specific routing protocols for compatibility with other networks. Included are protocols based on open standards and proprietary protocols from a variety of vendors.

One Cisco Systems router can forward packets concurrently from any combination of the following networking protocols:

- TCP/IP protocols, the most widely implemented protocol suite on networks of all media types. TCP/IP is today's standard for internetworking, and is supported by most computer vendors, including all UNIX-based workstation manufacturers.
- DECnet Phase IV protocol, Digital Equipment's proprietary networking protocol used on DECnet.
- XNS (Xerox Network Service) protocol, used by Xerox, 3COM, and other XNS vendors.
- Novell IPX, Novell's "External Bridge" protocol (Novell refers to their router functionality as *bridging*).
- Ungermann-Bass protocol, used by Ungermann-Bass routing products and other vendors of these routing products.
- AppleTalk (Phase 1 and Phase 2) protocol, Apple Computer's multiprotocol internetwork product.
- ISO Connectionless Network Services (CLNS) routing protocol, as defined by ISO documents 8473, Connectionless Network Protocol (CLNP), and 9542, End System-Intermediate System (ES-IS) Routing Exchange Protocol. The Cisco ISO CLNS routing implementation is compliant with the Government Open Systems Interconnection Profile (GOSIP).
- Apollo Domain network protocols, the native mode networking protocol for Apollo workstations.
- Banyan VINES networking protocol for networking personal computers.
- Xerox's Universal Protocol (PUP), developed at Xerox's Palo Alto Research Center (PARC), used by some Xerox workstations.
- CHAOSnet protocol, used by LISP machine vendors and other organizations in the AI community.
- X.25 protocols, which permit cost-effective, as-needed use of major public networks world wide. Cisco Systems products support both the X.25 protocol and the X.3/X.28/X.29 specifications for X.25 terminal service.
- DDN protocols, as used with the DDN X.25 Standard, 1822-LH/DH, and HDH (1822-J) attachments. The U.S. Department of Defense specifies DDN X.25 Standard for all new attachments to the Defense Data Network; many established research and military networks use 1822-LH/DH.

- Frame Relay serial encapsulation, which conforms to the Frame Relay Interface specification produced by Cisco Systems, StrataCom, Northern Telecom and Digital Equipment Corporation; allows a single, physical connection between the network server and a switch which provides service.
- Switched Multi-Megabit Data Service (SMDS) protocol, a cell relay technology that provides wide area networking service through service providers.
- Point-to-Point Protocol (PPP), an IP encapsulation method (RFC 1134) that encapsulates IP datagrams and other Network Layer Protocol information over point-to-point links.

Support for Standard Media

For convenient access to existing networks, Cisco Systems network servers support these industry standard networking media:

- Ethernet (IEEE 802.3)
- Token Ring (IEEE 802.5)
- FDDI
- Synchronous serial
- High Speed Serial Interface (HSSI)
- Ultranet

Network servers from Cisco Systems support synchronous serial circuits at many speeds. Customers may use 9.6 and 19.2 kilobits-per-second synchronous serial service, and 56 kilobits-per-second service for medium-traffic connections.

For fast serial service over routes with heavy data needs, Cisco's full duplex, High Speed Serial Interface (HSSI) is capable of transmitting and receiving data at up to 52 Mbps, and supports connectivity to T3, E3, SMDS at DS-3 and other high speed wide area services.

Cisco Systems network servers support T1 circuits at 1,544 kilobits per second, T1C circuits at 3,088 kilobits per second, and British Telecom Megastream and CEPT circuits at 2,048 to 4,096 kilobits per second.

Cisco Systems markets a broad line of synchronous serial media adapters, including RS-232, V.35, and RS-449 to allow their routers to convert to equipment such as modems and DSUs that utilize these interfaces.

Dynamic Network Routing

Transferring data among networks is the primary task of a network server, a task whose efficiency requires up-to-date information about network status. Dynamic network routing, used by Cisco Systems routers, automatically responds to network changes to ensure faster, more reliable packet routing. Continually updated information enables the routers to find the most reliable links and routes with fastest transmission.

A Cisco Systems router monitors traffic on each network link connected to it, and routes the traffic to the appropriate destinations. Because the router is an "intelligent" router, it sends each network segment only the packets destined for it. No segment is burdened by unnecessary traffic.

The Interior Gateway Routing Protocol (IGRP™), developed by Cisco Systems for TCP/IP and ISO CLNS, monitors the network to determine the status of each route and select the best route for each data packet. Network traffic, path reliability, and path speed all influence route selection. Cisco Systems specifically designed IGRP to address the problems of routing on complex networks with many alternative routes, built of media with diverse bandwidth and delay characteristics.

While running IGRP, Cisco Systems routers can concurrently receive and understand messages from other network segments sent using different routing protocols. For example, IP routing protocols supported by Cisco Systems, in addition to IGRP, include:

- Routing Information Protocol (RIP) is the interior routing protocol used by the routing process on Berkeley-derived UNIX systems.
- Exterior Gateway Protocol (EGP) is the routing protocol used by all routers attached to the DDN. The Cisco Systems implementation maintains contact with multiple EGP-speaking routers, preserving routing information when the DDN core routers do not respond.
- Border Gateway Protocol (BGP) is a replacement protocol for EGP. BGP is defined by RFC 1163.

All Cisco Systems routers support the native, dynamic routing protocols used by the various supported network protocols, such as DECnet, Novell IPX, and AppleTalk. This allows compatibility with other vendor's routers. Multiple network protocols and their dynamic routing protocols operate concurrently, sharing the same router and media.

- When routing DECnet packets, the Cisco Systems router acts as a DECnet Level 1 and/or Level 2 router.
- Cisco Systems routers can route XNS and XNS variants such as Novell IPX and Apollo Domain using variations of the RIP routing protocol.
- Cisco's AppleTalk routing implementation uses the Routing Table Management Protocol (RTMP) to provide dynamic routing information.
- Cisco's ISO Connectionless Network Services (CLNS) protocol supports ISO 9542 defining the End System-Intermediate System (ES-IS) routing exchange protocol, and ISO 8473 Connectionless Network Protocol (CLNP).

Network Management Software

Cisco Systems provides a full range of network management tools that network managers and system administrators will find invaluable in the day-to-day operations of their networks.

Dynamic Versus Static Routing Tables

Most network managers must resolve problems that arise from line failures, overloads, equipment outages (either planned or unplanned), and changes to network interconnections. With static routing tables, network managers must spend time developing and installing other routes.

With Cisco Systems' dynamic routing, routers automatically handle routing problems and optimize traffic flow. The network manager can concentrate on true management issues such as network planning, capacity management, and meeting the needs of different user groups.

With Cisco Systems' Interior Gateway Routing Protocol (IGRP), packet routing and flow optimization take place automatically. Therefore, organizations can concentrate on issues such as planning for network growth, high-level network troubleshooting, and user support.

Diagnostic Tools

Network servers from Cisco Systems also provide detailed network management statistics, including traffic statistics, and counts of messages transmitted and received. Remote echo and route-tracing diagnostics help network managers isolate faults and refine network measurements. Cisco Systems also supports the Simple Network Management Protocol (SNMP), the industry standard for network management.

NetCentral Software

Internet work management can be as easy as clicking a mouse button with the Cisco Systems' NetCentral™ software—a dynamic, user-configurable network map operating on a fully integrated relational database.

NetCentral is a high-performance software tool for management of multivendor internetworks. It is designed for network monitoring and in-depth network planning and analysis through use of a dynamic visual network map and integrated relational databases.

The NetCentral network map provides network managers with an instant visual status check of the entire network. Pop-up windows and icons provide real-time statistics for remote networks and devices, so that network analysts and administrators can maintain an accurate picture of their internetwork topology.

NetCentral software operates on the Sun 3/xx series, the SPARCstation workstations, and on the Solbourne workstation. Contact Cisco Systems for more information.

Network Security

Network security is an increasingly important aspect of managing complex networks. Cisco Systems network servers enable network managers to implement several different security features. Optional passwords limit access to the privileged command set, as well as to console and terminal lines. Access lists restrict transmissions to only the specified addresses, whether identifying server ports, hosts, or routers.

Packet-type control specifies which packets may pass, such as mail packets, file transfers, and remote logins. This access-by-service security operates on top of access-list control for complete flexibility and security. Cisco Systems also supports DDN security options for IP packets.

Maintaining Networks

As people use networks more, they come to depend on their capabilities. Network users begin to take for granted that the network will always be available. Network managers face the challenge of meeting this expectation.

At Cisco Systems, hardware reliability is a top engineering priority. Equipment failures rarely occur because all Cisco Systems products meet rigid testing standards.

If problems do occur, Cisco Systems' onboard diagnostic software helps isolate them quickly, enabling customers to identify problems and arrange repairs without being familiar with the equipment. Cisco Systems engineers are always available to help with failure diagnosis, if necessary.

Cisco Systems' standard service includes low-cost board exchange, with replacements delivered by next-day express service. A premium service contract provides 24-hour coverage throughout the continental United States. On-call, on-site service by Cisco Systems field engineers is available anywhere in the world. More information about Cisco's service and support is found in the *Customer Services Product Guide.*

Modular Components for Flexible Configuration

Part of the power and flexibility of Cisco Systems product components is derived from their modular physical configuration options. Customers can choose the chassis, processor, back-panel connector mountings, and communications interfaces best suited to their network.

Chassis Options

The chassis encloses a power supply, component cards, and a backplane. Six models are available for the network server.

- The AGS+™ model is also built on the Cisco A-chassis, and provides a five-slot proprietary cBus™ backplane within the nine-slot standard backplane. The cBus backplane allows connection of Cisco Systems' high-speed interface cards, including the FDDI

controller. It provides interaction with an environmental monitor that oversees an operating environment equipped with a larger power supply and a larger number of interface cards than are available in the AGS model.

- The AGS™ model is built on the Cisco A-chassis, a nine-slot, rack-mounted chassis for network servers in large network configurations requiring high fanouts.
- The MGS™ model is built on the M-chassis, a mid-range, four-slot, tabletop or rack-mountable chassis for medium-sized network servers.
- The CGS™ model is built on the C-chassis, a compact two-slot chassis, designed for remote network servers in an office or desktop environment. This model, which includes a quiet fan, is available in a limited number of configurations only.
- The IGS™ model is a single-board router with two network interfaces. The IGS is designed for remote offices, or to interconnect PC LANs.
- The CRM is Cisco's single board router module with two interfaces that is installed in a Cabletron System MMAC hub.

Processor Options

For high-speed operation, the Cisco network servers use processors based on the MC68020 microprocessor. The Cisco Systems processors contain onboard RAM, system ROM holding all operating system, bootstrap, and diagnostic software, and hardware and software support for a control console.

The CSC/2 processor offers one megabyte of RAM, and is suitable for medium performance applications.

The CSC/3 is Cisco's high-end processor offering four megabytes of RAM. The additional RAM increases the size of the maximum routing table. The CSC/3 processor card provides the increased switching speeds necessary to support Cisco's high-speed network interfaces.

Cisco Systems also offers optional nonvolatile memory that retains configuration information despite power losses or system reboots. With the nonvolatile memory option, the terminal and network servers need not rely on other network servers for configuration and boot service information.

Connector Panel Options

Each chassis model accepts connector panels (also referred to as appliques) in numerous formats, enabling easy configuration of network servers to meet current and future needs. Supported connectors include:

- FDDI standard Media Interface Connector (MIC)
- 25-pin RS-232C D connectors (DCE and DTE)
- 15-pin Ethernet connectors
- 9-pin (DE-9) Token Ring connectors
- V.35 connectors (DCE and DTE)

- RS-449 connectors (DCE and DTE).
- High Speed Serial Interface (HSSI) Connector
- Ultranet connector

Interface Options

Cisco Systems offers a full range of interface options on their network interface cards:

- The Multiport Communications Interface (MCI™) card provides up to two Ethernet ports and up to two synchronous serial ports on a single card. The Ethernet ports support Ethernet versions 1 and 2 and IEEE 802.3 electrical interfaces. The serial ports support RS-232, V.35, RS-449, and X.21 connections.

  The interface processes packets rapidly, without the interframe delays typical of other Ethernet interfaces. By minimizing packet processing time, the MCI card achieves high-circuit utilization.

- The Serial-Port Communications Interface (SCI™) card provides four high-speed serial ports on a single card that support RS-232, V.35, and RS-449, and X.21 connections.

  The serial interfaces on both the MCI and SCI Interface cards support signaling rates of 2.4 kilobits per second to 4 megabits per second. Each synchronous channel can perform full-rate transmission, independent of packet size or the load on adjacent channels.

- The High Speed Serial Interface (HSSI), which consists of the High Speed Serial Communications Interface (HSCI™) card and applique, provides a single, full duplex, synchronous serial interface capable of speeds to 52 megabits per second. The HSSI is a de facto industry standard for providing connections to high speed WAN services such as TS, E3, SMDS at DS-3 via the proper conversion device. These cards may be used only in the AGS+ chassis featuring the cBus high-speed controller card.

- The Ultranet interface, which consists of the HSCI and applique, provides a single, full duplex serial interface capable of speeds up to 125 megabits per second. The Ultranet interface connects to the products from Ultra Network Technologies. These cards may be used only in the AGS+ chassis featuring the cBus high-speed controller card.

- The Multiport Ethernet Controller (MEC™) interface card provides two, four, or six high-speed Ethernet connectors compatible with Versions 1 and 2, and the IEEE 802.3 protocol. These cards may be used only in the AGS+ chassis featuring the cBus high-speed controller card.

- The FDDI Controller Interface (CSC-FCI) card provides a Class A, dual-attach interface for connection to the FDDI standard 62.5/125 micron fiber optic cable using the Media Interface Connector (MIC). These cards may be used only in the AGS+ chassis featuring the cBus high-speed controller card.

- The Cisco 4/16 Token Ring interface (CSC-R16) card provides a single, user-selectable 4 or 16 megabit per second connection to a Token Ring LAN.

- The Cisco Token Ring interface (CSC-R) card provides service to IEEE 802.5 Token Rings running at 4 megabits per second.

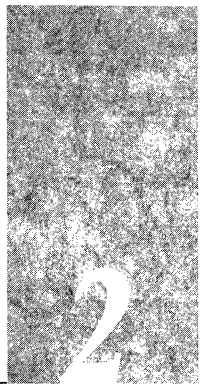
  Both of Cisco's Token Ring interface controller cards extends network service to microcomputers and IBM LANs in an easily expandable fashion.

The Cisco *Modular Products Hardware Installation and Reference* and the *IGS Hardware Installation and Reference* describe Cisco's hardware products and provides procedures for installing them into your system. Refer to these manuals for hardware-specific information.

Part 2
System Use and Management

Chapter 2
First-Time Startup and Basic Configuration

Chapter 2
First-Time Startup and Basic Configuration



This chapter describes basic system startup and use. Information in this chapter will help you with these tasks:

- First-time startup and basic configuration of the Cisco router/bridge using the **setup** command facility
- Understanding and using the system's command interpreter
- Entering configuration commands into a configuration file, and saving the configuration

Using the Setup Facility for Basic Configuration

The **setup** command facility enables you to start using your Cisco network server quickly and without extensive background knowledge. It does this by prompting you for the information required to perform basic configuration procedures. The **setup** command facility is available on all Cisco Systems internetworking products with Release 8.2 or later software.

Use the **setup** command facility both at initial system configuration and for basic changes at any time. In addition, use the facility as a teaching tool to become familiar with the expected command sequence as you step through the process. Because of these unique characteristics, Cisco refers to **setup** as a *command facility* rather than simply as a command.

Refer to the *Router Products Getting Started* publication for detailed information and a step-by-step description of the configuration procedure using the **setup** command facility.

Capabilities of the Setup Command Facility

Use the **setup** command facility to:

- Establish host names
- Set enable (or privileged mode) passwords
- Set virtual terminal passwords
- Enable SNMP network management
- Enable routing of protocols
- Enable transparent Ethernet bridging

Configure the following protocols with the **setup** command facility:

- IP, including IGRP and RIP dynamic routing
- DECnet
- XNS
- Novell IPX
- Appletalk Phase 1 and Phase 2
- CLNS
- VINES

For more advanced applications, you need to enter a privileged configuration session, as described in the section "Entering Configuration Mode" later in this chapter.

Using the Setup Command Facility

The **setup** command facility operates automatically the first time you power-on your network server and when you add new hardware components. To use **setup** on subsequent occasions, you must invoke it as you would any other command, by typing it at the EXEC prompt (described in the section "Entering Configuration Mode" later in this chapter).

Before you start using the **setup** command facility, you need to do the following:

Step 1: Attach an RS-232 ASCII terminal to the system console port located at the rear of the network server.

Refer to the Cisco publications, *Modular Products Hardware Installation and Reference* or the *IGS Hardware Installation and Reference,* for details about cabling considerations and establishing electrical connections.

Step 2: Configure the terminal to operate at 9600 baud, 8 data bits, no parity, 2 stop bits.

Step 3: Power on the network server and run the **setup** program.

Note: Network connections are not required in order to effectively use the **setup** command facility.

In addition, you need to know the following before you start:

- Which protocols you plan to route.

  Note that most protocols will prompt you for specific parameters, including host name, network numbers, addresses, and subnet masks (when applicable).
- Types of interfaces installed: Ethernet, Serial, Token Ring, or FDDI.
- Whether or not you plan to use bridging.

First-Time System Startup

The **setup** command facility determines which interfaces are installed and prompts you for configuration information for each one. Once you complete one interface, the facility automatically starts over for the next, continuing until each interface has been configured.

Note: Once you start the **setup** facility, the system runs through the entire configuration process; you cannot quit out of it. If you want to make a change or correct a mistake, simply press the Return key through the prompts, then restart the command.

When you first power on your console and network server, a script similar to the following will appear on the screen. The first section of the script displays the banner information, including the software version:

```
                Restricted Rights Legend

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 sec. 52.227-19 and subparagraph
(c) (1) (ii) of the Rights in Technical Data and Computer
Software clause at DFARS sec. 252.227-7013.

                cisco Systems, Inc.
                1525 O'Brien Drive
                Menlo Park, California 94025



GS Software (GS3-BFX), Version 8.3(1)
Copyright (c) 1986-1991 by cisco Systems, Inc.
Compiled Wed 14-Aug-91 13:31 by block
```

The next portion of the display is a list of the installed hardware. By reading the installed hardware, the system automatically presents the appropriate interfaces during the configuration process.

```
CSC3 (68020) processor with 4096K bytes of memory.
X.25 software.
Bridging software.
1 MCI controller.
2 Ethernet/IEEE 802.3 interface.
3 Token Ring/IEEE 802.5 interface.
2 Serial network interface.
16K bytes of multibus memory.
32K bytes of non-volatile configuration memory.
```

The first two sections of the configuration script (the banner and the installed hardware) appear each time the system is started up.

At the first-time system startup, the System Configuration Dialog automatically appears, offering the prompts for which you'll provide the answers to configure your system.

```
        --- System Configuration Dialog ---

At any point you may enter a question mark '?' for help.
Refer to the 'Getting Started' Guide for additional help.
Default settings are in square brackets '[]'.

Would you like to enter the initial configuration dialog? [yes]:
```

At this point, you may choose not to continue with the System Configuration Dialog and exit by answering "no" to this prompt.

Answer "yes" to continue with the **setup** configuration dialog. The remainder of the script is the actual configuration process, with each prompt appearing automatically. Press the Return key to accept the default settings.

There is no default for the final prompt; you must answer either "yes" or "no" as to whether you will use this configuration. Also note that the **setup** command only asks you to configure the protocols for each interface that you specified on a global basis. For instance, if you said "no" for XNS under the global parameters, the command does not prompt you to configure that protocol under the interface parameters.

Sample Configuration Session:

```
Configuring global parameters:

  Enter host name [Gateway]: SandBox
  Enter enable password: shovel
  Enter virtual terminal password: hammer
  Configure SNMP Network Management? [yes]:
  Configure IP? [yes]:
    Configure IGRP routing? [yes]:
      Your IGRP autonomous system number [1]: 109
  Configure DECnet? [no]: y
    Your area number [1]: 55
    Your node number [1]: 87
    Area (level 2) routing? [no]: y
  Configure XNS? [no]: y
  Configure Novell? [no]: y
  Configure AppleTalk? [no]: y
    Extended networks? [yes]:
  Configure CLNS? [no]: y
    CLNS router tag [area_1]:
    CLNS domain [49]:
    CLNS area [0001]:
    CLNS station id [0000.0C00.0A83]:
  Configure Vines? [no]: y
  Configure bridging? [no]:
  Configure MOP? [no]:
```

```
Configuring interface parameters:

Configuring interface TokenRing0:
  Is this interface in use? [yes]: no

Configuring interface TokenRing1:
  Is this interface in use? [yes]:
  Configure IP on this interface? [yes]:
    IP address for this interface: 131.108.81.2
    Number of bits in subnet field [0]: 8
    Class B network is 131.108.0.0, 8 subnet bits; mask is 255.255.255.0
  Configure DECnet on this interface? [yes]:
    DECnet cost [10]:
  Configure XNS on this interface? [yes]:
    XNS network number [2]:
  Configure Novell on this interface? [yes]:
    Novell network number [2]:
  Configure AppleTalk on this interface? [yes]:
    AppleTalk network number [2]:
    appletalk zone name [myzone]:
  Configure CLNS on this interface? [yes]: n
  Configure Vines on this interface? [yes]:

Configuring interface TokenRing2:
  Is this interface in use? [yes]:
  Configure IP on this interface? [yes]:
    IP address for this interface: 131.108.82.2
    Number of bits in subnet field [8]:
    Class B network is 131.108.0.0, 8 subnet bits; mask is 255.255.255.0
  Configure DECnet on this interface? [yes]: n
  Configure XNS on this interface? [yes]: n
  Configure Novell on this interface? [yes]: n
  Configure AppleTalk on this interface? [yes]: n
  Configure CLNS on this interface? [yes]: n
  Configure Vines on this interface? [yes]: n

Configuring interface Ethernet0:
  Is this interface in use? [yes]:
  Configure IP on this interface? [yes]:
    IP address for this interface: 131.108.17.2
    Number of bits in subnet field [8]:
    Class B network is 131.108.0.0, 8 subnet bits; mask is 255.255.255.0
  Configure DECnet on this interface? [yes]: n
  Configure XNS on this interface? [yes]: n
  Configure Novell on this interface? [yes]: n
  Configure AppleTalk on this interface? [yes]: n
  Configure CLNS on this interface? [yes]: n
  Configure Vines on this interface? [yes]: n

Configuring interface Serial0:
  Is this interface in use? [yes]: n

Configuring interface Ethernet1:
  Is this interface in use? [yes]:
  Configure IP on this interface? [yes]:
    IP address for this interface: 131.108.80.2
    Number of bits in subnet field [8]:
    Class B network is 131.108.0.0, 8 subnet bits; mask is 255.255.255.0
```

```
  Configure DECnet on this interface? [yes]:
    DECnet cost [10]:
  Configure XNS on this interface? [yes]:
    XNS network number [5]:
  Configure Novell on this interface? [yes]:
    Novell network number [5]:
  Configure AppleTalk on this interface? [yes]:
    AppleTalk network number [5]:
    appletalk zone name [myzone]:
  Configure CLNS on this interface? [yes]: n
  Configure Vines on this interface? [yes]: n

Configuring interface Serial1:
  Is this interface in use? [yes]: n

The following configuration command script was created:

hostname SandBox
enable password shovel
line vty 0 4
password hammer
snmp-server community
!
ip routing
decnet routing 55.87
decnet node-type area
xns routing
novell routing
appletalk routing
clns routing
clns router igrp area_1 net  49.0001.0000.0C00.0A83.00
vines routing
no bridge 1
!
! Turn off all XNS and Novell to prevent network conflicts.
interface TokenRing0
no xns network
no novell network
 --More--
interface TokenRing1
no xns network
no novell network
interface TokenRing2
no xns network
no novell network
interface Ethernet0
no xns network
no novell network
interface Serial0
no xns network
no novell network
interface Ethernet1
no xns network
no novell network
interface Serial1
no xns network
no novell network
!
```

```
interface TokenRing0
shutdown
no ip address
 --More--
!
interface TokenRing1
ip address 131.108.81.2 255.255.255.0
decnet cost 10
xns network 2
novell network 2
appletalk cable-range 2-2
appletalk zone myzone
vines metric
!
interface TokenRing2
ip address 131.108.82.2 255.255.255.0
!
interface Ethernet0
ip address 131.108.17.2 255.255.255.0
!
interface Serial0
shutdown
no ip address
!
 --More--
interface Ethernet1
ip address 131.108.80.2 255.255.255.0
decnet cost 10
xns network 5
novell network 5
appletalk cable-range 5-5
appletalk zone myzone
!
interface Serial1
shutdown
no ip address
!
router igrp 109
network 131.108.0.0
!
end

Use this configuration? [yes/no]: yes
[OK]
Use the enabled mode 'configure' command to modify this
configuration.




Press RETURN to get started!
```

The server displays the system name (SandBox), followed by an angle bracket (>), which is the prompt of the system's command interpreter.

Using the EXEC Command Interpreter

The command interpreter is called the EXEC. The EXEC interprets the commands you type and carries out the corresponding operations.

You can type commands when you see the system prompt, which is the system's host name ending with an angle bracket (>). Although the default system host name is *Gateway*>, this may have been changed during the initial configuration using the **setup** command, or with the **hostname** configuration command. The following sections describe how to use the EXEC.

Command Syntax

The EXEC accepts commands typed in uppercase letters, lowercase letters, or both. You may also abbreviate commands and other keywords to the number of characters that cause the command to be a unique abbreviation. For example, you can abbreviate the **show** command to **sh**.

If you make a typing mistake, you can erase characters one at a time with the Delete or the Backspace key. Press either key to erase the last character typed. To erase the entire line, type Ctrl-U. (This notation means "Hold down the Ctrl key and press the U key.") The server acts on most commands after you press the Return key.

You can list available EXEC commands by typing a question mark (?). You can also enter a question mark to obtain more information about commands. For example, type **terminal ?** to obtain a list of **terminal** commands or **show ?** to obtain a list of **show** commands.

Certain EXEC commands produce multiple screens of output. At the end of each screen, the EXEC pauses and displays:

```
-More-
```

Type a space to continue the output; type anything else to return to the system command prompt.

EXEC Command Levels

For security purposes, the EXEC has two levels of access: user and privileged. The commands available at the user level are a subset of the commands available at the privileged level. Because many of the privileged commands set operating parameters, the privileged level should be password-protected to prevent its unauthorized use. The system prompt for the privileged level ends with a pound sign (#) instead of an angle bracket (>).

The EXEC **enable** command allows access to the privileged level, prompting for a password if one has been set with the **enable-password** configuration command. (For more information, see the section "Establishing Passwords and System Security" in Chapter 4.)

Type the **?** (question mark) command at the user level to see a list of the user-level EXEC commands similar to the following:

```
Gateway>?

connect <host>  Connect to host - same as typing just a host name
disconnect <cn> Break the connection specified by name or number
exit, quit      Exit from the EXEC
name-connection Give a connection a logical name
resume          Make the named connection be current
show <cmd>      Information commands, type "show ?" for list
systat          Show terminal lines and users
telnet <host>   Connect to host using telnet protocol
terminal        Change terminal's parameters, type "terminal ?"
where           Show open connections
<cr>            To resume connection
```

Type **enable** and enter the password to access the privileged command level. Type the **?** (question mark) command to see a list of privileged-level EXEC commands similar to this example:

```
Gateway#?

clear           Reinitialization functions, type "clear ?" for list
configure       Configure from terminal or over network
connect <host>  Connect to host - same as typing just a host name
debug           Enable debugging functions, type "debug ?" for list
disable         Turn off privileged commands
disconnect <cn> Break the connection specified by name or number
enable          Turn on privileged commands
exit, quit      Exit from the EXEC
name-connection Give a connection a logical name
ping            Send echo messages
reload          Halt and reload system
resume          Make the named connection be current
send <line>|*   Send message to a terminal line or lines
setup           Initialize system configuration
show <cmd>      Information commands, type "show ?" for list
systat          Show terminal lines and users
telnet <host>   Connect to host using telnet protocol
terminal        Change terminal's parameters, type "terminal ?"
test            Run hardware tests, type "test ?"
trace <address> Trace route to <address>
undebug         Disable debugging functions, type "undebug ?" for list
where           Show open connections
write           Write configuration memory, type "write ?" for list
<cr>            To resume connection
```

To return to the user level prompt, type **disable** at the EXEC prompt.

The EXEC command **configure** begins the configuration mode, where you enter the commands to configure your network server for its particular routing or bridging function. The following section describes the use of this command.

Entering Configuration Mode

Use the privileged EXEC command **configure** to begin configuration of the network server.

Begin by entering the privileged level of the EXEC. This is done by entering the **enable** command at the EXEC prompt:

```
Gateway>enable
```

The EXEC then prompts you for privileged level password:

```
Password:
```

Type in the password. For security purposes, the password will not be displayed. (Also note that the password is case sensitive.) When you enter the correct password, the system displays the privileged mode system prompt:

```
Gateway#
```

To begin configuration mode, enter the **configure** command at the privileged mode prompt:

```
Gateway#configure
```

When you enter this command, the EXEC prompts you for the source of the configuration subcommands.

```
Configuring from terminal, memory, or network [terminal]?
```

The default is to type in commands from the terminal console. Pressing the Return key begins this configuration method. Each configuration technique—(terminal, memory, and network)—is described in more detail later in this chapter.

The EXEC provides you with a simple editor for entering the configuration commands, and explains the editing functions:

```
Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U;end with CTRL/Z
```

Table 2-1 lists the edit key functions and their meanings.

Table 2-1 Configuration Edit Keys

Key	Meaning
Delete or Backspace	Erases one character.
Ctrl-W	Erases a word.
Ctrl-U	Erases a line.
Ctrl-R	Redisplays a line.
Return	Executes single-line commands.
Ctrl-Z	Ends configuration mode and returns to the EXEC.

Entering the Configuration Commands

The configuration subcommands are categorized by these functions:

- Global configuration commands—Define system-wide parameters.
- Interface subcommands—Define the characteristics of an interface (a serial or Ethernet interface, for example) and must be preceded by an **interface** command.
- Line subcommands—Define the characteristics of a serial terminal line and must be preceded by a **line** command.
- Router subcommands—Configure an IP routing protocol and must be preceded by a **router** command.

The descriptions of the commands include the command type and give examples of their use.

As with EXEC commands, you can type configuration subcommands in uppercase letters, lowercase letters, or both. You may also shorten all commands and other keywords to unique abbreviations. You may add comments by preceding the line with an exclamation point (!). Comments do not affect command processing.

If you make a typing mistake, use the Delete or Backspace key to erase a character, Ctrl-W to erase a word, and Ctrl-U to erase a line. To redisplay a line, use Ctrl-R. See Table 2-1 for a list of valid commands.

The network server executes single-line commands when you press the Return key. The network server does not display confirmation messages as it executes the commands. If the network server encounters a problem, it displays an error message on the console terminal. When you type Ctrl-Z, the network server exits the configuration mode.

In most cases, you can negate a configuration subcommand or restore a default by typing **no** before the subcommand keyword. You can usually omit the arguments of the subcommand when you negate it with **no.** The command descriptions note any exceptions to these rules.

Examples of Configuration Files

Following are some examples of configuration files to illustrate how to enter the configuration commands.

Global Configuration Commands

Use global configuration commands to enable functions that affect the system rather than a particular line or interface, and can appear any place within the configuration file. An example of this is the global configuration command to define the host name, or the name of the router:

```
hostname router-1
```

Commands to enable a particular routing or bridging function are also global configuration commands. The following example illustrates how to enable the Xerox Network System routing protocol:

```
xns routing 0123.4567.abcd
```

Once enabled, interface characteristics for XNS routing are specified using the **interface** command and XNS-specific interface subcommands. Command descriptions in the sections describing configuration will define the command type.

Interface Subcommands

Interface subcommands modify the operation of an interface such as an Ethernet, FDDI, or serial port. Interface subcommands always follow an **interface** command which defines the interface type.

The following example illustrates how to enable XNS network 1 on interface Ethernet 0:

```
interface ethernet 0
xns network 1
```

The following example illustrates how to configure the token rotation timer on interface FDDI 0:

```
interface FDDI 0
fddi token-rotation-timer 24000
```

The EXEC accepts commands in uppercase and lowercase letters. Exclamation points are not parsed and serve as comment lines and delimiters between configuration commands.

If you forget to enter the **interface** command, the system displays the message "must specify a network interface."

Line Subcommands

Line subcommands modify the operation of a serial terminal line. Line subcommands always follow a **line** command which defines the line number. If you forget to enter the **line** command, the system displays the message "must specify a line or range of lines."

The following example illustrates how to set the password on line 5:

```
line 5
password secretword
```

Router Subcommands

Router subcommands are used to configure IP routing protocol characteristics and always follow a **router** command. The following example illustrates how to set the maximum hop metric for the Cisco IGRP routing protocol:

```
router igrp
metric maximum-hops 150
```

If you forget to enter the **router** command, the system displays the message "must specify a routing protocol."

Remember to type Ctrl-Z to end your configuration sessions, and to use the **disable** command to leave privileged level mode.

Creating the Configuration File

If you used the **setup** facility's interactive dialog prompts to start your configuration file, it was saved in nonvolatile memory when you finished the prompts. If you chose not to create your configuration file this way, there are several options you may now choose from to create the configuration file.

The network server holds configuration information in two places—in *running memory*, and in *nonvolatile memory*. Configuration information in running memory is temporary and will not be stored if power is shut off. Configuration information in nonvolatile memory is always available.

You use the EXEC command **write memory** to copy current (running) configuration information to nonvolatile memory. This command stores all nondefault configuration information as configuration commands in text format. The command also records a checksum for the information to protect against data corruption.

The EXEC command **show configuration** displays information stored in nonvolatile memory. You can use this command and the **write terminal** command to find differences between the current configuration (that in running memory) and that stored in nonvolatile memory. You use the EXEC command **write erase** to clear the contents of nonvolatile memory.

The **write** commands create their output by examining the state of the system currently running. The output produced by the **write** commands is generated by the software, and will not necessarily match the text the user entered to create the current configuration.

The network server also allows you to store the configuration file on a network host. (This allows you to use an editor on the host to edit and create the configuration file.) Use the EXEC command **write network** to copy the current configuration information to a server host on the network. Use of this command is described later in this section.

Configuring from the Console

To issue configuration commands from the console terminal, enter the EXEC command **configure** at the privileged-level EXEC prompt and enter configuration mode.

The network server responds with this prompt asking you to specify the terminal, a file, or nonvolatile memory as the source of configuration commands.

```
Configuring from terminal, memory, or network [terminal]?
```

To begin configuration, type **terminal** at the prompt or just press Return (since terminal is the default) to start command collection. (See the section "Entering Configuration Mode" in this chapter for more information.)

During command collection, the network server accepts one configuration command per line. You can enter as many configuration subcommands as you want.

Type Ctrl-Z when you finish entering configuration commands. This returns you to the EXEC where you can test your configuration, or write the configuration commands to memory.

At periodic intervals, you will want to write the configuration information into nonvolatile memory or to a configuration file stored on a remote host. This will make checking, adding information to, and booting the configuration file an easier task. The procedures for writing information to nonvolatile memory are described next.

Writing the Configuration File to Nonvolatile Memory

After you enter the desired configuration information at the console terminal, use the privileged EXEC command **write memory** to make a copy of the configuration information in the nonvolatile memory. Nonvolatile memory stores the current configuration information in text format as configuration commands, recording only nondefault settings. The memory is checksummed to guard against corrupted data.

As part of its start-up sequence, the network server startup software always checks for configuration information in the nonvolatile memory. Once the nonvolatile memory holds valid configuration commands, the network server executes the commands automatically at startup. If the network server detects a problem with the nonvolatile memory or the configuration information it contains, the network server may enter the setup mode, prompting for configuration information. Problems can include a bad checksum for the information in the nonvolatile memory and the absence of critical information.

To display the configuration information stored in the nonvolatile memory, enter the **show configuration** EXEC command at the privileged mode EXEC prompt.

To clear the contents of the nonvolatile memory, enter the **write erase** EXEC command at the privileged level EXEC prompt.

To re-execute the configuration commands stored in nonvolatile memory, enter **memory** at the configure mode prompt:

```
Configuring from terminal, memory, or network [terminal]?memory
```

Writing the Configuration File to a Remote Host

To store configuration information on a remote host, enter the privileged EXEC command **write network**. This command sends a copy of the current configuration information to a remote host. The command will prompt you for the destination host's address and a file name, as the following example illustrates.

Example:

```
Tokyo#write network
Remote host [131.108.2.155]?
Name of configuration file to write [tokyo-confg]?
Write file tokyo-confg on host 131.108.2.155? [confirm]y
Writing tokyo-confg...
 [OK]
```

To retrieve and/or add to the configuration information stored on a host file on a device on your network, enter **network** at the configure mode prompt (see the section "Entering Configuration Mode" in this chapter for more information):

```
Configuring from terminal, memory, or network [terminal]?network
```

The system will ask you to select a host or network configuration file, for the address of the host, and for a file name. The following example illustrates this process.

Example:

```
Host or network configuration file [host]?
IP address of remote host [255.255.255.255]? 131.108.2.155
Name of configuration file [tokyo-confg]?
Configure using tokyo-confg from 131.108.2.155? [confirm]y
Booting tokyo-confg from 131.108.2.155: !! [OK - 874/16000 bytes]
```

Setting Up Auto Load of the Configuration File

The network server may be configured to automatically load additional configuration information from a network host. You may want to keep an up-to-date version of configuration information on another host, where you can change it as necessary, and use the nonvolatile memory as a bootstrap or backup mechanism. You can instruct the network server to load configuration information over the network by entering the **service config** subcommand and then writing the information to nonvolatile memory using the **write memory** command. Loading configuration information over the network is the default if nonvolatile memory is not installed. (The **service** configuration subcommand is described in the section "Tailoring Use of Network Services" in Chapter 4.)

After loading configuration information from the nonvolatile memory, the network server will attempt to load two configuration files from remote hosts. The first is the network configuration file, which contains commands that apply to all network servers and terminal servers on a network. The second is the host configuration file, which contains commands that apply to one network server in particular.

The default name of the network configuration file is *network-confg*. The default name for the host configuration file is taken from the host name. The host name can be specified by the **hostname** configuration subcommand or can be derived from the Domain Name System (DNS); see the section "Setting the Host Name" in Chapter 4 for more information. To form the host configuration file name, the network server converts the host name to lower case, stripped of any DNS information, and appends "-confg." If no host name information is available, the default host configuration file name is *gateway-confg*. Other names for these configuration files can be set using the **boot** command, which is described in the section "Setting Configuration File Specifications" in Chapter 4.

The network server uses TFTP to load and save configuration files. By default, the network server uses an Internet address of all ones to broadcast TFTP Read Request messages. However, many hosts use an old style of broadcast address consisting of all zeros. You can change operation to accommodate hosts using the old style of broadcast address.

Note: TFTP is the IP Trivial File Transfer Protocol, defined in RFC 783. The details of setting up a TFTP server process and installing the configuration files on the server host vary from one operating system to another; see the documentation for your host computer if you need more information about TFTP support.

If the network server fails to load a configuration file during startup, it tries again every ten minutes (default setting) until a host provides the requested files. With each failed attempt, the network server displays a message on the console terminal.

Example:

If the network server is unable to load the file named `network-confg`, it displays the message.

```
Booting network-confg... [timed out]
```

To end these file load attempts, enter the following configuration command at the console terminal and save it in the nonvolatile memory:

```
no service config
```

This command prevents the network server from trying to access nonexistent TFTP servers when it is booted.

Note: Be aware that the system treats network and host configuration files differently when loading new parameters. When a host configuration file is loaded, all terminal line parameters are cleared before setting any new parameters. When a network configuration file is loaded, no old parameters are cleared. This means that terminal line parameters set by the network configuration file, which are generally loaded first, will be reset by the host configuration file, which is generally loaded second.

Loading Software Over the Network

As configured at the factory, the operating system software executes instructions in the onboard EPROM. You need not change the system EPROMs with each software update. Instead, you can download the latest software over the network. This process is called *netbooting.*

Netbooting works as follows: when you power on your Cisco network server product for the first time, it checks the processor configuration register or the nonvolatile memory for special netbooting instructions. If the system finds no special instructions, it executes the default EPROM software.

If the system finds netbooting instructions, it determines its interface address and then runs a special process to TFTP-load the new software into memory.

You can specify boot loading in two ways. The first way involves setting the low four bits of the processor configuration register; see the Cisco publication *Modular Products Hardware Installation and Reference*, or *IGS Hardware Installation and Reference* for details. If no bits are set, you must manually boot the system using the System Bootstrap program. If only the low bit is set, the system runs the default software. The system interprets any other binary bit combination as an octal number for use in forming the boot file name. The system forms the boot file name by starting with the word "cisco" and then appending the octal number, a hyphen, and the processor type name. The System Bootstrap program displays the processor type name at system start up.

For example, if bit one in the four-bit field is set and the processor type is CSC/3, the boot file name formed is *cisco3-csc3*. Assuming no other information is available, the system would try to TFTP-load the file *cisco3-csc3* by first sending a broadcast TFTP read request to determine which server host had the file.

The second way to specify netbooting uses the nonvolatile memory option, which enables you to provide more detailed instructions for software downloading. You can use the **boot** configuration command to specify both the boot file name and the IP address of the server host. You must, however, still set the bottom four bits of the configuration register to a netbooting value.

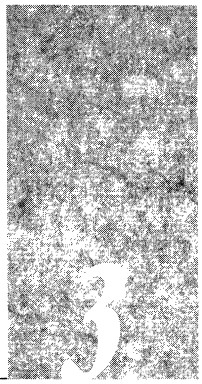
For the CSC/2 network server images, it is necessary to use the secondary bootstrap process for netbooting due to memory limitations and the expansion of system software. The secondary bootstrap process requires that there be an image called *boot-csc2* on the TFTP server (contact Cisco Customer Service for this). Also, bit 8 of the processor configuration register must be set to enable using this process. When booting, the system will request and load *boot-csc2*. This bootstrap image will then request and load the specified system image.

Reloading the Operating System

Use the following EXEC command to halt and restart the network server:

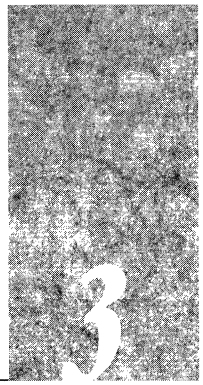
reload

If the system is set to restart on error, it reboots itself.



Chapter 3
Using Terminals

Chapter 3
Using Terminals



This chapter describes use of physical and virtual terminals on the Cisco router/bridge product. These tasks include:

- Making Telnet server connections from a console attached to the router/bridge
- Using the DEC MOP terminal server
- Making local changes to the terminal parameters

This chapter concludes with alphabetical summaries of the commands described in this chapter.

Making and Managing Terminal Connections

A TCP/IP Telnet connection is the basic way to communicate from a terminal to a host on a network. The Cisco Systems network servers provide Telnet communication as defined in RFC 854 and the MIL STD 1782 specification.

Making Telnet Connections

To start a Telnet connection, type a host name or a dotted-decimal Internet address at the EXEC prompt. You may precede the host name or Internet address with the command **connect** or **telnet**. This can be helpful if the host name you want to use conflicts with a network server command name.

The network server automatically numbers connections for you. Several commands use these numbers to identify connections, and you can display them using the **where** command described later in this chapter.

If you use a host name, the network server must first find the corresponding Internet address. To find this address, the network server searches its host-name-to-address cache. If the name is not in the cache, the network server uses a dynamic name lookup method. This method enables the network server to query a set of server hosts for the address.

As an option, you can specify a decimal TCP port number after the host name or Internet address when starting a Telnet connection. Normally, the network server uses the default Telnet server port, port number 23 (decimal).

After the network server determines the Internet address, or if you specify the address directly, the network server attempts to connect with the Telnet server port at that address. If the connection attempt fails, the network server displays a message to that effect and returns to the EXEC interpreter.

If the connection attempt succeeds, you can communicate with the server host as a terminal of that host. When you log off the host, the network server returns to the EXEC interpreter.

Example:

To connect to a host named *router-1*, you would simply type that name at the prompt, as seen in this example.

```
Gateway>router-1
```

This example illustrates how to connect to a router with IP address *103.81.25.2*:

```
Gateway>connect 103.81.25.2
```

Establishing Multiple Connections

The network server provides an escape sequence with which you can leave a Telnet connection without terminating it and return to the EXEC interpreter. This allows you to have any number of concurrent Telnet connections open, and to switch back and forth between them. Follow these steps to switch between connections:

Step 1: Type the escape sequence, which is usually the default key sequence Ctrl-^, X. This sequence is entered by pressing the Ctrl and ^ keys simultaneously, letting go, then pressing the X key.

Step 2: At the system command prompt, type the command to open another connection.

To make a new connection, use the procedure described in the previous section, "Making Telnet Connections." To return to an existing connection, use the **resume** command. Use the **where** command to show your open connections.

You can change the first part of the escape sequence with the **escape-character** command; see the section "Setting the Escape Character" in Chapter 4.

Listing Connections

Use the following command to get a listing of connections:

where

This command displays information about open connections associated with the current terminal line, as seen in the following example:

```
Gateway>where

Conn Host              Address           Byte  Idle Conn Name
   1 DREGGS            130.106.19.50        0     0 DREGGS
   2 EMBER             130.106.20.33        0     0 EMBER
*  3 CLASH             130.106.21.24        0     0 CLASH
```

The information includes the connection number, host name, address, number of characters waiting to be sent to the terminal, idle time, and connection name. An asterisk (*) indicates the current connection.

Resuming a Previous Connection

Use the EXEC **resume** command to resume a connection. This command has the following syntax:

resume [*connection*]

This command provides three ways to resume a previous connection:

- Typing the **resume** command with a connection number
- Typing only the connection number
- Pressing the Return key to return to the most recent connection.

The **where** command provides the connection number.

The following examples demonstrate use of the **resume** command.

Examples:

This command resumes connection 2.

```
Gateway>resume 2
```

You can omit the command name and simply type the connection number to resume that connection. This example resumes connection 3.

```
Gateway>3
```

To resume the most recent connection, simply press the Return key.

Naming a Connection

To name a connection, use the following command:

name-connection

This command assigns a logical name to a connection. The EXEC prompts for the connection number and name to assign when you enter this command. The **where** command displays a list of the assigned logical connection names.

Exiting a Session

To exit a session, use one of the following commands:

exit

quit

The **exit** and **quit** commands terminate the incoming connection and all outgoing connections from the network server. Enter one of these commands when you are finished with all sessions.

Disconnecting

To disconnect from a specified connection, use the following command:

disconnect [*connection*]

The optional argument *connection* is a connection name or number; the default is the current connection.

Do not use the **disconnect** command to end a session. Instead, log off the host, thus allowing the host to initiate the disconnect. If you cannot log off the host, then use the **disconnect** command.

Resetting a Line

To reset a terminal line, use the following privileged EXEC command:

clear line *line-number*

This command aborts any connections, terminates the associated processes, and resets the data structures associated with a terminal line.

The argument *line-number* specifies the terminal line number.

Incoming Telnet Connections

In addition to the console terminal, each network server supports up to five incoming Telnet connections. Each of these connections can start an EXEC interpreter process on the network server.

The user of an incoming Telnet connection can gain access to the privileged EXEC commands through the **enable** command, which requires a password. With access to the complete EXEC command set, the incoming connection acts as a remote console. A remote console connection provides a convenient way to monitor and adjust network server operation.

You can control access to the network server with access lists; see the section "Configuring IP Access Lists" in Chapter 13, "Routing IP," for more information.

The network server supports the following Telnet options:

- Echo
- Binary Transmission
- Suppress Go Ahead
- Terminal Type
- Send Location

Displaying TCP Connections

To show the status of a TCP connection, enter this EXEC command:

show tcp [*line-number*]

The **show tcp** command displays the status of all TCP connections. Specify the optional argument *line-number* in octal to display the status of the TCP connections for a particular line. The following example shows the command output:

```
con0 (console terminal), connection 1 to host MATHOM
Connection state is ESTAB, I/O status: 1, unread input bytes: 1
Local host: 192.31.7.18, 33537  Foreign host: 192.31.7.17, 23
Enqueued packets for retransmit: 0, input: 0, saved: 0
Event Timers (current time is 2043535532):
Timer:       Retrans   TimeWait    AckHold    SendWnd   KeepAlive
Starts:           69          0         69          0           0
Wakeups:           5          0          1          0           0
Next:     2043536089          0          0          0           0
iss: 2043207208 snduna: 2043211083  sndnxt: 2043211483    sndwnd: 1344
irs: 3447586816 rcvnxt: 3447586900  rcvwnd:       2144 delrcvwnd:   83
RTTO: 565 ms, RTV: 233 ms, KRTT: 0 ms, minRTT: 68 ms, maxRTT: 1900 ms
ACK hold: 282 ms
Datagrams (max data segment is 536 bytes):
Rcvd: 106 (out of order: 0), with data: 71, total data bytes: 83
Sent: 96 (retransmit: 5), with data: 92, total data bytes: 4678
```

Displaying Active Sessions

To show the current active sessions, use the following command:

show sessions

The **show sessions** command provides information about open Telnet connections. This command may be run at the user-level prompt.

Displaying Information About Active Lines

The **show users** and **systat** EXEC commands display information about the active lines of the network server, including the line number, connection names, and terminal location.

show users [**all**]

systat [**all**]

Specify the optional keyword **all** to display information for both active and inactive lines. These commands enable monitoring of virtual terminal use. You may issue these commands at the user-level prompt. They are synonymous.

Changing Terminal Parameters

The following sections describe how to change the terminal parameters using the **terminal** commands. The new settings temporarily override those made with the line configuration subcommands described in the section "Configuring Console and Virtual Terminal Lines" in Chapter 4.

To obtain information about the terminal configuration parameter settings for the current terminal line, use the **show terminal** command. To display information about the active ports of the server, use the **show users** command. The information displayed includes the line number, connection name, idle time, and terminal location.

Some **terminal** commands use the decimal representation of an ASCII character as an argument. See Appendix E, "ASCII Character Set," for ASCII-to-decimal conversion information.

To display a list of commands that you can enter to change the hardware and software parameters of the current terminal line, use the command:

terminal ?

Each command has a no variation that undoes the local setting.

Changing the Terminal Screen Width

To set or unset the number of characters (columns) on a single line of the current terminal screen, use the **terminal width** command:

terminal width *columns*

terminal no width

The login protocol uses the argument *columns* to set up terminal parameters on a remote host.

Example:

This example sets the terminal width to 132 columns.

```
Gateway>terminal width 132
```

Changing the Terminal Escape Character

To set or unset the escape character for the current terminal line, use the **terminal escape-character** command:

terminal escape-character *decimal-number*

terminal no escape-character

The argument *decimal-number* is the ASCII decimal representation of the desired escape character or an escape character (Ctrl-P, for example). Typing the escape character followed by the X key returns you to the EXEC when you are connected to another computer. The default escape character is Ctrl-^. (See Appendix E at the end of Volume II for a list of ASCII characters.)

The operating software interprets by pressing the Break key on the console as an attempt to halt the system.

Note: Depending upon the configuration register setting, console breaks will either be ignored or cause the server to shut down. The Break key cannot be used as the escape character on the Cisco router.

Example:

This example sets Ctrl-P as the escape characters:

```
Gateway>terminal escape-character 17
```

Displaying the Debug Messages on the Console and Terminals

To display the debug message on the console and terminals, use the **terminal monitor** command:

terminal monitor

terminal no monitor

The **terminal monitor** command copies **debug** command output and system error messages to the current terminal as well as to the console terminal.

To use this command, you must first issue the **enable** command and enter the password to access the privileged command mode.

Changing the Character Padding

To set the character padding on the current terminal line, use the **terminal padding** commands:

terminal padding *decimal-number count*

terminal no padding *decimal-number*

The argument *decimal-number* is the ASCII decimal representation of the character. The argument *count* is the number of NULL bytes sent after that character. (Appendix E at the end of Volume II provides a list of the ASCII characters.)

The **terminal no padding** command ends this padding.

Example:

This example pads RETURNS (ASCII character 25) with 20 NULL bytes:

```
Gateway>terminal padding 25 20
```

Displaying Terminal Parameter Settings

To display the configuration parameter settings for the current terminal, use this EXEC command:

show terminal

This command may be issued at the user-level prompt.

Using the DEC MOP Server

All Cisco internetworking products include a server which implements a subset of Digital Equipment Corporation's (DEC's) Maintenance Operation Protocol (MOP) for Ethernet interfaces. The MOP server supports the request ID message, periodic system ID messages, and the remote console carrier functions.

The MOP server periodically multicasts a system ID message, which is used by Digital's Ethernet configurator to determine what stations are present in an Ethernet network. The configurator is controlled by the Network Control Program (NCP) command **define module configurator**. For more information on this command, consult DECnet-VAX documentation.

The Cisco internetworking products use the MOP communication device code of 121. This code has been assigned to Cisco by Digital, although some versions of DECnet-VAX software may report the code numerically, rather than with a device name. The Digital Ethernim configurator product also makes use of receipt of system ID messages when building network maps.

The MOP server supports the Digital remote console function. In this capacity, a system manager on a DECnet system that does not include TCP/IP can create a virtual terminal connection to a Cisco router. The NCP commands **connect node** and **connect via** are used to connect to the remote console. Due to the nature of the MOP server, only a single inbound connection per Ethernet interface is supported. The MOP server does not contain the necessary mechanisms for supporting more than one connection at a time.

MOP is not a routable protocol. To bridge the MOP console carrier and system ID functions, enable bridging for protocol type 6002. The periodic system ID messages are sent to the multicast address *AB00.0002.0000.*

The EXEC command **debug mop** reports events occurring within the MOP server, including reception of request ID messages, transmission of system ID messages, and reservation and release of the remote console.

EXEC Terminal Commands Summary

This section lists the EXEC commands described in this chapter in alphabetical order.

clear line *line-number*

Aborts connections and processes and resets the line. Argument *line-number* specifies the line.

{**connect** | **telnet**} [*connection*]

Either of the commands connects to a remote host using the Telnet protocol. The optional argument *connection* specifies a host name or an IP address.

disconnect [*connection*]

Closes the specified connection. The optional argument *connection* is a connection name or number; the default is the current connection.

exit
quit

Either of the commands terminates the EXEC command processor and closes any active Telnet sessions.

name-connection

Assigns a logical name to a connection. The EXEC prompts for the connection number and name to assign when you enter this command.

resume [*connection*]

Resumes a connection. The optional argument *connection* is a connection name or number.

show sessions

Provides information about open Telnet connections.

show users [**all**]
systat [**all**]

Displays information about active lines. The optional **all** keyword provides information about inactive as well as active ports.

show terminal

Displays information about the terminal configuration parameter settings for the current terminal line and the active ports of the server. The optional keyword **all** requests information for both active and inactive ports.

terminal ?

Lists commands you can enter to change hardware and software parameters of the current line.

terminal [no] escape-character *decimal-number*

Sets the escape character for the current terminal line. The argument *decimal-number* is either the ASCII decimal representation of the desired escape character or an escape character. The default escape character is Ctrl-^.

terminal [no] monitor

Displays the debug message on the console and terminals. It copies **debug** command output and system error messages to the current terminal as well as to the console terminal. To use this command, you must first issue the **enable** command and enter the password to access the privileged command mode.

terminal [no] padding *decimal-number count*

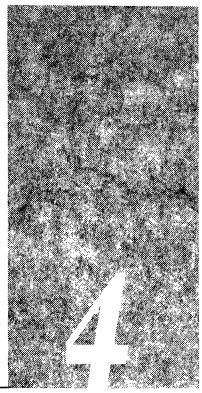
Sets the character padding on the current terminal line. The argument *decimal-number* is the ASCII decimal representation of the character. The argument *count* is the number of NULL bytes sent after that character. (Appendix E at the end of Volume II provides a list of the ASCII characters.) The **terminal no padding** command ends this padding.

terminal [no] width *columns*

Sets the number of characters (columns) on a single line of the current terminal screen. The login protocol uses the argument *columns* to set up terminal parameters on a remote host.

where

Displays information about open connections associated with the current terminal line and provides the connection number.



Chapter 4
Configuring the System

Chapter 4
Configuring the System

This chapter describes how to configure the system. These tasks include:

- Setting global system characteristics such as the host name and console banner message
- Defining the size of the system buffers
- Changing the system boot file specifications
- Establishing system and line passwords and system security
- Defining network services such as the IP Finger protocol
- Enabling and directing the logging of system debugging messages
- Configuring the console and virtual terminal lines

This chapter concludes with alphabetical summaries of the commands described in this chapter.

Configuring the Global System Parameters

The following sections contain procedures and command descriptions for configuring the global system characteristics: host name, and passwords, and configuring system security and system management functions. The global configuration commands described in the following sections are entered in configuration mode. See the section "Entering Configuration Mode" in Chapter 2 for the procedures to enter into this mode.

Setting the Host Name

Use the **hostname** global configuration command to specify the host name for the network server, which is used in prompts and default configuration file names.

hostname *name*

The argument *name* is the new host name for the network server and is case sensitive. The default host name is *Gateway.*

Example:

This command changes the host name to *sandbox*.

```
hostname sandbox
```

Displaying Banner Messages

A banner is the message that the EXEC command interpreter displays whenever a user starts any EXEC process or activates a line. The general form of the **banner** command follows.

banner {**motd** | **exec** | **incoming**} *c text c*

The **motd**, **exec**, and **incoming** keywords control when the banner message is displayed. The use of these keywords is described in the following sections.

The argument *c* specifies a delimiting character of your choice. The argument *text* specifies the message to be shown on the screen whenever an interface line is activated.

Follow **banner** with one or more blank spaces and then type the delimiting character then one or more lines of text *text*, terminating the message with the second occurrence of the delimiting character. There is no limit to the amount of characters that can be used for the banner, with the exception of buffer limits and what is appropriate for a banner.

Example:

The following example uses the # character as a delimiting character:

```
banner motd #
Building power will be off from 7:00 AM until 9:00 AM this coming Tuesday.
#
```

Note: You cannot use the delimiting character in the banner message.

Displaying a Message of the Day Banner

To specify a general-purpose message-of-the-day type banner, use the **banner motd** global configuration command.

banner motd *c text c*

This displays a message-of-the-day type banner whenever a line is activated, or when an incoming Telnet connection is created.

Note: The command **banner** is equivalent to the command **banner motd**, except that the banner is displayed on incoming connections.

Displaying a Banner with an EXEC Process

To be able to display a message when an EXEC process is created, use the **banner exec** global configuration command.

banner exec *c text c*

This command specifies a message to be displayed on when an EXEC process is created (line activated, or incoming connection to VTY).

Displaying a Incoming Message Banner

To display an incoming message to a particular terminal line, use the **banner incoming** global configuration command.

banner incoming *c text c*

This specifies a message to be displayed on incoming connections to particular terminal lines, (for example, lines used for "milking machine" applications).

Note: Messages are never displayed on incoming stream type connections, since they might interfere with printer daemons.

The EXEC banner can be suppressed on certain lines using the **no exec-banner** line subcommand (described in the section "Suppressing Banner Messages" later in this chapter.) Lines so configured will *not* display the EXEC or MOTD banners when an EXEC is created.

Example:

This example illustrates how to display a message-of-the-day, and a message that will be displayed when an EXEC process is created. Use the **banner** global configuration commands and **no exec-banner** line subcommand to accomplish these settings.

```
! Both messages are inappropriate for the VTYs.
line vty 0 4
no exec-banner
!
banner exec /
This is Cisco Systems training group server.
Unauthorized access prohibited.
/
!
banner motd /
The server will go down at 6pm for a software upgrade
/
```

Setting the System Buffers

In normal system operation, there are several pools of different sized buffers. These pools grow and shrink based upon demand. Some buffers are temporary and are created and destroyed as warranted. Other buffers are permanently allocated and cannot be destroyed. The **buffers** command allows a network administrator to adjust initial buffer pool settings, as well as the limits at which temporary buffers are created and destroyed. It is normally not necessary to adjust these parameters; do so only after consulting with Cisco support personnel. Improper settings could adversely impact router performance. The full syntax of this command follows:

buffers {**small** | **middle** | **big** | **large** | **huge**} {**permanent** | **max-free** | **min-free** | **initial**} *number*

no buffers {**small** | **middle** | **big** | **large** | **huge**} {**permanent** | **max-free** | **min-free** | **initial**} *number*

First choose the keyword that describes the size of buffers in the pool—small, big, huge, etc. The default number of the buffers in a pool is determined by the hardware configuration, and can be displayed with the EXEC **show buffers** command.

The following keyword specifies the buffer management parameter to be changed, and can be one of the following arguments:

- **permanent**—The number of permanent buffers that the system tries to allocate. Permanent buffers are normally not deallocated by the system.
- **max-free**—The maximum number of free or unallocated buffers in a buffer pool.
- **min-free**—The minimum number of free or unallocated buffers in a buffer pool.
- **initial**—The number of additional temporary buffers which should be allocated when the system is reloaded. This can be used to insure that the router has necessary buffers immediately after reloading in a high traffic environment.

The argument *number* specifies the number of buffers to be allocated.

The **no buffers** command with appropriate keywords and argument restores the default buffer values.

Examples:

In the following example, the system will try to keep at least 50 small buffers free.

```
buffers small min-free 50
```

In this example, the system will try to keep no more than 200 medium buffers free.

```
buffers medium max-free 200
```

With the following command, the system will try to create one large temporary extra buffer, just after a reload:

```
buffers large initial 1
```

In this example, the system will try to create one permanent huge buffer:

```
buffers huge permanent 1
```

To display statistics about the buffer pool on the network server, use the command **show buffers**. For more information, refer to the section "Monitoring System Processes" in Chapter 5.

Setting Configuration File Specifications

This section describes the **boot** global configuration commands used to configure boot files. The **boot** command can be used to perform these tasks:

- Change default file names.
- Specify a server host for netbooting configuration files and boot image files.
- Specify the size buffer to configure for netbooting a host or network configuration file.

The commands to load files over the network take effect the next time the software is reloaded, provided they have been written into nonvolatile memory.

Changing the Network Configuration File

The network configuration file contains commands that apply to all network servers and terminal servers on a network. The default name of this file is *network-confg*. See the section "Entering Configuration Mode" in Chapter 2. To change the name of this file use the **boot network** global configuration command. The full command syntax follows:

boot network *filename* [*address*]

no boot network [*filename address*]

The keyword **network** changes the network configuration file from *network-confg*. The argument *filename* is the new name for the network configuration file. If you omit the argument *address,* the network server uses the default broadcast address of *255.255.255.255*. If you use *address,* you can specify a specific network host or a subnet broadcast address.

Changing the Host Configuration File

The host configuration file contains commands that apply to one network server in particular. To change the host configuration file name, use the **boot host** global configuration command. The full command syntax follows:

boot host *filename* [*address*]

no boot host [*filename address*]

The keyword **host** changes the host configuration file name to a name you specify in the *filename* argument. The network server uses its name to form a host configuration file name. To form this name, the network server converts its name to all lowercase letters, removes all domain information, and appends "-confg." By default, the host file name is *gateway-confg*.

Obtaining the Boot File Over the Network

New versions of the software can be downloaded over the network. Use the **boot system** global configuration command to do this. The full command syntax follows.

boot system *filename* [*address*]

no boot system [*filename address*]

The keyword **system** indicates that the file name and host addresses for booting operating software over the network are in the nonvolatile memory. In this case, the argument *filename* is the file name of the operating software to load, and the argument *address* is the address of the network host holding that file.

The **boot system** command overrides the processor configuration register setting unless the register specifies the use of default (ROM) operating software. Therefore, to permit netbooting, set the configuration register bits on the processor card to any pattern other than 0-0-0-0 or 0-0-0-1.

Note: The Cisco software boots images over a network by using one system image to load another system image. This means that there must be enough room in memory for two complete system images. Some versions of the software are so large that two copies of it will not fit in memory. Therefore, the CSC/2 netboot algorithm uses a secondary bootstrap system image to netboot the desired system image. (You need Software Release 8.0 or later EPROMs to use this secondary bootstrap.)

The secondary bootstrap is a very small system image which is netloaded and invoked to netboot the desired system image. The secondary bootstrap for CSC/2 processors is named *boot-csc2*. A secondary bootstrap is *not* required for the CSC/3 processor, since it has enough memory to net boot any CSC/3 image.

If bit 9 of the configuration register is set (the factory default), use of the secondary bootstrap is enabled. Contact Cisco Customer Service for copies of the secondary bootstrap software.

Refer to the Cisco publications *Modular Products Hardware Installation and Reference* or the *IGS Hardware Installation and Reference* for more information about the processor configuration registers.

Note: The IGS requires four megabytes of RAM to netboot.

Example:

To use the nonvolatile memory option to specify netbooting, place a **boot system** command in the nonvolatile memory. You use this command to specify both the file name of the operating software to load, and the Internet address of the server host holding that file:

```
boot system /usr/local/tftpdir/cisco.ts2 192.7.31.19
```

Specifying a Boot File Buffer Size

To specify the size of the buffer to be used for netbooting a host or a network configuration file, use the **boot buffersize** global configuration command. The full command syntax follows:

boot buffersize *bytes*

no boot buffersize *bytes*

The argument *bytes* specifies the size of the buffer to be used. By default it is the size of your non-volatile memory, or 32 kilobytes if you do not have nonvolatile memory. There is no minimum or maximum size that may be specified.

The EXEC commands **write terminal** and **write network** use the information specified by the **buffersize** keyword when performing their functions (see the section "Entering Configuration Mode" in Chapter 2 for more information about these EXEC commands).

Configuring Multiple Instances of the Boot Commands

You can configure multiple instances of the **boot** commands. When issued, each command is executed in order and so can be used to begin a systematic search or to build a specific list. For example, you can issue multiple **boot** commands to build an ordered list of configuration-file-name-and-host-address pairs. The network server scans this list until it successfully loads the appropriate network or host configuration file or system boot image. In this example, the network server looks first for *fred-confg* on network 192.31.7.24 and, if it cannot load that file, then for *wilma-confg* on network 192.31.7.19:

```
boot host /usr/local/tftpdir/fred-confg 192.31.7.24
boot host /usr/local/tftpdir/wilma-confg 192.31.7.19
```

Note: This example uses fictitious file names; the syntax of these file names depends on the TFTP server you are loading the files from.

If the network server cannot find either file, a background process tries at ten-minute intervals (default) to load one or the other of the files.

You may issue multiple instances of all variations of the **boot** command, including the **no boot** forms. This feature can be useful for removing configuration files. To remove a configuration file-name and host-address pair from the list, use the **no boot** command syntax.

Establishing Passwords and System Security

This section describes how to configure password protection and terminal access security.

You may set passwords to control access to the privileged command level and to individual lines. The Terminal Access-Controller Access System (TACACS) protocol controls terminal use by means of a user-ID-and-password pair. The Defense Data Network developed TACACS to control access to its TAC terminal servers; Cisco patterned its TACACS support after the DDN application.

These system security measures may not provide the level of protection needed for some environments; individual routing protocols and bridging support may have additional security procedures. You may also need to use access lists for additional protection. For procedures for configuring access lists, refer to the sections describing configuration of a particular routing protocol or bridging support.

Establishing the Privileged-Level Password

To assign a password for the privileged command level, use the **enable password** global configuration command:

enable password *password*

The argument *password* is case sensitive and specifies the password prompted for in response to the EXEC command **enable.** The *password* argument may contain any alphanumeric characters, including spaces, up to 80 characters. The password checking is also case sensitive. The password *Secret* is different than the password *secret*, for example, and the password *two words* is an acceptable password.

Note: On systems with software Release 8.2 and earlier, the command syntax was **enable-password**.

To enter the privileged command level, type the following EXEC command and then press Return:

enable

Next, type the password for the privileged command level at the `Password:` prompt.

When you use the **enable** command at the console terminal, the EXEC will not prompt for a password if the privileged mode password is not set. This behavior allows access to the **configure** command to enter configuration command collection mode, so you can set parameters—such as the password for the privileged command level—for other lines. To restrict access to the console line, set a line password as described in the section "Establishing Line Passwords" later in this chapter.

Example:

The following example sets the password *secretword* for the privileged command level on all lines, including the console:

```
enable password secretword
```

Specifying a Password

When an EXEC is started on a line with password protection, the EXEC prompts for the password. If you enter the correct password, the EXEC prints its normal nonprivileged prompt. You may try three times to enter a password before the EXEC exits and returns the terminal to the idle state.

To specify a password, use the **password** line subcommand. The full command syntax follows:

password *text*

no password

The *text* argument may contain any alphanumeric character, including spaces, up to 80 characters. The password checking is also case sensitive. The password *Secret* is different than the password *secret*, for example, and the password *two words* is an acceptable password.

To enable checking for the password specified by the **password** command, use the line subcommand **login**:

login

Alternatively, to use the TACACS user ID and password-checking mechanism instead, use the following subcommand:

login tacacs

To disable all password checking, use the command:

no login

The server prints the message-of-the-day banner before prompting for a password, so you immediately see messages such as no trespassing notifications. By default, virtual terminals require a password. If you do not set a password for a virtual terminal, it will respond to attempted connections by displaying an error message and closing the connection. Use the **no login** subcommand to disable this behavior and allow connections without a password.

Example:

The following example sets the password *letmein* on line 5:

```
line 5
password letmein
login
```

Recovering from a Lost Password

If your network server has the nonvolatile memory option, you can lock yourself out if you enable password checking on the console terminal line and then forget the line password.

To recover from this, force the network server into factory diagnostic mode by turning off the network server, inserting a jumper in bit 15 of the processor configuration register, (or bit 7 of the processor configuration register in the IGS or CRM), and turning on the network server. Follow these steps.

Step 1: You will be asked if you want to set the manufacturers' addresses. Respond by typing "Yes." You then see the following prompt:

```
TEST-SYSTEM>
```

Step 2: Type the **enable** command to get the privileged prompt:

```
TEST-SYSTEM> enable
```

Step 3: Type the **show configuration** command to review the system configuration and find the password.

Step 4: To resume normal operation, turn off the network server, remove the jumper from bit 15 (or bit 7) of the configuration register, and turn on the network server again.

Step 5: Log in to the network server with the password that was shown in the configuration file.

The processor configuration registers are described in Appendix A, "The CPU Bootstrap Program" in the *Modular Products Hardware Installation and Reference* publication, or Appendix A, "IGS Configuration Register" in the *IGS Hardware Installation and Reference* publication.

When the network server restarts in factory diagnostic mode, it does not read the nonvolatile memory, thus avoiding the command to set a password for the console terminal. Do not change anything in the factory diagnostic mode.

Note: All debugging capabilities are turned on during diagnostic mode.

Establishing Terminal Access Control

Cisco Systems provides unsupported versions of both a standard and an extended TACACS server. The servers run on most UNIX systems available from Cisco using FTP on the *ftp.cisco.com* directory. You may use the servers to create UNIX accounting applications that monitor use of a system and user logins.

The configuration commands in the following sections tailor the behavior of the standard TACACS server.

Setting the Server Host Name

The **tacacs-server host** global configuration command specifies a TACACS host. The full syntax of this command follows.

tacacs-server host *name*

no tacacs-server host *name*

The argument *name* is the name or Internet address of the host. You can use multiple **tacacs-server host** subcommands to specify multiple hosts. The server will search for the hosts in the order you specify them. The **no tacacs-server host** global configuration command deletes the specified name or address.

Limiting Login Attempts

The **tacacs-server attempts** global configuration command controls the number of login attempts that may be made on a line set up for TACACS verification.

tacacs-server attempts *count*

no tacacs-server attempts

The argument *count* is the number of attempts. The default is three attempts.

The **no tacacs-server attempts** global configuration command restores the default.

Example:

This command changes the login attempt to just one try:

```
!
tacacs-server attempts 1
!
```

Controlling Retries

The **tacacs-server retransmit** global configuration command specifies the number of times the server will search the list of TACACS server hosts before giving up. The server will try all servers, allowing each one to time-out before increasing the retransmit count.

tacacs-server retransmit *retries*

no tacacs-server retransmit

The argument *retries* is the retransmit count. The default is two retries.

The **no tacacs-server retransmit** global configuration command restores the default.

Example:

This command specifies a retransmit counter value of five times:

```
!
tacacs-server retransmit 5
!
```

Setting the Timeout Intervals

The **tacacs-server timeout** global configuration command sets the interval the server waits for a server host to reply.

tacacs-server timeout *seconds*

no tacacs-server timeout

The argument *seconds* specifies the number of seconds. The default interval is five seconds. The **no tacacs-server timeout** global configuration command restores the default.

Example:

This command changes the interval timer to ten seconds:

```
!
tacacs-server timeout 10
!
```

Setting the Last Resort Login Feature

If, when running the TACACS server, the TACACS server does not respond, the default action is to deny the request. Use the **tacacs-server last-resort** global configuration command to change that default.

tacacs-server last-resort {**password** | **succeed**}

no tacacs-server last-resort {**password** | **succeed**}

The command causes the network server to request the privileged password as verification, or forces successful login without further input from the user, depending upon the keyword specified, as follows:

- **password**—Allows the user to access the privileged-level command mode by entering the password set by the **enable** command.
- **succeed**—Allows the user to access the privileged-level command mode without further question.

Note: The last resort login feature can be useful when it is important to be able to ensure that login can occur. An example of such a condition is when a systems administrator needs to login in order to troubleshoot TACACS servers which are down.

The **no tacacs-server last-resort** global configuration command restores the system to the default behavior.

Establishing Privileged-Level TACACS

The following variations of the **enable** command may be used to configure privileged-level command access using the TACACS protocol.

Enabling the Privileged Mode

The **enable use-tacacs** global configuration command is used for setting the TACACS protocol for determining whether a user can access the privileged command level.

enable use-tacacs

no enable use-tacacs

If you use this command, the EXEC **enable** command will ask for both a new user name and password. This is then passed to the TACACS server for authentication. If you are using the extended TACACS, it will also pass any existing UNIX user identification code to the server.

Note: When used without extended TACACS, this command allows anyone with a valid user name and password to access the privileged command level, creating a potential security problem. This is because the TACACS query resulting from entering the **enable** command is indistinguishable from an attempt to log in without extended TACACS.

Enabling the Privileged Mode Last Resort Login Feature

The **enable last-resort** global configuration command allows you to specify what happens if the TACACS servers used by the **enable** command do not respond.

enable last-resort {password | succeed}

no enable last-resort {password | succeed}

The default action is to fail. Use of the keyword changes the action, as follows:

- **password**—Allows you to enable by entering the privileged command level password.
- **succeed**—Allows you to enable without further question.

The **no enable last-resort** global configuration command restores the default.

Configuring TACACS Accounting

What follows are the configuration commands that tailor the behavior of the extended TACACS client.

Enabling Extended TACACS Mode

The **tacacs-server extended** global configuration command enables an extended TACACS mode.

tacacs-server extended

no tacacs-server extended

This mode provides information about the terminal requests for use in setting up host auditing trails and accounting files for tracking use of terminal servers and routers. Information includes responses from terminal servers and routers, and validation of user requests. An unsupported, extended TACACS server is available from Cisco Systems via anonymous FTP for UNIX users who want to create the auditing programs.

The **no tacacs-server extended** command disables this mode.

Login Notification

The **tacacs-server notify** global configuration command causes a message to be transmitted to the TACACS server, with retransmission being performed by a background process for up to five minutes. The terminal user, however, receives an immediate response allowing access to the terminal. The full syntax of this command follows.

tacacs-server notify {**connect** | **slip** | **enable** | **logout**}

no tacacs-server notify {**connect** | **slip** | **enable** | **logout**}

The keywords specify notification of the TACACS server whenever a user does one of the following:

- **connect**—User makes TCP connections.
- **slip**—User turns SLIP on or off (Terminal Server only).
- **enable**–User enters the **enable** command.
- **logout**—User logs out.

The **no tacacs-server notify** command with the appropriate keyword disables notification.

Note: When used with extended TACACS, the command **tacacs-server notify enable** allows anyone with a valid user name and password to access the privileged-level command mode.

Login Authentication

The **tacacs-server authenticate** command requires a response from the network or communications server to indicate whether the user may perform the indicated action.

tacacs-server authenticate {**connect** | **slip** | **enable**}

no tacacs-server authenticate {**connect** | **slip** | **enable**}

Actions that require a response include the following, specified as optional keywords:

- **connect**—User TCP connections
- **slip**—SLIP connections (Terminal Server only)
- **enable**—Use of **enable** command

The **no tacacs-server authenticate** command with the appropriate keyword disables the action.

Configuring the Simple Network Management Protocol (SNMP)

The Simple Network Management Protocol (SNMP) provides a way to access and set configuration and run time parameters for the network server. Cisco System's implementation of SNMP is compatible with RFCs 1155, 1156, and 1157. The Cisco Management Information Base (MIB) supports RFCs 1155 to 1213, and provides Cisco-specific variables.

A separate document, available in RFC 1212-type (MIB II) format, describes all the Cisco-specific SNMP variables in the Cisco portion of the MIB. It also describes what is required to establish minimum configuration. Contact Cisco Systems to obtain a copy of this document, which includes instructions for accessing the variables using SNMP.

Enabling and Disabling the SNMP Server

To be able to configure the SNMP server, you need to be in the configuration command collection mode. You enter this mode using the EXEC command **configure** at the EXEC prompt. See the section "Entering Configuration Mode" in Chapter 2 for a description of the procedure.

You begin SNMP operation by entering the configuration commands that define the desired operation. To disable SNMP server operations on the network server after it has been started, use the **no snmp-server** global configuration command:

no snmp-server

Defining the SNMP Server Access List

To set up an access list that determines which hosts can send requests to the network server, use the **snmp-server access-list** global configuration command. The full command syntax follows.

snmp-server access-list *list*

no snmp-server access-list *list*

This command sends all traps to the host. The network server ignores packets from hosts that the access list denies.

The argument *list* is an integer from 1 through 99 that specifies an IP access list number.

The **no snmp-server access-list** global configuration command removes the specified access list.

Example:

This command sends traps to all hosts defined by access list 21:

```
!
snmp-server access-list 21
!
```

Setting the Community String

To set up the community access string, use the **snmp-server community** global configuration command. The full command syntax follows:

snmp-server community *string* [**RO|RW**][*list*]

no snmp-server community *string* [**RO|RW**][*list*]

This command enables SNMP server operation on the network server. The argument *string* specifies a community string that acts like a password and permits access to the SNMP protocol.

By default, an SNMP community string permits read-only access (keyword **RO**); use the keyword **RW** to allow read-write access. The optional argument *list* is an integer from 1 through 99 that specifies an access list of Internet addresses that may use the community string.

The **no snmp-server community** global configuration command removes the specified community string or access list.

Example:

This command assigns the string *comaccess* to the SNMP server, allows read-only access, and specifies that addresses that match the criteria in access list 4 may use the community string. (Notice that the string is entered *without* quotes or any other parsing characters.)

```
snmp-server community comaccess RO 4
```

Establishing the Message Queue Length

To establish the message queue length for each TRAP host, use the **snmp-server queue-length** global configuration command:

snmp-server queue-length *length*

no snmp-server queue-length

This command defines the length of the message queue for each TRAP host.

The argument *length* is the number of TRAP events that can be held before the queue must be emptied; the default is 10. Once a TRAP message is successfully transmitted, software will continue to empty the queue, but never faster than at a rate of four TRAP messages per second.

The **no snmp-server queue-length** command resets the queue length to its default value of 10.

Example:

This command establishes a message queue that traps four events before it must be emptied:

```
snmp-server queue-length 4
```

Establishing Packet Filtering

To establish the packet filtering size, use the **snmp-server packet-size** global configuration command. The full command syntax follows:

snmp-server packet-size *bytes*

no snmp-server packet-size

This command allows control over the largest SNMP packet size permitted when the SNMP server is receiving a request or generating a reply.

The argument *bytes* is a byte count from 484 through 8,192. The default is 484. The **no snmp-server packet-size** command resets this default.

Example:

This command establishes a packet filtering maximum size of 1024 bytes:

```
snmp-server packet-size 1024
```

Establishing the TRAP Message Recipient

To specify the recipients of TRAP messages, use the **snmp-server host** global configuration command. The full syntax follows:

snmp-server host *address community-string* [**snmp** | **tty**]

no snmp-server host *address community-string*

This command specifies which host or hosts should receive TRAP messages. You need to issue the **snmp-server host** command once for each host acting as a TRAP recipient.

The argument *address* is the name or Internet address of the host. The argument *community-string* is the password-like community string set with the **snmp-server community** command.

The optional keywords define whether the TRAPS be included, as follows:

- **snmp**—Causes all SNMP-type TRAP messages to be sent and starts the Cisco-specific RELOAD TRAP message.
- **tty**—Causes TCP connection TRAP messages to be included.

With the **snmp-server host** command, you get all the SNMP TRAP messages about TTY events by default. The **no snmp-server host** command removes the specified host.

Note: Previous versions of the Cisco software enabled traps by default. Traps are now *disabled* by default.

Examples:

This command sends all possible SNMP TRAPS to 131.108.2.160, including TTY TRAPS.

```
snmp-server host 131.108.2.160
```

In order to turn these TRAP messages off, use the **no snmp-server host** command. For example, the following sequence of commands only sends the 7 SNMP TRAPS to 131.108.2.160, not all the others.

```
snmp-server host 131.108.2.160
no snmp-server host 131.108.2.160 tty
```

This example causes all the SNMP-type messages to be sent to the host specified by the name cisco.com. The command uses the community string *comaccess* as the password:

```
snmp-server host cisco.com comaccess snmp
```

Establishing TRAP Message Authentication

To establish the TRAP message authentication, use the **snmp-server trap-authentication** global configuration command:

snmp-server trap-authentication

no snmp-server trap-authentication

This command enables the network server to send a TRAP message when it receives a packet with an incorrect community string.

The SNMP specification requires that a TRAP message be generated for each packet with an incorrect community string. However, because this action can result in a security breach, the network server by default does not return a TRAP message when it receives an incorrect community string.

Establishing the TRAP Message Timeout

To define how often to try resending TRAP messages on the retransmission queue, use these global configuration commands:

snmp-server trap-timeout *seconds*

no snmp-server trap-timeout

The argument *seconds* sets the interval for resending the messages. The default is set to 30 seconds. The **no snmp-server trap-timeout** command restores this default.

Example:

This command sets an interval of 20 seconds to try resending TRAP messages on the retransmission queue:

```
snmp-server trap-timeout 20
```

Enabling SNMP System Shutdown Feature

Using SNMP packets, a network management tool can send messages to users on virtual terminals and the network server's console. This facility operates in a similar fashion to the SNMP **send** command; however, the SNMP request that causes the message to be issued to the users, also specifies the action to be taken after the message is delivered. One possible action is a shutdown request.

Requesting a *shutdown-after-message* is similar to issuing a **send** command followed by a **reload** command. Because the ability to cause a reload from the network is a powerful feature, it is protected by this configuration command. To use this SNMP message reload feature the device configuration must include the **snmp-server system-shutdown** global configuration command. The full command syntax follows:

snmp-server system-shutdown

no snmp-server system-shutdown

The **no snmp-server system-shutdown** option prevents a SNMP system-shutdown request (from an SNMP manager) from resetting the Cisco agent.

To understand how to use this feature with SNMP requests, read the document *mib.txt* available by anonymous FTP from ftp.cisco.com. This document is available in RFC 1213-type format. It describes all the Cisco-specific SNMP variables in the Cisco portion of the MIB. It also describes what is required to establish minimum configuration. Contact Cisco Systems to obtain a copy of this document, which includes instructions for accessing the variables using SNMP.

Configuring the Trivial File Transfer Protocol (TFTP) Server

You can configure the network server to act as a limited Trivial File Transfer Protocol (TFTP) server from which other Cisco servers can boot their software. As a TFTP server host, the network server responds to TFTP read request messages by sending a copy of its ROM software to the requesting host. The TFTP read request message must use the file name that you specified in the network server configuration.

To specify TFTP server operation for a communications server, use the **tftp-server system** global configuration command. The full syntax follows.

tftp-server system *filename list*

no tftp-server system *filename list*

This command has two arguments: *filename* and *list*. The argument *filename* is the name you give the communications server ROM file, and the argument *list* is an IP access-list number.

The system sends a copy of the ROM software to any host which issues a TFTP read request with this file name. To learn how to specify an access list, see the "Configuring IP Access Lists" section in Chapter 13.

You can specify multiple file names by repeating the **tftp-server system** command. To remove a previously defined file name, use the **no tftp-server system** command and append the appropriate file name and an access-list number.

Example:

This command causes the router to send, via TFTP, a copy of the ROM software when it receives a TFTP read request for the file *configfile*. The requesting host is checked against access list 22.

```
tftp-server system configfile 22
```

Tailoring Use of Network Services

The **service** global configuration command tailors use by the network server of network-based services. Some **service** commands also configure system defaults; see **decimal-tty** for an example. The full command syntax follows:

service *keyword*

no service *keyword*

The argument *keyword* is one of the following:

- **config**—Specifies TFTP autoloading of configuration files; disabled by default on system with nonvolatile memory.
- **decimal-tty**—Specifies that line numbers be displayed and interpreted as decimal numbers rather than octal numbers; disabled by default.
- **finger**—Allows Finger protocol requests (defined in RFC 742) to be made of the network server; enabled by default. This service is equivalent to issuing a remote **show users** command.
- **tcp-keepalives-{in | out}**—Generates keepalive packets on idle network connections. The **in** keyword generates them on incoming connections (initiated by remote host); the **out** keyword generates them on outgoing connections (initiated by a user). There is a column in the EXEC **show tcp** display showing the keepalive statistics. The `wakeups` row shows how many keepalives have been transmitted without receiving any response (this is reset to 0 when a response is received).

The **no service** command disables the specified service or function.

Example:

The following command enables TFTP autoloading of configuration files:

```
service config
```

Redirecting System Error Messages

By default, the network server sends the output from the EXEC command **debug** and system error messages to the console terminal.

To redirect these messages, as well as output from asynchronous events such as interface transition, to other destinations, use the **logging** configuration command options.

These destinations include the console terminal, virtual terminals, and UNIX hosts running a syslog server; the syslog format is compatible with 4.3 BSD UNIX.

To configure the logging of messages, you need to be in the configuration command collection mode. To enter this mode, use the EXEC command **configure** at the EXEC prompt (see the section "Entering Configuration Mode" in Chapter 2 for the procedure).

The following sections describe how to implement these redirection options.

Enabling Message Logging

To enable or disable message logging, use the following global configuration commands:

logging on

no logging on

The **logging on** command enables message logging to all supported destinations other than the console. This behavior is the default.

The **no logging on** command enables logging to the console terminal only.

Logging Messages to an Internal Buffer

The default logging device is the console; all messages are displayed on the console unless otherwise specified.

To log messages to an internal buffer, use the logging buffered global configuration command. The full command syntax follows.

logging buffered

no logging buffered

The **logging buffered** command copies logging messages to an internal buffer instead of writing them to the console terminal. The buffer is circular in nature, so newer messages overwrite older messages. To display the messages that are logged in the buffer, use the EXEC command **show logging**. The first message displayed is the oldest message in the buffer.

The **no logging buffered** command cancels the use of the buffer and writes messages to the console terminal, which is the default.

Logging Messages to the Console

To limit how many messages are logged to the console, use the **logging console** global configuration command. The full syntax of this command follows:

logging console *level*

no logging console

The **logging console** command limits the logging messages displayed on the console terminal to messages with a level at or above the specified severity, which is specified by the *level* argument.

The argument *level* can be one of the following keywords, listed here in order from the most severe to the least severe level.

- **emergencies**—System unusable
- **alerts**—Immediate action needed
- **critical**—Critical conditions
- **errors**—Error conditions
- **warnings**—Warning conditions (output from **debug** commands are logged at this level)
- **notifications**—Normal but significant conditions
- **informational**—Informational messages only
- **debug**—Debugging messages

The default is to log messages at the **warnings** level to the console.

The **no logging console** command disables logging to the console terminal.

Example:

This command sets console logging of messages at the debug level:

```
!
logging console debug
!
```

Logging Messages to Another Monitor

To limit the level of messages to log to the terminal lines (monitors), use logging monitor command. The full syntax of this command follows.

logging monitor *level*

no logging monitor

The **logging monitor** command limits the logging messages displayed on terminal lines other than the console line to messages with a level at or above *level*. The argument *level* is one of the keywords described for the **logging console** command in the previous section,

"Logging Messages to the Console."To display logging messages on a terminal, use the privileged EXEC command **terminal monitor.**

The **no logging monitor** command disables logging to terminal lines other than the console line.

Example:

This command sets the level of messages displayed on monitors other than the console to notifications:

```
!
logging monitor notifications
!
```

Logging Messages to a UNIX Syslog Server

To log messages to the syslog server host, use the **logging** global configuration command. The full syntax is as follows:

logging *internet-address*

no logging *internet-address*

The **logging** command identifies a syslog server host to receive logging messages. The argument *internet-address* is the Internet address of the host. By issuing this command more than once, you build a list of syslog servers that receive logging messages.

The **no logging** command deletes the syslog server with the specified address from the list of syslogs.

Limiting Messages to a Syslog Server

To limit how many messages are sent to the syslog servers, use the **logging trap** global configuration command. Its full syntax follows:

logging trap *level*

no logging trap

The **logging trap** command limits the logging messages sent to syslog servers to messages with a level at or above *level*. The argument *level* is one of the keywords described for the **logging console** command in the earlier section, "Logging Messages to the Console."

To send logging messages to a syslog server, specify its host address with the **logging** command.

The **no logging trap** command disables logging to syslog servers.

The current software generates four categories of the syslog messages:

1. Error messages about software or hardware malfunctions, displayed at the errors level.
2. Output from the **debug** commands, displayed at the warnings level.
3. Interface up/down transitions and system restart messages, displayed at the notifications level.
4. Reload requests and low-process stack messages, displayed at the informational level.

The EXEC command **show logging** displays the addresses and levels associated with the current logging setup. The command output also includes ancillary statistics.

Example:

To set up the syslog daemon on a 4.3 BSD UNIX system, include a line such as the following in the file */etc/syslog.conf*:

```
local7.debug /usr/adm/logs/tiplog
```

The `local7` keyword specifies the logging facility to be used.

The `debug` argument specifies the syslog level. See the previous *level* arguments list for other arguments that can be listed.

The UNIX system sends messages at or below this level to the file specified in the next field. The file must already exist, and the syslog daemon must have permission to write to it.

Configuring Console and Virtual Terminal Lines

To configure your console and virtual terminal lines you need to be in the configuration command collection mode. To enter this mode, use the EXEC command **configure** at the EXEC prompt (see the section "Entering Configuration Mode" in Chapter 2 for the procedure).

Starting Line Configuration

To start configuring a terminal line, use the **line** command. This command identifies a specific line for configuration and starts line configuration command collection.

The **line** command has the following syntax:

line [*type-keyword*] *first-line* [*last-line*]

This command can take up to three arguments: a keyword, a line number, or a range of lines numbers.

The optional argument *type-keyword* specifies the type of line to be configured; it is one of the following keywords:

- **console**—Console terminal line
- **aux**—Auxiliary line, (described in the following section)
- **vty**—Virtual terminal for remote console access

When the line type is specified, the argument *first-line* is the relative number of the terminal line (or the first line in a contiguous group) you want to configure. Numbering begins with zero.

The optional argument *last-line* then is the relative number of the last line in a contiguous group you want to configure.

If you omit *type,* then *first-line* and *last-line* are absolute rather than relative line numbers. To display absolute line numbers, use the EXEC command **show users all.**

The network server displays an error message if you do not specify a line number.

Note: Line numbers, by default, are octal on the network servers.

The **line** command enables you to easily configure a large group of lines all at once. After you set the defaults for the group, you can use additional **line** commands and subcommands to set special characteristics, such as location, for individual terminal lines.

Example:

The following command starts configuration for the first five virtual terminal lines:

```
line vty 0 4
```

Configuring the CPU Auxiliary Port

The **line** command keyword **aux** allows use of an auxiliary RS-232 DTE port available on all processor cards. Use this port to attach to an RS-232 port of a CSU/DSU, protocol analyzer, or modem. You can monitor that port remotely by connecting to the TCP port whose number is 2000 decimal plus the line number of the auxiliary port. For example, if the auxiliary port was line 1 (obtained from the EXEC command **show users all**), then the TCP port would be 2001. You must order a special cable from Cisco Systems for use with this auxiliary port.

Note: You cannot use the auxiliary port as a second console port, nor can you initiate connections from this port. Its purpose is to *receive* connections from remote systems.

To configure the auxiliary port, use this variation of the **line** command:

line aux *port-address*

When configuring the auxiliary port, address it as line 0, as in this sample:

```
line aux 0
```

The auxiliary ports assert DTR only when a Telnet connection is established. The console port does not use RTS/CTS handshaking for flow control.

By default, the auxiliary port supports an EXEC process. This default can be re-enabled using the **exec** line subcommand.

Example:

These commands configure a second console port with a line speed of 2400 baud, and enable the EXEC.

```
line aux 0
exec
speed 2400
```

No modem control signals are supported on this line. If an auto-answer modem is configured on the line, you must dial up, log in, then hang up. The DTR signal will be active whenever an EXEC is configured on the auxiliary port.

Note: The EXEC is still present and may be used by the next person that dials into the number. This could cause security problems.

Establishing Line Passwords

When you start an EXEC on a line with password protection, the EXEC prompts for the password. If you enters the correct password, the EXEC prints its normal prompt. You may try three times to enter a password before the EXEC exits and returns the terminal to the idle state.

To specify a password, use the **password** line subcommand. Its full syntax follows.

password *text*

no password

The *text* argument may contain any alphanumeric characters, including spaces, up to 80 characters. The password checking is case sensitive. The password *Secret* is different than the password *secret*, for example, and the password *two words* is an acceptable password.

Example:

This command sets the words "Big Easy" as the password on line 1:

```
!
line 1
password Big Easy
!
```

Establishing Connection Restrictions

To establish connection restrictions on the lines to some Internet addresses, use the **access-class** line subcommand. The full command syntax follows.

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

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

The **access-class** subcommand 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** command removes access restrictions on the line for the specified connections.

Example:

This example subcommand sets restrictions on access list 1 for outgoing Telnet connections:

```
access-class 1 out
```

See the section "Configuring IP Access Lists" in Chapter 13 for information about configuring access lists.

Suppressing Banner Messages

By default, messages defined by the **banner motd** and **banner exec** commands are always displayed. This condition is defined by the **exec-banner** line subcommand. Its full syntax follows:

exec-banner

no exec-banner

To suppress display of a banner, enter the **no exec-banner** command.

Example:

These commands suppresses the banner on virtual terminal lines 0 through 4:

```
line vty 0 4
no exec-banner
```

Turning On/Off the Vacant Banner

The router will display a message on the console when there is no active EXEC. This message, called the vacant message, is different from the banner message displayed when an EXEC process is activated.

To turn the vacant message banner on or off, use the **vacant-message** line configuration subcommands. The **vacant-message** command enables the banner to be displayed on the screen of an idle terminal. The full syntax of this command follows.

vacant-message

vacant-message *c message c*

no vacant-message

The **vacant-message** subcommand without any arguments causes the default message to be displayed. If you desire a banner, follow **vacant-message** with one or more blank spaces and a delimiting character *(c)* you choose. Then type one or more lines of text (*message*), terminating the text with the second occurrence of the delimiting character.

The **no vacant-message** line configuration subcommand suppresses a banner message.

Example:

This example will turn on the system banner and display a message.

```
line 0
vacant-message #
                Welcome to Cisco Systems, Inc.
        This is the console terminal of the router Dross.
#
```

Note: You cannot use the delimiting character in the banner message.

Setting the Escape Character

The **escape-character** line subcommand defines the escape character. The following illustrates the full syntax of this command:

escape-character *decimal-number*

no escape-character

The argument *decimal-number* is the ASCII decimal representation of the desired escape character or an escape character (Ctrl-P, for example). Typing the escape character followed by the X key returns you to the EXEC when you are connected to another computer. The default escape character is Ctrl-^. (See Appendix E at the end of Volume II for a list of ASCII characters.)

The operating software interprets Break on the console as an attempt to halt the system.

Note: Depending upon the configuration register setting, console breaks will either be ignored or cause the server to shut down. The Break key *cannot* be used as the escape character on the Cisco router.

The **no escape-character** line configuration subcommand reinstates the default escape character.

Example:

This command changes the escape characters to Ctrl-P (ASCII character 17):

```
!
line 5
escape-character 17
!
```

Setting the Terminal Location

To set the location of the terminal, use the **location** line subcommand. The full syntax of this command follows:

location *text*

no location

This subcommand is for informational purposes only; it is not used by any aspects of the system software. The argument *text* is the desired description. The description appears in the output of the EXEC command **systat.** A maximum of 80 characters can be entered.

The **no location** subcommand removes the information.

Example:

This command describes the location of the terminal on line 2 as being Andrea's terminal:

```
!
line 2
location Andrea's terminal
!
```

Setting the EXEC Timeout Intervals

The EXEC command interpreter waits for a specified interval of time until the user starts input. If no input is detected, the EXEC resumes the current connection. If no connections exist, the EXEC returns the terminal to the idle state and disconnects the incoming session.

To set this interval, use the **exec-timeout** line configuration subcommand. The full syntax of the command follows.

exec-timeout *minutes* [*seconds*]

no exec-timeout

The argument *minutes* is the number of minutes, and the optional argument *seconds* specifies additional interval time in seconds. The default interval is ten minutes; an interval of zero specifies no time-outs.

The **no exec-timeout** subcommand removes the time-out definition. It is the same as entering **exec timeout 0**.

Example:

This command sets an interval of 2 minutes, 30 seconds.

```
exec-timeout 2 30
```

This command sets an interval of 10 seconds:

```
exec-timeout 0 10
```

Setting the Screen Length

To set the terminal screen length, use the **length** line configuration subcommand. The full syntax of the command follows.

length *screen-length*

no length

The argument *screen-length* is the number of lines on the screen. The network server uses this value to determine when to pause during multiple-screen output. The default length is 24 lines. A value of zero disables pausing between screens of output.

The **no length** command is the same as entering the command **length 0**.

Note: Not all commands pay attention to the configured screen length. For example, the **show terminal** command assumes a screen length of 24 lines or more.

Setting Notification

To enable the terminal to notify the user about pending output, use the **notify** line subcommand. The full syntax of the command follows.

notify

no notify

The **notify** subcommand sets a line to inform a user who has multiple, concurrent Telnet connections when output is pending on a connection other than the current connection.

The **no notify** line configuration subcommand ends notification and is the default.

Setting Character Padding

To set the padding on characters, use the **padding** line configuration subcommand. The full syntax of the command follows.

padding *decimal-number count*

no padding *decimal-number*

The **padding** subcommand sets padding for a specified output character. The argument *decimal-number* is the ASCII decimal representation of the character, and the argument *count* is the number of NUL bytes sent after that character.

The **no padding** line configuration subcommand removes padding for the specified output character.

Example:

The following command pads Return (ASCII character 13) with 25 NUL bytes:

```
padding 13 25
```

Global System Configuration Command Summary

This section lists all of the global system configuration commands in alphabetical order.

banner {**motd** | **exec** | **incoming**} *c text c*

Displays the message that the EXEC command interpreter displays whenever a user starts any EXEC process or activates a line. The **motd**, **exec**, and **incoming** keywords control when the banner message is displayed. The argument *c* specifies a delimiting character of your choice. The argument *text* specifies the message to be shown on the screen whenever an interface line is activated.

[**no**] **boot buffersize** *bytes*

Specifies the size of the buffer to be used for netbooting a host or a network configuration file. The argument *bytes* by default is the size of your nonvolatile memory, or 32 kilobytes if you do not have nonvolatile memory. There is no minimum or maximum size that may be specified.

[**no**] **boot host** *filename* [*address*]

Specifies the host configuration file name. The argument *filename* is the new name for the host configuration file. If you omit the argument *address,* the network server uses the default broadcast address of *255.255.255.255.* The optional argument *address* allows you to specify a specific network host or a subnet broadcast address.

[**no**] **boot network** *filename* [*address*]

Specifies the network configuration file. The argument *filename* is the new name for the network configuration file. If you omit the optional argument *address,* the network server uses the default broadcast address of *255.255.255.255.* The optional argument *address* allows you to specify a specific network host or a subnet broadcast address.

[**no**] **boot system** *filename* [*address*]

Specifies a second operating software image from a file that is not in nonvolatile memory. The argument *filename* is the file name of the operating software to load, and the optional argument *address* is the address of the network host holding that file.

[no] buffers {small | middle | big | large | huge} {permanent | max-free | min-free | initial} *number*

Allows a network administrator to adjust initial buffer pool settings and set limits at which temporary buffers are created and destroyed. The first keyword denotes the size of buffers in the pool; the default number of the buffers in a pool is determined by the hardware configuration. The second keyword specifies the buffer management parameter to be changed, as follows:

- **permanent**—The number of permanent buffers that the system tries to allocate.
- **max-free**—The maximum number of free or unallocated buffers in a buffer pool.
- **min-free**—The minimum number of free or unallocated buffers in a buffer pool.
- **initial**—The number of additional temporary buffers which should be allocated when the system is reloaded.

The argument *number* specifies the number of buffers to be allocated.

The **no buffers** command with appropriate keywords and arguments restores the default buffer values.

enable password *password*

Assigns a password for the privileged command level. The argument password is case sensitive and specifies the password prompted for in response to the EXEC command **enable**.

[no] enable last-resort {succeed | password}

Allows you to specify what happens if the TACACS servers used by the **enable** command do not respond. The default action is to fail. The keywords change this default action:

- **succeed**—Allows you to enable without further question.
- **password**—Allows you to enable by entering the privileged command level.

[no] enable use-tacacs

Enables or disables use of TACACS to check the user ID and password supplied to the EXEC **enable** command.

hostname *name*

Specifies the name for the network server. The default is *Gateway.*

[no] logging *internet-address*

Identifies a syslog server host to receive logging messages. The argument *internet-address* is the Internet address of the host.

[no] logging buffered

Copies logging messages to an internal buffer instead of writing them to the console.

[no] logging console *level*

Limits the logging of messages displayed on the console terminal to messages with a level at or above the specified severity, which is specified by the *level* argument. The argument *level* can be one of the following keywords, listed here in order from the most severe to the least severe level.

- **emergencies**—System unusable
- **alerts**—Immediate action needed
- **critical**—Critical conditions
- **errors**—Error conditions
- **warnings**—Warning conditions (default)
- **notifications**—Normal but significant conditions
- **informational**—Informational messages only
- **debug**—Debugging messages

[no] logging monitor *level*

Limits the logging messages displayed on terminal lines other than the console line to messages with a level at or above *level*. The argument *level* is one of the keywords described for the **logging console** command.

[no] logging on

Enables or disables message logging to all supported destinations except the console. This behavior is the default.

[no] logging trap *level*

Limits the logging messages sent to syslog servers to messages with a level at or above *level*. The argument *level* is one of the keywords described for the **logging console** command.

[no] service *keyword*

Tailors use by the network server of network-based services. The argument *keyword* is one of the following:

- **config**—Specifies TFTP autoloading of configuration files; disabled by default on system with nonvolatile memory.
- **decimal-tty**—Specifies that line numbers be displayed and interpreted as decimal numbers rather than octal numbers; disabled by default.
- **finger**—Allows Finger protocol requests (defined in RFC 742) to be made of the network server; enabled by default.
- **tcp-keepalives-{in | out}**—Generates keepalive packets on idle network connections. The **in** keyword generates them on incoming connections; the **out** keyword generates them on outgoing connections.

no snmp-server

Disables the SNMP operations.

[no] snmp-server access-list *list*

Sets up an access list that determines which hosts can send requests to the network server. The argument *list* is an integer from 1 through 99 that specifies an IP access list.

[no] snmp-server community *string* **[RO | RW]**[*list*]

Enables or disables SNMP server operation on the network server. The argument *string* specifies a community string that acts like a password and permits access to the SNMP protocol.

[no] snmp-server host *address community-string* **[snmp | tty | x25]**

Specifies which host or hosts should receive TRAP messages. Issue the **snmp-server host** command once for each host acting as a TRAP recipient. The argument *address* is the name or Internet address of the host. The argument *community-string* is the password-like community string set with the **snmp-server community** command. These optional keywords direct the TRAP:

- **snmp**—Causes all SNMP-type TRAP messages to be sent and starts the Cisco-specific RELOAD TRAP message.
- **tty**—Causes TCP connection TRAP messages to be included.

[no] snmp-server packet-size *bytes*

Sets or removes control over the largest SNMP packet size permitted when the SNMP server is receiving a request or generating a reply. The argument *bytes* is a byte count from 484 through 8,192. The default is 484.

[no] snmp-server queue-length *length*

Defines the length of the message queue for each TRAP host. The argument *length* is the number of TRAP events that can be held before the queue must be emptied; the default is ten.

[no] snmp-server system-shutdown

Allows or restricts use of the SNMP message reload feature, and prevents an SNMP system-shutdown request from resetting the Cisco agent.

[no] snmp-server trap-authentication

Allows or restricts the network server from sending a TRAP message when it receives a packet with an incorrect community string.

[no] snmp-server trap-timeout *seconds*

Defines how often to try resending TRAP messages on the retransmission queue. The argument *seconds* sets the interval for resending the messages. The default is set to 30 seconds. The **no** form of the command restores the default.

[no] tacacs-server attempts *count*

Controls the number of login attempts that may be made on a line set up for TACACS verification. The argument *count* is the number of attempts. The default is three attempts.

[no] tacacs-server authenticate {connect | slip | enable}

Specifies that a response is required from the network or communications server to indicate whether the user may perform the indicated action. Actions which require a response include the following:

- **connect**—Makes TCP connections
- **slip**—SLIP connections (Terminal Server only)
- **enable**—Use of **enable** command

The **no** form of the command disables the action.

[no] tacacs-server extended

Enables or disables an extended TACACS mode. This mode provides information about the terminal requests for use in setting up UNIX auditing trails and accounting files for tracking use of terminal servers and routers.

[no] tacacs-server host *name*

Specifies a TACACS host. The argument *name* is the name or Internet address of the host. You can use multiple commands to specify multiple hosts.

[no] tacacs-server last-resort {password | succeed}

Causes the network server to request the privileged password as verification, or forces successful login without further input from the user, depending upon the keyword specified, as follows.

- **password**—Allows the user to access the privileged-level command mode by entering the password set by the **enable** command.
- **succeed**—Allows the user to access the privileged-level command mode without further question.

The **no** form of the command disables the action.

tacacs-server notify {connect | slip | enable | logout}

Causes a message to be transmitted to the TACACS server, with retransmission being performed by a background process for up to five minutes. The keywords specify notification of the TACACS server whenever a user does one of the following:

- **connect**—Makes TCP connections.
- **slip**—Turns SLIP on or off (Terminal Server only).
- **enable**–Enters the **enable** command.
- **logout**—Logs out.

The **no** form of the command disables the messages.

[no] tacacs-server retransmit *retries*

Specifies the number of times the server will search the list of TACACS server hosts before giving up. The argument *retries* is the retransmit count. The default is two retries. The **no** form of the command restores the default.

[no] tacacs-server timeout *seconds*

Sets the interval the server waits for a server host to reply. The argument *seconds* specifies the number of seconds. The default interval is five seconds. The **no** form of the command restores the default.

[no] tftp-server system *filename ip-access-list*

Specifies or removes TFTP server operation for a communications server. The argument *filename* is the name given to the network server ROM file, and the argument *access-list* is an IP access list number.

Line Configuration Subcommand Summary

This section contains a summary of all the line configuration subcommands in alphabetical order.

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

Restricts or permits 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] escape-character *decimal-number*

Sets or removes the escape character on the specified line. The argument *decimal-number* is either the ASCII decimal representation of the character or a control sequence (Ctrl-E, for example). The default escape character is Ctrl-^.

[no] exec-banner

Enables or disables a banner. By default, a banner is displayed on the console. To suppress display of a banner, enter the **no** variation of the command.

[no] exec-timeout *minutes* [*seconds*]

Sets the interval the EXEC waits for user input before resuming the current connection, or if no connections exist, before returning the terminal to the idle state and disconnecting the incoming session. The argument *minutes* is the number of minutes, and the optional argument *seconds* specifies additional interval time in seconds. The default interval is ten minutes; an interval of zero specifies no time-outs. The **no** form of the command restores the default.

[no] length *screen-length*

Sets the terminal screen length. The argument *screen-length* is the number of lines on the screen. The network server uses this value to determine when to pause during multiple-screen output. The default length is 24 lines. A value of 0 (zero) or the **no** form of the command disables pausing between screens of output.

line [*type-keyword*] *first-line* [*last-line*]

Identifies a specific line for configuration and starts line configuration command collection. The optional argument *type-keyword* specifies the type of line to be configured, it is **console**, **aux**, or **vty** . When the line type is specified, the argument *first-line* is the relative number of the terminal line (or the first line in a contiguous group) you want to configure. The optional argument *last-line* then is the relative number of the last line in a contiguous group you want to configure. If you omit *type,* then *first-line* and *last-line* are absolute rather than relative line numbers.

[no] location *text*

Enters or removes informational only information about the terminal location and/or status. The argument *text* is the desired description.

[no] login

Enables or disables password checking for the password specified by the **password** command.

[no] login tacacs

Causes the TACACS user ID and password checking mechanism to be used instead of the regular password checking. The **no** form of the command disables this mechanism.

[no] notify

Enables or disables line notification of when a user who has multiple, concurrent Telnet connections has output pending on a connection other than the current line.

[no] padding *decimal-number count*

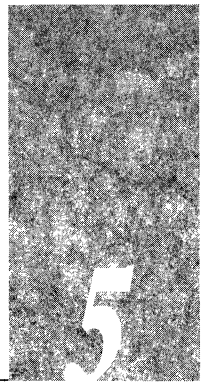
Sets or unsets padding for a specified output character. The argument *decimal-number* is the ASCII decimal representation of the character, and the argument *count* is the number of NUL bytes sent after that character.

[no] password *text*

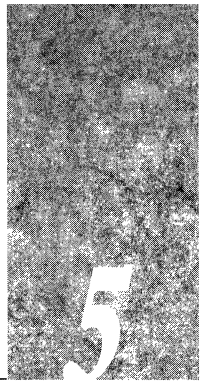
Specifies a password. The *text* argument specifies a password. It may contain any alphanumeric characters, including spaces, up to 80 characters.

vacant-message *c message c*
[no] vacant-message

Controls whether or not a banner is displayed on the screen of an idle terminal. The command without any arguments causes the default message to be displayed. The **no vacant-message** command suppresses a banner message. To display a banner, follow **vacant-message** with one or more blank spaces and a delimiting character (*c*) you choose, then type one or more lines of text (*message*), and terminate the text with the second occurrence of the delimiting character.



Chapter 5
Managing the System



Chapter 5
Managing the System

This chapter describes the EXEC commands you use to monitor and troubleshoot general system processes on your network server. These processes include:

- System memory allocator and buffer pools
- System configuration and processes
- Stack utilization
- System error message logs

This chapter also provides a general overview of these system tasks:

- Using the **debug** commands to troubleshoot network problems
- Testing connectivity
- Tracing routes
- Testing the system

See the chapters containing information about the interfaces and the protocols supported by Cisco software for descriptions of the interface- and protocol-specific debugging and monitoring commands.

Most of the network management commands are executed at the privileged-level prompt, although there is a subset of monitoring commands that you may enter at the user-level prompt.

An alphabetically arranged summary of the commands described in this chapters is included at the end of this chapter.

Monitoring System Processes

Use the EXEC show commands to display data structures, configuration parameters, and usage statistics for the network server. To list all the available **show** command options, enter this command at the EXEC prompt.

show ?

Two different lists may be displayed, one at the user-level prompt, and one at the enabled, privileged-level prompt. The lists include a summary of the command function, for easy reference. See Chapter 2 for the procedure to enter these levels.

Displaying Buffer Pool Statistics

The network server has one pool of queuing elements and five pools of packet buffers of different sizes. For each pool, the network server keeps counts of the number of buffers outstanding, the number of buffers in the free list, and the maximum number of buffers allowed in the free list. To display statistics for the buffer pools on the network server, use the **show buffers** command. Enter this command at the EXEC prompt:

show buffers [*interface*]

The optional argument *interface* causes a search of all buffers that have been associated with that interface for longer than one minute. The contents of these buffers will be printed to the screen. This option is useful in diagnosing problems where the input queue count on an interface is consistently nonzero.

Following is sample output without the optional *interface* argument. Table 5-1 describes the fields seen.

```
Buffer elements:
     250 in free list (250 max allowed)
     10816 hits, 0 misses, 0 created
Small buffers, 104 bytes (total 120, permanent 120):
     120 in free list (0 min, 250 max allowed)
     26665 hits, 0 misses, 0 trims, 0 created
Middle buffers, 600 bytes (total 90, permanent 90):
     90 in free list (0 min, 200 max allowed)
     5468 hits, 0 misses, 0 trims, 0 created
Big buffers, 1524 bytes (total 90, permanent 90):
     90 in free list (0 min, 300 max allowed)
     1447 hits, 0 misses, 0 trims, 0 created
Large buffers, 5024 bytes (total 0, permanent 0):
     0 in free list (0 min, 100 max allowed)
     0 hits, 0 misses, 0 trims, 0 created
Huge buffers, 12024 bytes (total 0, permanent 0):
     0 in free list (0 min, 30 max allowed)
     0 hits, 0 misses, 0 trims, 0 created

0 failures (0 no memory)
```

Table 5-1 Show Buffers Field Descriptions

Field	Description
Buffer elements	Blocks of memory used in internal operating system queues
Small buffers Middle buffers Big buffers Large buffers Huge buffers	Blocks of memory used to hold network packets
hits	Count of successful attempts to allocate a buffer when needed
misses	Count of allocation attempts which failed for lack of a free buffer in the pool
created	Count of new buffers created
trims	Count of buffers destroyed
in free list	Number of buffers of a given type which are not currently allocated and are available for use
max allowed	The maximum number of buffers of a given type allowed in the system
failures	The total number of allocation requests that have failed for lack of a free buffer
no memory	Number of failures due to a lack of memory to create a new buffer

Displaying System Memory Statistics

To show statistics about the memory, use the **show memory** command. Enter this command at the EXEC prompt:

show memory

This command displays memory free pool statistics. These statistics include summary information about the activities of the system memory allocator, and a block-by-block listing of memory use. Sample output follows. Table 5-2 describes the fields seen; Table 5-3 lists the characteristics of each block of memory in the system.

```
 Head       Free Start   Total Bytes     Used Bytes      Free Bytes
Processor  148B8C     1D66B0       2847860         561252          2286608

Address   Bytes Prev.   Next    Free? PrevF   NextF   Alloc PC  What
148B8C      916 0       148F20                        7B8E      *Init*
148F20     2024 148B8C  149708                        ADEE      *Init*
149708      536 148F20  149920                        AE18      *Init*
149920     2024 149708  14A108                        4A7C      *Init*
14A108       72 149920  14A150                        2492C     *Init*
14A150       44 14A108  14A17C                        36114     *Init*
14A17C      152 14A150  14A214                        1DBC      *Init*
14A214     2024 14A17C  14A9FC                        1DE0      *Init*
14A9FC      152 14A214  14AA94                        1DBC      *Init*
14AA94     2024 14A9FC  14B27C                        1DE0      *Init*
14B27C      100 14AA94  14B2E0                        3F2FE     Logger
14B2E0      152 14B27C  14B378                        1DBC      Router Init
14B378      480 14B2E0  14B558    y   1CE270  1CA938  5E85A     IGRP Router
14B558      100 14B378  14B5BC                        57A8E     DECnet Input
14B5BC       60 14B558  14B5F8    y   1CAFC4  1CA508  5EF08     IGRP Router
14B5F8       72 14B5BC  14B640                        7B830     XNS Router
14B640       72 14B5F8  14B688                        7B830     XNS Router
14B688       52 14B640  14B6BC                        7B840     XNS Router
14B6BC       88 14B688  14B714                        C724C     Virtual Exec
14B714       60 14B6BC  14B750                        C7602     Virtual Exec
14B750      480 14B714  14B930    y   1D1768  1CAFC4  5E85A     IGRP Router
```

Table 5-2 Show Memory Field Descriptions

Field	Description
Head	The hexadecimal address of the head of the memory allocation chain
Free Start	The hexadecimal address of the base of the free list
Total Bytes	The total amount of system memory
Used Bytes	The amount of memory in use
Free Bytes	The amount of memory not in use

Table 5-3 Characteristics of Each Block of Memory

Field	Description
Address	Hexadecimal address of block
Bytes	Size of block in bytes
Prev	Address of previous block (should match Address on previous line)
Next	Address of next block (should match address on next line)
Free?	Tells if the block is free
Alloc PC	Address of the system call that allocated the block
What	Name of process that owns the block

Displaying Active System Processes

To see information about the active processes, use the **show processes** command. Enter this command at the EXEC prompt:

show processes

Following is a partial display of the command output. Table 5-4 describes the fields seen.

```
PID Q T     PC Runtime (ms)     Invoked   uSecs   Stacks  TTY Process
  1 M E  17518        40072         830   48279  606/800    0 Net Background
  2 M E   5040          932          11   84727  486/800    0 Logger
 32 M E  4C390        73012        6141   11889  480/800    0 IGRP Router
  4 M E  22984          172         252     682  662/800    0 BOOTP Server
  5 H E   5040       100324       66619    1505  606/900    0 IP Input
  6 M E  21278        12188       22451     542  508/800    0 IP Protocols
  7 M E  32F32           32       10926       2  592/800    0 TCP Timer
  8 L E  33C1E          508          28   18142  576/800    0 TCP Protocols
  9 L E   5040         1104         935    1180  666/800    0 ARP Input
 10 L E   5040          352         458     768  674/800    0 Probe Input
 11 H E   5040         2636        9077     290  710/800    0 Net Input
 12 M T   2CF2        36976       49175     751  602/800    0 TTY Background
 13 H E   5040            0           2       0  852/900    0 DECnet Input
 14 M E  44AE4        21964       18029    1218  742/900    0 DECnet Routing
 15 H E   5040         2848        6556     434  686/800    0 XNS Input
 16 L E   5040          220        1642     133  748/800    0 XNS Protocols
 17 L E  59794        13660        3310    4126  632/800    0 XNS Router
 18 H E   5040        16652       59410     280  832/1000   0 AT Input
 19 L E   5040        13508       59201     228  884/1000   0 AT Protocols
 20 L E  73E0C       107496       40799    2634  832/1000   0 AT RTMP
 21 L E   5040           56          31    1806  814/1000   0 AT NBP
 22 L E  73E4A        84928       28526    2977  736/1000   0 AT ZIP
```

Table 5-4 Show Processes Field Descriptions

Field	Description
PID	Process ID
Q	Queue priority (high, medium, low)
T	Scheduler test (Event, Time, Suspended)
PC	Current program counter
Runtime (ms)	CPU time the process has used, in milliseconds
Invoked	Number of times the process has been invoked
uSecs	Microseconds of CPU time for each invocation
Stacks	Low water mark/Total stack space available
TTY Process	Terminal that controls the process and name of process

Note: Because the network server has a 4-millisecond clock resolution, run times are considered reliable only after a large number of invocations or a reasonable, measured run time.

Displaying Stack Utilization

To show stack utilization, use the **show stacks** command. Enter this command at the EXEC prompt:

show stacks

The **show stacks** command monitors the stack utilization of processes and interrupt routines. Its display includes the reason for the last system reboot. If the system was reloaded because of a system failure, a saved system stack trace is displayed. This information can be useful for analyzing crashes in the field.

Displaying the System Configuration

To display the contents of the nonvolatile memory, if present and valid, use the **show configuration** command. Enter this command at the EXEC prompt:

show configuration

The nonvolatile memory stores the configuration information in the network server in text form as configuration commands.

Displaying the Error Logging Conditions

To show the state of logging (syslog), use the **show logging** command. Enter this command at the EXEC prompt:

show logging

This command displays the state of syslog error and event logging, including host addresses, and whether console logging is enabled. This command also displays SNMP (Simple Network Management Protocol) configuration parameters and protocol activity. See the section "Redirecting System Error Messages" in Chapter 4, for an explanation of how to configure message logging. Following is a sample output. Table 5-5 describes the fields seen.

```
Syslog logging: enabled
    Console logging: disabled
    Monitor logging: level debugging, 266 messages logged.
    Trap logging: level informational, 266 messages logged.
    Logging to 131.108.2.238

SNMP logging: disabled, retransmission after 30 seconds
    0 messages logged
```

Table 5-5 Show Logging Field Descriptions

Field	Description
Syslog logging	When enabled, system logging messages are sent to a UNIX host which acts as a syslog server—that is, it captures and saves the messages.
Console logging	If enabled, states the level; otherwise this field displays disabled.
Monitor logging	The minimum level of severity required for a log message to be sent to a monitor terminal (not the console).
Trap logging	The minimum level of severity required for a log message to be sent to syslog server.
SNMP logging	Shows whether SNMP logging is enabled and the number of messages logged, and the retransmission interval.

Displaying Protocol Information

To display the configured protocols, use the **show protocols** command. Enter this command at the EXEC prompt:

show protocols

The command shows the global and interface-specific status of any configured Level 3 protocol; for example, IP, DECnet, Novell, AppleTalk, and so forth. The following is sample output:

```
Global values:
  Internet Protocol routing is enabled
  DECNET routing is enabled
  XNS routing is enabled
  Appletalk routing is enabled
  X.25 routing is enabled
Ethernet 0 is up, line protocol is up
  Internet address is 131.108.1.1, subnet mask is 255.255.255.0
  Decnet cost is 5
  XNS address is 2001.AA00.0400.06CC
  AppleTalk address is 4.129, zone Twilight
Serial 0 is up, line protocol is up
  Internet address is 192.31.7.49, subnet mask is 255.255.255.240
Ethernet 1 is up, line protocol is up
  Internet address is 131.108.2.1, subnet mask is 255.255.255.0
  Decnet cost is 5
  XNS address is 2002.AA00.0400.06CC
  AppleTalk address is 254.132, zone Twilight
Serial 1 is down, line protocol is down
  Internet address is 192.31.7.177, subnet mask is 255.255.255.240
  AppleTalk address is 999.1, zone Magnolia Estates
```

Troubleshooting Network Operations

The network server includes software to aid in tracking down problems with the network server or with other hosts on the network. The privileged EXEC command **debug** enables the display of several classes of network events on the console terminal. The privileged **undebug** command turns off the display of these classes. The EXEC command **show debugging** displays the state of each debugging option. See the section "Redirecting System Error Messages" in Chapter 4 for an explanation of how to configure message logging.

Note: Debugging output is given high priority by the system. For this reason, debugging commands should be turned on only for troubleshooting specific problems, or during troubleshooting sessions with Cisco engineers. Excessive debugging output can render the system unusable.

To list and briefly describe all the **debug** command options, enter the **debug ?** command at the privileged-level EXEC prompt. This section provides an overview of how to use the debugging commands. See the interface and protocol-specific chapters for the **debug** command descriptions.

To turn on all system diagnostics, enter this command at the EXEC prompt:

debug all

Its converse, the **undebug all** command, turns off all diagnostic output.

Testing Connectivity with the Ping Command

As an aid to diagnosing basic network connectivity, many network protocols support the *packet internet groper* (ping) program, which sends an echo request packet to an address, then awaits a reply. Results from this echo protocol can help in evaluating the path-to-host reliability, delays over the path, and whether the host can be reached or is functioning.

Note: Not all protocols require hosts to support pings, and for some protocols, the pings are Cisco-defined and are only answered by another Cisco router.

To implement this program, use the privileged EXEC command ping. When the ping command is entered, the system issues a prompt for one of the following protocol keywords—**appletalk**, **clns**, **ip**, **novell**, **pup**, or **xns**.

The default protocol is IP. After determining the protocol type, the **ping** command prompts for an address or host name, repeat count (default is 5), datagram size (default is 100 bytes), time-out interval (default is 2 seconds), and extended commands (default is none). The precise dialog varies from protocol to protocol.

If a host name or address is typed on the same line as the EXEC **ping** command, the default actions will be taken as appropriate for the protocol type of that name or address.

The **ping** command uses the exclamation point (!) and period (.) in its display. Each exclamation point indicates receipt of a reply. A period (.) indicates the network server timed out while waiting for a reply. Other characters may appear in the **ping** output display, depending on the protocol type. The output concludes with the success rate and minimum, average, and maximum round-trip times.

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

Sample displays and tips for using these protocols are included in the chapters describing the protocols supported by the Cisco **ping** command.

Checking Routes with the Trace Command

The **trace** command is a useful debugging command which allows the network administrator to discover the routes packets will actually take when travelling to their destination. The **trace** command supports IP, CLNS and VINES route tracing.

trace [*destination*]

To invoke a simple **trace** test, enter the destination address or host name on the command line. The default parameters for the appropriate protocol are assumed and the tracing action begins.

To use nondefault parameters and invoke an extended **trace** test, enter the command without a destination argument. You will be stepped through a dialog to select the desired parameters.

Typing the escape sequence (by default, Ctrl-^, X) terminates a **trace** command. See the IP, ISO CLNS and VINES chapters for more information about using this command.

Writing System Configuration Information

This section describes the privileged **write** commands used to manage the system configuration information.

To erase the configuration information, use the following EXEC command:

write erase

This command erases the configuration information in the nonvolatile memory. This command does not affect the configuration in use.

To copy the configuration to memory, use the following EXEC command:

write memory

This command copies the current configuration information to the nonvolatile memory.

To copy the configuration to the network, use the following EXEC command:

write network

This command sends a copy of the current configuration information to a server host. You are prompted for a destination host and a file name.

To write configuration on the terminal, use the following EXEC command:

write terminal

This command displays the current configuration information on the terminal.

Testing the System

Included as part of the EXEC command set are commands that allow testing of system interfaces and memory.

Caution: Use of these commands is not recommended, as they are intended to aid Cisco manufacturing personnel in checking out system functionality.

To test the system interfaces, use this EXEC command:

test interfaces

The EXEC **test interfaces** command is intended for the factory checkout of network interfaces. It is not intended for diagnosing problems with an operational router. The **test interfaces** output will not report correct results if the router is attached to a "live" network. For each network interface that has an IP address that can be tested in loop back (MCI and cBus Ethernet and all serial interfaces), the **test interface** command sends a series of ICMP echoes. Error counters are examined to determine the operational status of the interface.

To test system memory, use this EXEC command:

test memory

The EXEC command **test memory** performs a test of Multibus memory, including the nonvolatile memory.

Caution: This test will overwrite the contents of memory. You will need to rewrite nonvolatile memory after running this command. If you test Multibus memory (for example, the memory used by the CSC-R four megabit Token Ring interfaces), you will need to reload the system to restore correct operation of the network interfaces.

EXEC System Management Command Summary

This section list all the EXEC system management and user commands in alphabetical order.

debug ?

Lists and briefly describes all the **debug** command options.

[un]debug all

Enables all diagnostic output. Its converse, the **undebug all** command, turns off all diagnostic output.

ping

Provides a diagnostic tool for testing connectivity. Results help evaluate path-to-host reliability, delays over the path, and whether the host is functioning.

show ?

The **show ?** command lists all the **show** command options. Two lists may be displayed, one at the user-level prompt, and one at the enabled, privileged-level prompt.

show buffers [*interface*]

Displays statistics for the buffer pools on the network server. The network server has one pool of queuing elements and four pools of packet buffers of different sizes. For each pool, the network server keeps counts of the number of buffers outstanding, the number of buffers in the free list, and the maximum number of buffers allowed in the free list. With the optional argument *interface*, this command searches all buffers that have been associated with the interface for longer than one minute.

show configuration

Displays the contents of the nonvolatile memory, if present and valid. The nonvolatile memory stores the configuration information in the network server in text form as configuration commands.

show debugging

Displays the current settings of the **debug** command options.

show logging

Displays the state of syslog error and event logging, including host addresses and whether console logging is enabled. This command also displays SNMP configuration parameters and protocol activity.

show memory

Displays memory free pool statistics. These statistics include summary information about the activities of the system memory allocator, and a block-by-block listing of memory use.

show processes

Displays information about all active processes.

show protocols

Displays the global and interface-specific status of any configured Level 3 protocol.

show stacks

Monitors the stack utilization of processes and interrupt routines. Its display includes the reason for the last system reboot. If the system was reloaded because of a system failure, a saved system stack trace is displayed. This information can be useful to support personnel for analyzing crashes in the field.

test interfaces

Intended for the factory checkout of network interfaces. It is not intended for diagnosing problems with an operational router.

test memory

Performs a test of Multibus memory, including the nonvolatile memory

trace [*destination*]

Allows the network administrator to discover the routes packets will actually take when travelling to their destination. The **trace** command supports both IP and CLNS route tracing. Typing the escape sequence terminates **trace**.

write erase

Erases the configuration information in the nonvolatile memory. This command does not affect the configuration in use.

write memory

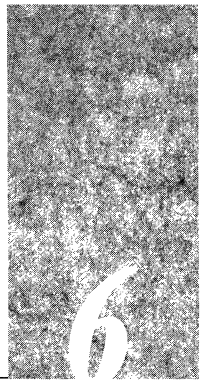
Copies the current configuration information to the nonvolatile memory.

write network

Sends a copy of the current configuration information to a server host. You are prompted for a destination host and a file name.

write terminal

Displays the current configuration information.



Chapter 6
Configuring the Interfaces

Interface Support EXEC Command Summary 6-51

Part 3
Interface Configuration

Chapter 6
Configuring the Interfaces

This chapter contains information on enabling, shutting down, and displaying information pertinent to interface configuration and operation. Also discussed in this chapter are the procedures and command descriptions for configuring and maintaining the following interface components of the router:

- Serial interfaces
- Ethernet interfaces
- Token Ring interfaces
- Fiber Distributed Data Interface (FDDI)
- High Speed Serial Interface (HSSI)
- UltraNet interface

You will also find information about configuring the serial Dial Backup feature, and how to configure a null interface and the Point-to-Point Protocol (PPP) in this chapter.

To enable an interface, you must be in the configuration command collection mode. To enter this mode, type the EXEC command **configure** at the EXEC prompt; the **configure** command will place the system into the configuration command collection mode. Once in the command collection mode, start configuring the interface by entering the **interface** command. Once an interface is configured, you can check its status by entering EXEC **show** commands at the EXEC prompt.

This chapter provides software configuration information only. For hardware technical descriptions, and for information about installing these interfaces, refer to these Cisco publications: *Modular Products Hardware Installation and Reference* or the *IGS Hardware Installation and Reference.*

Summaries of the interface configuration commands and EXEC monitoring commands described in this chapter are included at the end of the chapter.

Specifying an Interface

The **interface** command is entered in configuration mode and identifies a specific network interface (for example, a serial port, Ethernet port, or a Token Ring port.) By entering this command you begin the command collection mode for the specified interface.

The **interface** command has the following syntax:

interface *type unit*

The argument *type* identifies the interface type and the argument *unit* identifies the connector or interface card number. Unit numbers are assigned at the factory at the time of installation, or when added to a system, and can be displayed with the **show interfaces** command.

Example:

This example begins interface configuration command collection mode for serial connector 0 (interface serial 0).

```
interface serial 0
```

Use the EXEC command **show interfaces** (described later in this chapter), to determine the interface type and unit numbers.

In the interface configuration command collection mode, you enter the interface subcommands for your particular routing or bridging protocol. The interface configuration command collection mode ends when you enter a command that is not an interface subcommand, or when you type the Ctrl-Z sequence.

Adding a Descriptive Name to an Interface

To add a descriptive name to an interface, use the **description** interface subcommand.

description *name-string*

no description

The argument *name-string* is text, or a description string to help you remember what is attached to this interface. The **description** command is meant solely as a comment to be put in the configuration to help you remember what certain interfaces are for. The description will appear in the output of the following commands: **show configuration, write terminal**, and **show interfaces.**

Example:

This example describes a 3174 controller on serial 0.

```
interface serial 0
description 3174 Controller for test lab
```

Shutting Down and Restarting an Interface

You disable an interface using the **shutdown** interface subcommand. The full syntax for this command follows:

shutdown

no shutdown

The **shutdown** command disables all functions on the specified interface, as well as prevents the transmission of all the packets. The command also marks the interface as unavailable, which is communicated to other network servers through all dynamic routing protocols. The interface will not be mentioned in any routing updates. On serial interfaces, this command causes the DTR signal to be dropped. On FDDI interfaces, this command causes the optical bypass switch, if present, to go into bypass mode. On Token Ring interfaces, this command causes the interface to de-insert from the ring.

To restart a disabled interface, use the **no shutdown** interface subcommand.

To check whether an interface is disabled, use the EXEC command **show interfaces** as described in the next section. An interface that has been shut down is shown as administratively down in the display from this command.

Examples:

These commands turn off the interface Ethernet 0.

```
interface ethernet 0
shutdown
```

These commands turn the interface back on.

```
interface ethernet 0
no shutdown
```

Clearing Interface Counters

To clear the interface counters shown with the **show interface** command, enter the following command at the EXEC prompt:

clear counters [*type unit*]

The command clears all the current interface counters from the interface unless the optional arguments *type* and *unit* are specified to clear only a specific interface type (serial, Ethernet, Token Ring, and so on.) from a specific unit or card number.

Note: This command will not clear counters retrieved using SNMP, but only those seen with the EXEC **show interface** command.

Displaying Information About an Interface

The Cisco software contains commands that you can enter at the EXEC prompt to display different information about the interface including the version of the software and the hardware, the controller status, and some statistics about the different interfaces. These commands begin with the word "show." (The full list of these commands can be displayed by entering the command **show ?** at the EXEC prompt.) A description of interface-specific **show** commands follow.

Displaying the System Configuration

The **show version** command displays the configuration of the system hardware, the software version, the names and sources of configuration files, and the boot images. Enter this command at the EXEC prompt:

show version

The following shows sample output from this command:

```
GS Software, Version 8.3
Copyright (c) 1986-1991 by cisco Systems, Inc.
Compiled Sat 14-Sep-91 04:05 by satz

System Bootstrap, Version 4.3

dross uptime is 1 week, 23 hours, 28 minutes
System restarted by reload
System image file is unknown, booted via tftp from 131.108.1.111
Host configuration file is "dross-confg", booted via tftp from
131.108.1.111

CSC3 (68020) processor with 4096K bytes of memory.
X.25 software.
Bridging software.
5 MCI controller.
8 Ethernet/IEEE 802.3 interface.
12 Serial network interface.
32K bytes of non-volatile configuration memory.
```

Displaying Controller Status

The **show controller** command displays current internal status information for different interface cards. Enter this command at the EXEC prompt:

show controller {serial | token | mci | cbus | fddi}

Use the following keywords to display the information about that card:

- **serial**—for the Serial Interface Card
- **token**—for the CSC-R and CSC-R16 Token Ring Interface Cards
- **mci**—for the Multiport Communications Interface Card
- **cbus**—for the cBus Controller Card
- **fddi**—for the FDDI Controller Card

Sample output for the MCI controller card follows. Table 6-1 describes the fields seen.

```
MCI 0, controller type 1.1, microcode version 1.8
  128 Kbytes of main memory, 4 Kbytes cache memory
22 system TX buffers, largest buffer size 1520
  Restarts: 0 line down, 0 hung output, 0 controller error
  Interface 0 is Ethernet0, station address 0000.0c00.d4a6
    15 total RX buffers, 11 buffer TX queue limit, buffer size 1520
    Transmitter delay is 0 microseconds
  Interface 1 is Serial0, electrical interface is V.35 DTE
    15 total RX buffers, 11 buffer TX queue limit, buffer size 1520
    Transmitter delay is 0 microseconds
    High speed synchronous serial interface
  Interface 2 is Ethernet1, station address aa00.0400.3be4
    15 total RX buffers, 11 buffer TX queue limit, buffer size 1520
    Transmitter delay is 0 microseconds
  Interface 3 is Serial1, electrical interface is V.35 DCE
    15 total RX buffers, 11 buffer TX queue limit, buffer size 1520
    Transmitter delay is 0 microseconds
    High speed synchronous serial interface
```

Table 6-1 Show Controller Field Descriptions

Field	Description
MCI (number)	The unit number of the MCI card
controller type	The version number of the MCI card
microcode version	The version number of the MCI card's internal software (in read-only memory)
main memory cache memory	The amount of main and cache memory on the card
system TX	Number of buffers that hold packets to be transmitted buffers

Restarts	Count of restarts due to the following conditions:
line down	Communication line down
hung output	Output unable to transmit
controller error	Internal error
interface..is	Names of interfaces, by number
electrical interface	Line interface type for serial connections
RX buffers	Number of buffers for received packets
TX queue limit	Maximum number of buffers in transmit queue
Transmitter delay	Delay between outgoing frames
Station address	The hardware address of the interface

Displaying Interface Statistics

The Cisco software also provides the **show interfaces** command which displays statistics for the network interfaces on the network server. Enter this command at the EXEC prompt:

show interfaces [*type unit*]

Specify the optional arguments *type* and *unit* to display statistics for a particular network interface. The argument *type* can be one of the following: **ethernet**, **serial**, **tokenring**, **fddi**, **hssi**, or **ultranet**. Use the argument *unit* to specify the interface unit number.

You will use the **show interfaces** command frequently while configuring and monitoring your modules.

For further explanations and examples about a specific interface, refer to the following sections in this chapter: "Monitoring the Serial Interface," "Monitoring the Ethernet Interface," "Monitoring the Token Ring Interface," "Monitoring the FDDI," "Monitoring the HSSI," and "Monitoring the UltraNet Interface."

Serial Interface Support

Support for the serial interface is supplied on one of Cisco Systems' serial network interface cards:

- The Multiport Communications Interface (MCI) card provides up to two high-speed synchronous serial port connectors on a single card that support RS-232, V.35, and RS-449 connections, and X.21 connections using the RS-449 connector.
- The Serial-Port Communication Interface (SCI) card provides up to four high-speed serial ports on a single card that support RS-232, V.35, and RS-449 connections, and X.21 connections using the RS-449 connector.

The high-speed synchronous serial interface is also supported on the IGS network server.

Specifying a Serial Interface

To specify a serial interface, use this configuration command:

interface serial *unit*

Specify the serial interface connector number with the argument *unit*.

Follow this command with the routing or bridging interface subcommands for your particular protocol or application as described in the chapters in Part Four and Part Five.

The SCI and MCI cards can query the appliques to determine their types. However, they do so only at system start-up, so the appliques must be attached when the system is started. Issue a **show controller serial** or **show controller mci** command to determine how the serial card has identified them. The command will also show the capabilities of the serial card and report controller-related failures.

Example:

This command begins configuration on interface serial 0.

```
interface serial 0
```

Serial Encapsulation Methods

The serial interfaces support the following kinds of serial encapsulations:

- High-Level Data Link Control (HDLC)
- HDLC Distant Host (HDH)
- Frame Relay
- Point-to-Point Protocol (PPP)
- Switched Multi-megabit Data Services (SMDS)
- X.25-based encapsulations

With the exception of the PPP and HDLC encapsulations, these serial encapsulation methods are described in Chapter 8, "Configuring Packet-Switched Software." The PPP and HDLC serial encapsulation methods are described in this chapter.

The encapsulation method is changed by using the interface configuration subcommand **encapsulation** followed by a keyword that defines the encapsulation method.

encapsulation *encapsulation-type*

The *encapsulation-type* argument is a keyword that identifies one of the following serial encapsulation types that Cisco Systems' software supports:

- **bfex25**—Blacker Front End Encryption X.25 operation
- **ddnx25-dce**—DDN X.25 DCE operation
- **ddnx25**—DDN X.25 DTE operation

- **frame-relay**—Frame Relay
- **hdh**—HDH Protocol
- **hdlc**—Cisco Systems HDLC Protocol
- **lapb-dce**—X.25 LAPB DCE operation
- **lapb**—X.25 LAPB DTE operation
- **multi-lapb-dce**—X.25 LAPB multiprotocol DCE operation
- **multi-lapb**—X.25 LAPB multiprotocol DTE operation
- **ppp**—Point-to-Point Protocol (PPP)
- **smds**—SMDS service
- **x25-dce**—X.25 DCE operation
- **x25**—X.25 DTE operation

HDLC Serial Encapsulation Method

Cisco provides HDLC serial encapsulation for serial lines. This encapsulation method provides the synchronous framing and error detection functions of HDLC without windowing or retransmission. Although HDLC is the default serial encapsulation method, it can be re-installed using the **hdlc** keyword with the **encapsulation** command as follows:

encapsulation hdlc

Maintaining the Serial Interface

Use the command **clear interface** to reset the hardware logic on an interface. Enter this command at the EXEC prompt:

clear interface *type unit*

The arguments *type* and *unit* specify a particular interface type and its unit number. In this case, the argument *type* is **serial**.

Note: Under normal circumstances, you do not need to clear the hardware logic on interfaces.

Monitoring the Serial Interface

Use the command **show interfaces** to display information about the serial interface and the state of source bridging. Enter this command at the EXEC prompt:

show interfaces [*type unit*]

The argument *type* is the keyword **serial**, and *unit* is the interface unit number. If you do not provide values for the parameters *type* and *unit*, the command will display statistics for all the network interfaces.

Sample output of this command for Cisco's synchronous serial interfaces is provided below: Table 6-2 describes the fields seen.

```
Serial 0 is up, line protocol is up
  Hardware is MCI Serial
  Internet address is 150.136.190.203, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input 0:00:07, output 0:00:00, output hang never
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 0 bits/sec, 0 packets/sec
     16263 packets input, 1347238 bytes, 0 no buffer
     Received 13983 broadcasts, 0 runts, 0 giants
     2 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 2 abort
     22146 packets output, 2383680 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets, 0 restarts
     1 carrier transitions
```

Table 6-2 Show Serial Interface Field Descriptions

Field	Description
Serial ... is {up | down} ...is administratively down	Tells whether the interface hardware is currently active (whether carrier detect is present) and if it has been taken down by an administrator.
line protocol is {up | down | administratively down}	Tells whether the software processes that handle the line protocol think the line is usable (are keepalives successful?).
Hardware is	Specifies the hardware type.
Internet address is	Specifies the Internet address and subnet mask, followed by packet size.
Encapsulation	Encapsulation method assigned to interface.
loopback	Tells whether loopback is set or not.
keepalive	Tells whether keepalives are set or not.
Last input	The number of hours, minutes, and seconds since the last packet was successfully received by an interface. Useful for knowing when a dead interface failed.
output hang	The number of hours, minutes, and seconds (or never) since the interface was last reset because of a transmission that took too long. When the number of hours in any of the "last" fields exceeds 24 hours, the number of days and hours is printed. If that field overflows, asterisks are printed.

Output queue, Input Queue, drops	Number of packets in output and input queues. Each number is followed by a slash, the maximum size of the queue, and the number of packets dropped due to a full queue.
Five minute input rate, Five minute output rate	The average number of bytes and packets transmitted per second in the last five minutes.
packets input	The total number of error-free packets received by the system.
broadcasts	The total number of broadcast or multicast packets received by the interface.
runts	The number of packets which are discarded because they are smaller than the medium's minimum packet size.
giants	The number of packets which are discarded because they exceed the medium's maximum packet size.
input error	The total number of no buffer, runts, giants, CRCs, frame, overrun, ignored, and abort counts. Other input-related errors can also increment the count, so that this sum may not balance with the other counts.
CRC	The Cyclic Redundancy Checksum generated by the originating station or far-end device does not match the checksum calculated from the data received. On a serial link, CRCs usually indicate noise, gain hits, or other transmission problems on the data link.
frame	The number of packets received incorrectly having a CRC error and a noninteger number of octets. On a serial line, this is usually the result of noise or other transmission problems.
overrun	The number of times the serial receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeded the receiver's ability to handle the data.
ignored	The number of received packets ignored by the interface because the interface hardware ran low on internal buffers. These buffers are different than the system buffers mentioned previously in the buffer description. Broadcast storms and bursts of noise can cause the ignored count to be increased.
abort	An illegal sequence of one bits on a serial interface. This usually indicates a clocking problem between the serial interface and the data link equipment.
packets output	Total number of messages transmitted by the system.

bytes output	Total number of bytes, including data and MAC encapsulation, transmitted by the system.
underruns	Number of times that the transmitter has been running faster than the router can handle. This may never happen (be reported) on some interfaces.
output errors	The sum of all errors which prevented the final transmission of datagrams out of the interface being examined. Note that this may not balance with the sum of the enumerated output errors, as some datagrams may have more than one error, and others may have errors that do not fall into any of the specifically tabulated categories.
interface resets	The number of times an interface has been completely reset. This can happen if packets queued for transmission were not sent within several seconds' time. On a serial line, this can be caused by a malfunctioning modem which is not supplying the transmit clock signal, or by a cable problem. If the system notices that the carrier detect line of a serial interface is up, but the line protocol is down, it periodically resets the interface in an effort to restart it. Interface resets can also occur when an interface is looped back or shut down.
restarts	The number of times the controller was restarted because of errors.
carrier transitions	The number of times the carrier detect signal of a serial interface has changed state. Indicates modem or line problems if the carrier detect line is changing state often.

Debugging the Serial Interface

Use the command **debug serial-interface** to debug serial interface events. Enter this command at the EXEC prompt:

```
debug serial-interface
```

Use the **undebug serial-interface** command to turn off messaging.

Ethernet Interface Support

Support for the Ethernet interface is supplied on one of Cisco Systems' Ethernet network interface cards:

- The Multiport Communications Interface (MCI) card provides up to two Ethernet connectors compatible with Ethernet Versions 1 and 2, and the IEEE 802.3 protocol.
- The Multiport Ethernet Controller (MEC) interface card provides two, four, or six high-speed Ethernet connectors compatible with Ethernet Versions 1 and 2, and the IEEE 802.3 protocol.

The Ethernet interface is also supported on the IGS network server.

Specifying an Ethernet Interface

To specify an Ethernet interface, use this configuration command:

interface ethernet *unit*

Specify the Ethernet interface connector number with the argument *unit*.

Follow this command with the routing or bridging interface subcommands for your particular protocol or application as described in the chapters in Part Four and Part Five.

Example:

This command begins configuration on interface Ethernet 1.

```
interface ethernet 1
```

Ethernet Encapsulation Methods

The Ethernet interface supports a number of encapsulation methods. These methods are assigned by using the interface subcommand **encapsulation** followed by a keyword that defines the encapsulation method. The particular encapsulation method used depends on the protocol. Currently, there are three common Ethernet encapsulation methods:

- The standard Ethernet version 2.0 encapsulation that uses a 16-bit protocol type code
- The IEEE 802.3 encapsulation, where the type code becomes the frame length for the IEEE 802.2 LLC encapsulation (destination and source Service Access Points, and a control byte)
- The SNAP method, as specified in RFC 1042, which allows Ethernet protocols to run on IEEE 802.2 media

The syntax of the **encapsulation** command follows:

encapsulation *encapsulation-type*

The *encapsulation-type* is one of the following three keywords:

- **arpa**—Standard Ethernet version 2.0 encapsulation
- **iso1**—IEEE 802.3 encapsulation
- **snap**—IEEE 802.2 Ethernet media

Example:

These commands enable standard Ethernet version 2.0 encapsulation on interface Ethernet 0.

```
interface ethernet 0
encapsulation arpa
```

Maintaining the Ethernet Interface

Use the command **clear interface** to reset the hardware logic on an interface. Enter this command at the EXEC prompt:

clear interface *type unit*

The arguments *type* and *unit* specify a particular interface type and its unit number. In this case, the argument *type* is **ethernet**.

Note: Under normal circumstances, you do not need to clear the hardware logic on interfaces.

Monitoring the Ethernet Interface

Use the command **show interfaces** to display information about the Ethernet interface. Enter this command at the EXEC prompt:

show interfaces [*type unit*]

Where *type* is the **ethernet** keyword and *unit* is the interface unit number. If you do not provide values for the parameters *type* and *unit*, the command will display statistics for all the network interfaces.

Sample output of this command is provided on the following page. Table 6-3 describes the fields seen.

```
Ethernet 0 is up, line protocol is up
  Hardware is MCI Ethernet, address is aa00.0400.0134 (bia
0000.0c00.4369)
  Internet address is 131.108.1.1, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, PROBE, ARP Timeout 4:00:00
  Last input 0:00:00, output 0:00:00, output hang never
  Output queue 0/40, 0 drops; input queue 0/75, 2 drops
  Five minute input rate 61000 bits/sec, 4 packets/sec
  Five minute output rate 1000 bits/sec, 2 packets/sec
     2295197 packets input, 305539992 bytes, 0 no buffer
     Received 1925500 broadcasts, 0 runts, 0 giants
     3 input errors, 3 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     3594664 packets output, 436549843 bytes, 0 underruns
     8 output errors, 1790 collisions, 10 interface resets, 0 restarts
```

Table 6-3 Show Ethernet Interface Field Descriptions

Field	Description
Ethernet ... is up ...is administratively down	Tells whether the interface hardware is currently active and if it's been taken down by an administrator.
line protocol is {up | down | administratively down}	Tells whether the software processes that handle the line protocol believe the interface is usable (are keepalives successful?).
Hardware	Specifies the hardware type (for example, MCI Ethernet, cBus Ethernet) and address.
Internet address	Lists the Internet address followed by subnet mask and then the packet size.
Encapsulation	Encapsulation method assigned to interface.
ARP type:	Type of Address Resolution Protocol assigned.
loopback	Tells whether loopback is set or not.
keepalive	Tells whether keepalives are set or not.
Last input	The number of hours, minutes, and seconds since the last packet was successfully received by an interface. Useful for knowing when a dead interface failed.
output	Number of hours, minutes, and seconds since the last packet was successfully transmitted by the interface. Useful for knowing when a dead interface failed.

output hang	The number of hours, minutes, and seconds (or never) since the interface was last reset because of a transmission that took too long. When the number of hours in any of the "last" fields exceeds 24 hours, the number of days and hours is printed. If that field overflows, asterisks are printed.
Last clearing	The time at which the counters that measure cumulative statistics (such as number of bytes transmitted and received) shown in this report were last reset to zero. Note that variables that might affect routing (for example, load and reliability) are not cleared when the counters are cleared.
Output queue, Input Queue, drops	Number of packets in output and input queues. Each number is followed by a slash, the maximum size of the queue, and the number of packets dropped due to a full queue.
Five minute input rate, Five minute output rate	The average number of bytes and packets transmitted per second in the last five minutes.
packets input	The total number of error-free packets received by the system.
Received ... broadcasts	The total number of broadcast or multicast packets received by the interface.
runts	The number of packets which are discarded because they are smaller than the medium's minimum packet size. For instance, any Ethernet packet which is less than 64 bytes is considered a runt.
giants	The number of packets which are discarded because they exceed the medium's maximum packet size. For example, any Ethernet packet which is greater than 1,518 bytes is considered a giant.
input error	Includes runts, giants, no buffer, CRC, frame, overrun, and ignored counts. Other input-related errors can also cause the input errors count to be increased, and some datagrams may have more than one error; therefore, this sum may not balance with the sum of enumerated input error counts.
CRC	The Cyclic Redundancy Checksum generated by the originating LAN station or far-end device does not match the checksum calculated from the data received. On a LAN, this usually indicates noise or transmission problems on the LAN interface or the LAN bus itself. A high number of CRCs is usually the result of collisions or a station transmitting bad data.

frame	The number of packets received incorrectly having a CRC error and a noninteger number of octets. On a LAN, this is usually the result of collisions or a malfunctioning Ethernet device.
overrun	The number of times the receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeded the receiver's ability to handle the data.
ignored	The number of received packets ignored by the interface because the interface hardware ran low on internal buffers. These buffers are different than the system buffers mentioned previously in the buffer description. Broadcast storms and bursts of noise can cause the ignored count to be increased.
packets output	Total number of messages transmitted by the system.
bytes	Total number of bytes, including data and MAC encapsulation, transmitted by the system.
underruns	Number of times that the transmitter has been running faster than the router can handle. This may never happen (be reported) on some interfaces.
output errors	The sum of all errors which prevented the final transmission of datagrams out of the interface being examined. Note that this may not balance with the sum of the enumerated output errors, as some datagrams may have more than one error, and others may have errors that do not fall into any of the specifically tabulated categories.
collisions	The number of messages retransmitted due to an Ethernet collision. This is usually the result of an overextended LAN (Ethernet or transceiver cable too long, more than two repeaters between stations, or too many cascaded multiport transceivers). A packet that collides is counted only once in output packets.

interface resets	The number of times an interface has been completely reset. This can happen if packets queued for transmission were not sent within several seconds' time. On a serial line, this can be caused by a malfunctioning modem which is not supplying the transmit clock signal, or by a cable problem. If the system notices that the carrier detect line of a serial interface is up, but the line protocol is down, it periodically resets the interface in an effort to restart it. Interface resets can also occur when an interface is looped back or shut down.
restarts	The number of times a Type 2 Ethernet controller was restarted because of errors.

Debugging the Ethernet Interface

Use the command **debug broadcast** to debug MAC broadcast packets. Enter this command at the EXEC prompt.

debug broadcast

Use the **undebug broadcast** command to turn off messaging.

Use the command **debug packet** to enable a log of packets that the network is unable to classify. Examples of this are packets with unknown link type, or IP packets with an unrecognized protocol field. Enter this command at the EXEC prompt.

debug packet

Use the **undebug packet** command to turn off messaging.

Token Ring Interface Support

Support for the Token Ring interface is supplied on one of Cisco Systems' Token Ring network interface cards:

- The Cisco Systems Token Ring Card (CSC-R) provides interconnection of Cisco network servers to IEEE 802.5 and IBM-compatible Token Ring media.
- The Cisco Systems 4/16 Mbps Token Ring Card (CSC-R16) provides interconnection of Cisco network servers to IEEE 802.5 and IBM-compatible Token Ring media at speeds of 16 or 4 megabits per second.

The Cisco Token Ring interface supports both routing (Level 3 switching) and source-route bridging (Level 2 switching). The use of routing and bridging is on a per-protocol basis. For example, IP traffic could be routed while SNA traffic is bridged. The routing support interacts correctly with source-route bridges.

Specifying a Token Ring Interface

To configure a Token Ring interface, use this configuration command:

interface tokenring *unit*

Specify the card number with the argument *unit*.

Follow this command with the bridging interface subcommands for your particular protocol or application as described in the chapters in Part Five.

Example:

This command begins configuration on the first Token Ring interface.

```
interface tokenring 0
```

Token Ring Encapsulation Methods

Cisco's Token Ring interface, by default, uses the SNAP encapsulation format defined in RFC 1042. It is not necessary to define an encapsulation method for this interface.

Maintaining the Token Ring Interface

Use the command **clear interface** to reset the hardware logic on an interface. Enter this command at the EXEC prompt:

clear interface *type unit*

The arguments *type* and *unit* specify a particular interface type and its unit number. In this case, the argument *type* is **tokenring**.

Note: Under normal circumstances, you do not need to clear the hardware logic on interfaces.

Monitoring the Token Ring Interface

Use the command **show interface** to display information about the Token Ring interface and the state of source bridging. Enter this command at the EXEC prompt:

show interface [*type unit*]

The argument *type* is the keyword **tokenring** and *unit* is the interface unit number. If you do not provide values for the parameters *type* and *unit*, the command will display statistics for all the network interfaces. Sample output of this command is provided below. Table 6-4 describes the fields seen.

```
TokenRing 0 is up, line protocol is up
  Hardware is 16/4 Token Ring, address is 5500.2000.dc27 (bia
0000.3000.072b)
  Internet address is 150.136.230.203, subnet mask is 255.255.255.0
  MTU 8136 bytes, BW 16000 Kbit, DLY 630 usec, rely 255/255, load 1/255
  Encapsulation SNAP, loopback not set, keepalive set (10 sec)
  ARP type: SNAP, ARP Timeout 4:00:00
  Ring speed: 16 Mbps
  Single ring node, Source Route Bridge capable
  Group Address: 0x00000000, Functional Address: 0x60840000
  Last input 0:00:01, output 0:00:01, output hang never
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 0 bits/sec, 0 packets/sec
     16339 packets input, 1496515 bytes, 0 no buffer
     Received 9895 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     32648 packets output, 9738303 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets, 0 restarts
     5 transitions
```

Table 6-4 Show Token Ring Interface Field Descriptions

Field	Description
Token Ring is up | down	The interface is currently active and inserted into ring (up) or inactive and not inserted (down).
Token Ring is Reset	Hardware error has occurred.
Token Ring is Initializing	Hardware is up, in the process of inserting the ring.
Token Ring is Administratively Down	Hardware has been taken down by an administrator.
line protocol is {up | down | administratively down}	Tells whether the software processes that handle the line protocol believe the interface is usable (are keepalives successful?).
Hardware	Specifies the hardware type (Token Ring or 16/4 Token Ring) and provides the address.
Internet address	Lists the Internet address followed by subnet mask.
Encapsulation	Encapsulation method assigned to interface.
loopback	Tells whether loopback is set or not.
keepalive	Tells whether keepalives are set or not.
ARP type:	Type of Address Resolution Protocol assigned.
Ring speed:	Speed of Token Ring — 4 or 16 Mbps.
{Single ring | multiring node}	Indicates whether a node is enabled to collect and use source routing information (RIF) for routable Token Ring protocols.

Group Address:	The interface's group address, if any. The group address is a multicast address; any number of interfaces on the ring may share the same group address. Each interface may have at most one group address.
Last input	The number of hours, minutes, and seconds since the last packet was successfully received by an interface. Useful for knowing when a dead interface failed.
output hang	The number of hours, minutes, and seconds (or never) since the interface was last reset because of a transmission that took too long. When the number of hours in any of the "last" fields exceeds 24 hours, the number of days and hours is printed. If that field overflows, asterisks are printed.
Output queue, Input Queue, drops	Number of packets in output and input queues. Each number is followed by a slash, the maximum size of the queue, and the number of packets dropped due to a full queue.
Five minute input rate, Five minute output rate	The average number of bytes and packets transmitted per second in the last five minutes.
packets input	The total number of error-free packets received by the system.
broadcasts	The total number of broadcast or multicast packets received by the interface.
runts	The number of packets which are discarded because they are smaller than the medium's minimum packet size.
giants	The number of packets which are discarded because they exceed the medium's maximum packet size.
CRC	The Cyclic Redundancy Checksum generated by the originating LAN station or far-end device does not match the checksum calculated from the data received. On a LAN, this usually indicates noise or transmission problems on the LAN interface or the LAN bus itself. A high number of CRCs is usually the result of a station transmitting bad data.
frame	The number of packets received incorrectly having a CRC error and a noninteger number of octets.

overrun	The number of times the serial receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeded the receiver's ability to handle the data.
ignored	The number of received packets ignored by the interface because the interface hardware ran low on internal buffers. These buffers are different than the system buffers mentioned previously in the buffer description. Broadcast storms and bursts of noise can cause the ignored count to be increased.
packets output	Total number of messages transmitted by the system.
bytes output	Total number of bytes, including data and MAC encapsulation, transmitted by the system.
underruns	Number of times that the far-end transmitter has been running faster than the near-end router's receiver can handle. This may never happen (be reported) on some interfaces.
output errors	The sum of all errors which prevented the final transmission of datagrams out of the interface being examined. Note that this may not balance with the sum of the enumerated output errors, as some datagrams may have more than one error, and others may have errors that do not fall into any of the specifically tabulated categories.
collisions	Since a Token Ring cannot have collisions, this statistic is nonzero only if an unusual event occurred when frames were being queued or dequeued by the system software.
interface resets	The number of times an interface has been reset. The interface may be reset by the administrator or automatically when an internal error occurs.
restarts	Should always be zero for Token Ring interfaces.
transitions	The number of times the ring made a transition from up to down, or vice versa. A large number of transitions indicates a problem with the ring or the interface.

Debugging the Token Ring Interface

Use the EXEC command **debug token-ring** to display messages about the Token Ring interface activity. This command reports several lines of information for each packet sent or received and is intended for low traffic, detailed debugging.

debug token-ring

The Token Ring interface records detailed information regarding the current state of the ring. These messages are only displayed when **debug token-events** is enabled.

Enter the **undebug token-ring** and **undebug token-events** commands to turn off the messages.

The last ring status message is displayed in the EXEC command **show interfaces** display for a Token Ring interface. Table 6-5 describes the messages displayed by this command.

Table 6-5 Debug Token Ring Messages

Message	Description
Signal Loss	The controller detected loss of signal on the interface. Several situations can cause this to happen, but the most likely is that another station has just inserted, causing a disruption in service that is reported as signal loss.
Hard Error	This error indicates a significant problem that is preventing transmission of data. There may be a break in the physical cabling or an inserted interface may have died. This message is displayed when the interface is either transmitting or receiving beacon frames.
Soft Error	The interface has detected an aberration on the ring and is transmitting a Report Error MAC frame. These frames are used to report the following types of errors: • Line Error (code violation, token code violation, CRC violation) • Burst Error • MAC AC Set Error • Lost Frame Error • Frame Copied • Receiver Congestion • Token Error These errors are described more fully in the IEEE 802.5 standard.

Ring Beacon	The interface is transmitting beacon frames onto the ring. Something is wrong with the ring.
Wire Fault	The interface has detected an open or short circuit in the lobe data path. The data path starts at the edge of the chipset, and includes the Token Ring transition cable and any other cabling connection on the Multistation Access Unit.
HW Removal	The interface has detected an internal hardware error and has removed itself from the ring.
Remote Removal	The interface received a Remove Ring Station MAC frame from another station on the ring. The interface has removed itself from the ring.
Counter Overflow	Indicates an internal counter is close to reaching its maximum value. The Token Ring monitor firmware automatically reads and clears this condition.
Only Station	The interface has detected that it is the only interface connected and inserted on the ring.
Ring Recovery	The interface is either transmitting or receiving Claim Token MAC frames. This condition is cleared when an Active Monitor has been determined and it transmits a Ring Purge MAC frame.

FDDI Support

FDDI is an ANSI-defined standard for timed, 100-Mbps token-passing over fiber-optic cable. Support for the Fiber Distributed Data Interface (FDDI) is supplied on Cisco Systems' FDDI interface (CSC-FCI) card. Cisco's implementation of FDDI complies with Version 6.1 of the X3T9.5 FDDI specification, offering a Class A dual attach interface that supports the fault recovery methods of the Dual Attach Stations (DASs).

An FDDI network consists of two counter token-passing fiber optic rings. On most networks, the primary ring is used for data communication and the secondary ring is used as a hot standby. The FDDI standard sets a 200 kilometers total fiber length for multimode fiber, and 10 kilometers for single-mode fiber, both of which are supported by Cisco's FDDI interface controller. (The maximum circumference of the FDDI network is only half the specified kilometers because of the *wrapping*, or the looping back of the signal, that occurs during fault isolation.) The FDDI standard allows a maximum of 500 stations with a maximum distance between active stations of two kilometers. The FDDI frame can contain a minimum of 76 bytes and a maximum of 4500 bytes.

Specifying an FDDI

To specify an FDDI, use the following configuration command:

interface fddi *unit*

Specify the interface number with the argument *unit*.

Follow this command with the routing or bridging interface subcommands for your particular protocol or application as described in the chapters in Part Four and Part Five.

FDDI Encapsulation Methods

Cisco's FDDI interface, by default, uses the SNAP encapsulation format defined in RFC 1042. It is not necessary to define an encapsulation method for this interface.

Monitoring the FDDI

The EXEC command **show interfaces** displays information about the FDDI interface, and its use is recommended when configuring the interface.

What follows is a sample of a partial display of FDDI-specific data from the **show interfaces** command output. Table 6-6 describes the fields displayed.

```
Fddi 0 is up, line protocol is up
Hardware type is cBus Fddi, hardware address is AA00.0400.6510
Internet address is 131.111.21.6, subnet mask is 255.255.255.0
MTU 4470 bytes, BW 100000 Kbit, DLY 100 usec, rely 255/255, load 1/255
Encapsulation is SNAP, loopback is not set, keepalive is not set (10
sec.)
ARP type: SNAP
Phy-A state is active, neighbor is B, cmt signal bits 08/20C, status ALS
Brk 1, Con 1, Tra 0, Nxt 11, Sig 10, Join 1, Vfy 1, Act 1
Phy-B state is active, neighbor is A, cmt signal bits 20C/08, status ILS
Brk 1, Con 1, Tra 0, Nxt 11, Sig 10, Join 1, Vfy 1, Act 1
CFM is wrap A, token rotation 5000 usec, ring operational 0:01:42
Upstream neighbor 0800.2008.C52E, downstream neighbor 0800.2008.C52E
Last input 0:00:24, output 0:00:15, output hang never
  Output queue 0/25, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 1 bits/sec, 0 packets/sec
  2725 packets input, 166225 bytes, 0 no buffer
  Received 2725 broadcasts, 0 runts, 0 giants
    2 input errors, 2 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
   5184 packets output, 316224 bytes
    0 output errors, 0 collisions, 1 interface resets, 0 restarts
   4 transitions
```

Table 6-6 Show FDDI Interface Field Descriptions

Field	Description
Fddi is {up |down} ...is administratively down	Tells whether the interface hardware is currently active and can transmit and receive, and if it's been taken down by an administrator.
line protocol is {up | down | administratively down}	Tells whether the interface hardware is currently active and can transmit and receive, or if it's been taken down by an administrator.
Hardware	Provides the hardware type followed by the hardware address.
Internet address	Lists the Internet address followed by subnet mask.
Encapsulation	Encapsulation method assigned to interface.
loopback	Tells whether loopback is set or not.
keepalive	Tells whether keepalives are set or not.
ARP type:	Type of Address Resolution Protocol assigned.
Phy-{A | B}	Lists the state the Physical A or Physical B connection is in, one of: off, active, trace, connect, next, signal, join, verify, or break.
neighbor	State of the neighbor: •A—Indicates that the CMT process has established a connection with its neighbor. The bits received during the CMT signaling process indicate that the neighbor is a Physical A type dual attachment station or concentrator that attaches to the primary ring IN and the secondary ring OUT when attaching to the dual ring. •S—Indicates that the CMT process has established a connection with its neighbor and that the bits received during the CMT signaling process indicate that the neighbor is one Physical type in a single attached station (SAS). •B—Indicates that the CMT process has established a connection with its neighbor and that the bits received during the CMT signaling process indicate that our neighbor is a Physical B dual attached station or concentrator that attaches to the secondary ring IN and the primary ring OUT when attaching to the dual ring. •M—Indicates that the CMT process has established a connection with its neighbor and that the bits received during the CMT signaling process indicate that the router's neighbor is a Physical M type concentrator that serves as a Master to a connected station or concentrator.

	•unk—Indicates that the Cisco network server has not completed the CMT process, and as a result, does not know about its neighbor. See the section "Setting Bit Control" for an explanation of the bit patterns.
cmt signal bits	Shows the transmitted/received CMT bits. The transmitted bits are 0x008 for a Physical A type and 0x20C for Physical B type. The number after the slash (/) is the received signal bits. If the connection is not active, the received bits are zero (0); see the line beginning Phy-B in the above display.
status	The status value displayed is the actual status on the fiber. The FDDI standard defines the following values: • LSU—Line State Unknown, the criteria for entering or remaining in any other line state have not been met. • NLS—Noise Line State is entered upon the occurrence of 16 potential noise events without satisfying the criteria for entry into another line state. • MLS—Master Line State is entered upon the reception of eight or nine consecutive HQ or QH symbol pairs. • ILS—Idle Line State is entered upon receipt of four or five idle symbols. • HLS—Halt Line State is entered upon the receipt of 16 or 17 consecutive H symbols. • QLS—Quiet Line State is entered upon the receipt of 16 or 17 consecutive Q symbols or when carrier detect goes low. • ALS—Active Line State is entered upon receipt of a JK symbol pair when carrier detect is high. • OVUF—Elasticity buffer Overflow/Underflow. The normal states for a connected Physical type is ILS or ALS. If the report displays the QLS status, this indicates that the fiber is disconnected from Physical B, or that it is not connected to another Physical type, or that the other station is not running.
Off	Indicates the CMT is not running on the Physical Sublayer. The state will be off if the interface has been shutdown or if the **cmt disconnect** command has been issued for Physical A or Physical B.
Brk	The Break State is the entry point in the start of a PCM connection.
Tra	The Trace State localizes a stuck beacon condition.
Con	The Connect State is used to synchronize the ends of the connection for the signaling sequence.

Nxt	The Next State separates the signaling performed in the Signal State, and transmits Protocol Data Units (PDUs) while MAC Local Loop is performed.
Sig	The Signal State is entered from the Next State when a bit is ready to be transmitted.
Join	The Join State is the first of three states in a unique sequence of transmitted symbol streams received as line states—the Halt Line State, Master Line State, and Idle Line State, or HLS-MLS-ILS—that leads to an active connection.
Vfy	The Verify State is the second state in the path to the Active State, and will not be reached by a connection that is not synchronized.
Act	The Active State indicates that the CMT process has established communications to its physical neighbor. The transition states are defined in the X3T9.5 specification, and you are referred to the specification for details about these states.
CFM is ...	Contains information about the current state of the MAC connection. The Configuration Management (CFM) state may be one of the following: • isolated—The MAC is not attached to any Physical type. • wrap A—The MAC is attached to Physical A. Data is received on Physical A and transmitted on Physical A. • wrap B—The MAC is attached to Physical B. Data is received on Physical B and transmitted on Physical B. • thru A—The MAC is attached to Physical A and B. Data is received on Physical A and transmitted on Physical B. This is the normal mode for a dual attachment station (DAS) with one MAC. The ring has been operational for 1 minute and 42 seconds.
token rotation	The token rotation value is the default or configured rotation value as determined by the **fddi token rotation-time** command. This value is used by all stations on the ring. The Cisco default is 5,000 microseconds.
ring operational	When the ring is operational, the displayed value will be the negotiated token rotation time of all stations on the ring. Operational times are displayed by the number of hours:minutes:seconds the ring has been up. If the ring is not operational, the message ring not operational is displayed.
Upstream | downstream neighbor	Displays the canonical MAC address of outgoing upstream and downstream neighbors. If the interface is not up, then these values will be zero (0).

Last input	The number of hours, minutes, and seconds since the last packet was successfully received by an interface. Useful for knowing when a dead interface failed.
output hang	The number of hours, minutes, and seconds (or never) since the interface was last reset because of a transmission that took too long. When the number of hours in any of the "last" fields exceeds 24 hours, the number of days and hours is printed. If that field overflows, asterisks are printed.
Output queue, Input Queue, drops	Number of packets in output and input queues. Each number is followed by a slash, the maximum size of the queue, and the number of packets dropped due to a full queue.
Five minute input rate, Five minute output rate	The average number of bytes and packets transmitted per second in the last five minutes.
packets input	The total number of error-free packets received by the system.
broadcasts	The total number of broadcast or multicast packets received by the interface.
runts	The number of packets which are discarded because they are smaller than the medium's minimum packet size.
giants	The number of packets which are discarded because they exceed the medium's maximum packet size.
CRC	The Cyclic Redundancy Checksum generated by the originating LAN station or far-end device does not match the checksum calculated from the data received. On a LAN, this usually indicates noise or transmission problems on the LAN interface or the LAN bus itself. A high number of CRCs is usually the result of collisions or a station transmitting bad data.
frame	The number of packets received incorrectly having a CRC error and a noninteger number of octets. On a LAN, this is usually the result of collisions or a malfunctioning Ethernet device.
overrun	The number of times the serial receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeded the receiver's ability to handle the data.
ignored	The number of received packets ignored by the interface because the interface hardware ran low on internal buffers. These buffers are different than the system buffers mentioned previously in the buffer description. Broadcast storms and bursts of noise can cause the ignored count to be increased.
packets output	Total number of messages transmitted by the system.

bytes output	Total number of bytes, including data and MAC encapsulation, transmitted by the system.
output errors	The sum of all errors which prevented the final transmission of datagrams out of the interface being examined. Note that this may not balance with the sum of the enumerated output errors, as some datagrams may have more than one error, and others may have errors that do not fall into any of the specifically tabulated categories.
collisions	Since an FDDI ring cannot have collisions, this statistic is always zero.
interface resets	The number of times an interface has been reset. The interface may be reset by the administrator or automatically when an internal error occurs.
restarts	Should always be zero for FDDI interfaces.
transitions	The number of times the ring made a transition from ring operational to ring nonoperational, or vice versa. A large number of transitions indicates a problem with the ring or the interface.

The received CMT bits are sometimes helpful if the Cisco network server is having problems establishing a connection to the remote Physical connection.

In the above display, Physical A (`Phy-A`) has completed CMT with its neighbor. The state is active and the display indicates a Physical B type neighbor. The neighbor is determined from the received signal bits, as follows:

Bit Positions	9	8	7	6	5	4	3	2	1	0
Value Received	1	0	0	0	0	0	1	1	0	0

1242

The received value equals 0x20C. Bit positions 1 and 2 (0 1) indicate a Physical B type connection.

The transition states displayed indicate that the CMT process is running and actively trying to establish a connection to the remote physical connection. The CMT process requires state transition with different signals being transmitted and received before moving on to the next state. The ten bits of CMT information are transmitted and received in the Signal State. Each state displays the number of times it was entered. In the above example, the Next State (`Nxt`) was entered 11 times.

Note: This line is not displayed if the FDDI interface has been shut down or if the **cmt disconnect** command has been issued.

The CFM state is `wrap A` in the sample output because the Cisco network server has not completed CMT with its neighbor to connect to Physical B.

The display (or nondisplay) of the Cisco upstream and downstream neighbor does not affect the ability to route data. The determination of the upstream and downstream neighbors is dependent upon all stations on the ring running the same version of Station Management (SMT). Since the Cisco upstream neighbor is also its downstream neighbor in the sample supplied previously, there are only two stations in the ring, the Cisco network server and the router at address *0800.2008.C52E*.

Debugging the FDDI

Use the following EXEC commands to monitor the state of the FDDI interface. (Note that each command has a corresponding **undebug** command that turns off the messages displayed by these commands.)

debug fddi-smt-packets

debug fddi-cmt-events

The **debug fddi-smt-packets** command enables logging of FDDI station management (SMT) packets, whereas the **debug fddi-cmt-events** command enables logging of FDDI Connection Management (CMT) transactions. See the X3T9.5 specification for more information about these packets and transactions.

FDDI Special Commands and Configurations

Using special FDDI interface subcommands you can set the token rotation time, set the transmission valid timer, control the transmission time, set the bit control, and start and stopping FDDI. This section also describes FDDI dual homing—a built-in configuration capability of the Cisco FDDI software.

Setting Token Rotation Time

Use the **fddi token-rotation-time** interface subcommand to control ring scheduling during normal operation, and to detect and recover from serious ring error situations.

fddi token-rotation-time *microseconds*

The argument *microseconds* determines the token rotation time (TRT). The default value is 5,000 microseconds. The FDDI standard restricts the allowed time to be greater than 4,000 microseconds and less than 165,000 microseconds.

As defined in the X3T9.5 specification, the value remaining in the TRT is loaded into the token holding timer (THT). Combining the values of these two timers provides the means to determine the amount of bandwidth available for subsequent transmissions.

Example:

These commands set the rotation time to 24,000 microseconds.

```
interface fddi 0
fddi token-rotation-time 24000
```

Setting the Transmission Valid Timer

Use the **fddi valid-transmission-time** interface subcommand to recover from a transient ring error.

fddi valid-transmission-time *microseconds*

The argument *microseconds* sets the transmission valid timer (TVX) interval. The default valid transmission timer value is 2,500 microseconds.

Example:

These commands change the transmission timer interval to 3,000 microseconds.

```
interface fddi 0
fddi valid-transmission-time 3000
```

Controlling Transmission Time

Use the **fddi tl-min-time** interface subcommand to control the FDDI TL_MIN time (the minimum time to transmit a Physical Sublayer, or PHY line state before advancing to the next Physical Connection Management, or PCM state, as defined by the X3T9.5 specification).

fddi tl-min-time *microseconds*

The specification defines the argument *microseconds* to be a value of 30. This value is used during the Connection Management (CMT) phase to ensure that signals are maintained for at least the value of TL_MIN so the remote station can acquire the signal.

Note: Interoperability tests have shown that some implementations of the FDDI standard need more than 30 microseconds to sense a signal.

Example:

These commands change the TL_MIN time from 30 microseconds to 100 microseconds.

```
interface fddi 0
fddi tl-min-time 100
```

Setting Bit Control

Use the **fddi cmt-signal-bits** interface subcommand to control the information transmitted during the CMT signaling phase.

fddi cmt-signal-bits *signal-bits* **phy-a | phy-b**

The argument *signal-bits* is written as a hexadecimal number preceded by "0x," for example, "0x208."The keywords **phy-a** and **phy-b** select the Physical Sublayer (Physical A or Physical B station), for control of each fiber.

The FDDI standard defines nine bits of signaling information (*signal-bits*) that must be transmitted:

- **bit 0**—Escape bit. Reserved for future assignment by the FDDI standards committee.
- **bits 1 and 2**—Physical type, as follows:

bit 2	bit 1	
0	0	Physical A
1	0	Physical B
0	1	Physical S
1	1	Physical M

1265

Bits 1 and 2 are transmitted (signaled), and each physical type determines if it is allowed to connect to a type at the other end.

- **bit 3**—Physical compatibility. Set if topology rules include the connection of a physical-to-physical type at the end of the connection.
- **bits 4 and 5**—Link Confidence test duration; set as follows:

bit 5	bit 4	
0	0	Short test (default 50 milliseconds)
1	0	Medium test (default 500 milliseconds)
0	1	Long test (default 5 seconds)
1	1	Extended test (default 50 seconds)

1241

- **bit 6**—Media Access Control (MAC) available for link confidence test.
- **bit 7**—Link confidence test failed.
- The setting of bit 7 indicates that the link confidence was failed by the Cisco end of the connection.

- **bit 8**—MAC for local loop.
- **bit 9**—MAC on physical output.

The default signal bits for the **phy-a** and **phy-b** keywords are as follows:

- **phy-a** is set to 0x008 (hexadecimal) or 00 0000 1000 (binary). Bits 1 and 2 are set to 00 to select Physical A. Bit 3 is set to 1 to indicate "accept any connection."
- **phy-b** is set to 0x20c (hexadecimal) or 10 0000 1100 (binary). Bits 1 and 2 are set to 10 to select Physical B. Bit 3 is set to 1 to indicate "accept any connection." Bit 9 is set to 1 to select MAC on output. The normal data flow on FDDI is input on Physical A and output on Physical B.

If the **phy-a** or **phy-b** keyword is not specified, then the signal bits apply to both physical connections.

Note: Use of the **fddi cmt-signal-bits** subcommand is not recommended. This subcommand has been helpful in resolving CMT implementation issues.

Example:

Sun's FDDI version 3.2 implementation requires that bit 9 be set on Physical A. The following example allows the Cisco network server to go into through-mode with Sun's FDDI.

```
interface fddi 0
fddi cmt-signal-bits 0x208 phy-a
```

These commands configure both the Physical A- and B-type connections:

```
interface fddi 0
fddi cmt-signal-bits 0x008 phy-a
fddi cmt-signal-bits 0x20c phy-b
```

Starting and Stopping the FDDI

In normal operation, the FDDI interface is operational once the interface is connected and configured, and is turned off using the **shutdown** interface subcommand described in the section "Shutting Down and Restarting an Interface," earlier in this chapter. The privileged EXEC commands **cmt connect** and **cmt disconnect** allow the operator to start and stop the processes that perform the Connection Management (CMT) function, and particularly, allows the ring on one fiber to be stopped.

The EXEC commands **cmt connect** and **cmt disconnect** are not needed in the normal operation of FDDI; these commands are mainly used in interoperability tests, and are entered at the EXEC prompt:

cmt connect [*interface* [**phy-a** | **phy-b**]]

cmt disconnect [*interface* [**phy-a** | **phy-b**]]

The optional argument *interface* specifies the particular FDDI interface. The optional keywords **phy-a** and **phy-b** specify a Physical Sublayer (A or B).

Examples—Starting FDDI:

The following commands demonstrate use of the **cmt connect** commands for starting the CMT processes on the FDDI ring.

This command starts all FDDI interfaces.

```
cmt connect
```

This command starts both fibers on the FDDI interface unit 0 (zero).

```
cmt connect fddi 0
```

This command starts only Physical Sublayer A on the FDDI interface unit 0 (zero).

```
cmt connect fddi 0 phy-a
```

Examples—Stopping FDDI:

The following commands demonstrates using the **cmt disconnect** command for stopping the CMT processes on the FDDI ring.

This command stops all FDDI interfaces.

```
cmt disconnect
```

This command stops FDDI interfaces unit 0 (zero).

```
cmt disconnect fddi 0
```

This command stops only Physical Sublayer A on the FDDI interface unit 0 (zero). This command causes the FDDI media to go into a wrapped state, so that the ring will be broken.

```
cmt disconnect fddi 0 phy-a
```

FDDI Dual Homing Configuration

Configuration of the FDDI interface is not required for dual homing. The FDDI interface recognizes that it is attached to two M ports on the concentrators, and automatically supports dual homing.

High-Speed Serial Interface (HSSI) Support

The Cisco High-Speed Serial Interface (HSSI) consists of two cards: The CSC-HSCI controller card, which is cBus resident, and the CSC-HSA, which is a back panel applique. The card provides a single full duplex synchronous serial interface capable of transmitting and receiving data at up to 52 megabits per second. The HSSI is a de facto industry standard providing connectivity to T3 (DS-3), E3, SMDS at DS-3, and other high speed wide area services through a DSU or Line Termination Unit.

The high-speed, full duplex synchronous serial interface is supported only on Cisco's modular network server products.

Specifying the HSSI

To specify the HSSI, use this configuration command:

interface hssi *unit*

Specify the interface connector number with the argument *unit*.

Follow this command with the routing or bridging interface subcommands for your particular protocol or application as described in the chapters in Part Four and Part Five.

The cBus cards can query the appliques to determine their types. However, they do so only at system start-up, so the appliques must be attached when the system is started. Issue a **show controller cbus** command to determine how the HSSI card has identified them. The command will also show the capabilities of the card and report controller-related failures.

Example:

This command begins configuration on interface HSSI 0.

```
interface hssi 0
```

HSSI Encapsulation Methods

The HSSI supports the serial encapsulation methods described in the section "Serial Encapsulation Methods" earlier in this chapter.

Maintaining the HSSI

Use the command **clear interface** to reset the hardware logic on an interface. Enter this command at the EXEC prompt:

clear interface *type unit*

The arguments *type* and *unit* specify a particular interface type and its unit number. In this case, the argument *type* is **hssi**.

Note: Under normal circumstances, you do not need clear the hardware logic on interfaces.

Monitoring the HSSI

Use the command **show interfaces** to display information about the serial interface and the state of source bridging. Enter this command at the EXEC prompt:

show interfaces [*type unit*]

Where *type* is the keyword **hssi** and *unit* is the interface unit number. If you do not provide values for the parameters *type* and *unit*, the command will display statistics for all the network interfaces.

Sample output of this command for Cisco's HSSI is provided below. Table 6-7 describes the fields seen.

```
HSSI 0 is up, line protocol is up
  Hardware is cBus HSSI
  Internet address is 150.136.67.190, subnet mask is 255.255.255.0
  MTU 4470 bytes, BW 45045 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input 0:00:03, output 0:00:00, output hang never
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants
              0 parity, 0 rx disabled
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     17 packets output, 994 bytes, 0 underruns
     0 output errors, 0 applique, 4 interface resets, 0 restarts
     2 carrier transitions
```

Table 6-7 Show HSSI Interface Field Descriptions

Field	Description
HSSI is {up |down} ...is administratively down	Tells whether the interface hardware is currently active (whether carrier detect is present) and if it's been taken down by an administrator.
line protocol is {up | down | administratively down}	Tells whether the software processes that handle the line protocol thinks the line is usable (are keepalives successful?).
Hardware	Specifies the hardware type.

Encapsulation	Encapsulation method assigned to interface.
loopback	Tells whether loopback is set, and type of loopback test.
keepalive	Tells whether keepalives are set or not.
Last input	The number of hours, minutes, and seconds since the last packet was successfully received by an interface. Useful for knowing when a dead interface failed.
output hang	The number of hours, minutes, and seconds (or never) since the interface was last reset because of a transmission that took too long. When the number of hours in any of the "last" fields exceeds 24 hours, the number of days and hours is printed. If that field overflows, asterisks are printed.
Last clearing	The time at which the counters that measure cumulative statistics (such as number of bytes transmitted and received) shown in this report were last reset to zero. Note that variables that might affect routing (for example, load and reliability) are not cleared when the counters are cleared.
Output queue, Input Queue, drops	Number of packets in output and input queues. Each number is followed by a slash, the maximum size of the queue, and the number of packets dropped due to a full queue.
Five minute input rate, Five minute output rate	The average number of bytes and packets transmitted per second in the last five minutes.
packets input	The total number of error-free packets received by the system.
broadcasts	The total number of broadcast or multicast packets received by the interface.
runts	The number of packets which are discarded because they are smaller than the medium's minimum packet size.
giants	The number of packets which are discarded because they exceed the medium's maximum packet size.
parity	Report of the parity errors on the HSSI.
rx disabled	Indicates inability to get a buffer when accessing a packet.

CRC	The Cyclic Redundancy Checksum generated by the originating LAN station or far-end device does not match the checksum calculated from the data received. On a LAN, this usually indicates noise or transmission problems on the LAN interface or the LAN bus itself. A high number of CRCs is usually the result of collisions or a station transmitting bad data. On a serial link, CRCs usually indicate noise, gain hits, or other transmission problems on the data link. CRC errors are also reported when a far-end abort occurs, and when the idle flag pattern is corrupted. This makes it possible to get CRC errors even when there is no data traffic.
frame	The number of packets received incorrectly having a CRC error and a noninteger number of octets. On a serial line, this is usually the result of noise or other transmission problems.
overrun	The number of times the serial receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeded the receiver's ability to handle the data.
ignored	The number of received packets ignored by the interface because the interface hardware ran low on internal buffers. These buffers are different than the system buffers mentioned previously in the buffer description. Broadcast storms and bursts of noise can cause the ignored count to be increased.
packets output	Total number of messages transmitted by the system.
bytes output	Total number of bytes, including data and MAC encapsulation, transmitted by the system.
underruns	Number of times that the far-end transmitter has been running faster than the near-end router's receiver can handle. This may never happen (be reported) on some interfaces.
congestion drop	The number of messages discarded because the output queue on an interface grew too long. This can happen on a slow, congested serial link.
output errors	The sum of all errors which prevented the final transmission of datagrams out of the interface being examined. Note that this may not balance with the sum of the enumerated output errors, as some datagrams may have more than one error, and others may have errors that do not fall into any of the specifically tabulated categories.

applique	Indicates an unrecoverable error has occurred on the HSA applique. The system then invokes an interface reset.
interface resets	The number of times an interface has been completely reset. This can happen if packets queued for transmission were not sent within several seconds time. On a serial line, this can be caused by a malfunctioning modem which is not supplying the transmit clock signal, or by a cable problem. If the system notices that the carrier detect line of a serial interface is up, but the line protocol is down, it periodically resets the interface in an effort to restart it. Interface resets can also occur when an interface is looped back or shut down.
restarts	The number of times the controller was restarted because of errors.
carrier transitions	The number of times the carrier detect signal of a serial interface has changed state. Indicates modem or line problems if the carrier detect line is changing state often.

Debugging the HSSI

Use the command **debug serial-interface** to debug HSSI events. Enter this command at the EXEC prompt:

debug serial-interface

Enter the **undebug serial-interface** to turn off messaging.

UltraNet Interface Support

The UltraNet Interface consists of two cards: the CSC-HSCI which is a cBus resident card, and the CSC-ULA, which is a back-panel applique. The UltraNet Interface provides connectivity to the UltraNet product offered by Ultra Network Technologies.

Specifying an UltraNet Interface

To specify an UltraNet interface, use this configuration command:

interface ultranet *unit*

Specify the UltraNet interface connector number with the argument *unit.*

Follow this command with the routing or bridging interface subcommands for your particular protocol or application as described in the chapters in Part Four and Part Five.

Example:

This command begins configuration on UltraNet interface 0.

```
interface ultranet 0
```

UltraNet Encapsulation Method

The UltraNet interface supports the UltraNet encapsulation only. Therefore, there is no encapsulation command for the interface. It will default to UltraNet encapsulation.

Maintaining the UltraNet Interface

Use the command **clear interface** to reset the hardware logic on an interface. Enter this command at the EXEC prompt:

clear interface *type unit*

The arguments *type* and *unit* specify a particular interface type and its unit number. In this case, the argument *type* is **ultranet**.

Note: Under normal circumstances, you do not need to clear the hardware logic on interfaces.

Monitoring the UltraNet Interface

Use the command **show interfaces** to display information about the serial interface and the state of source bridging. Enter this command at the EXEC prompt:

show interfaces [*type unit*]

The argument *type* is the keyword **ultranet** and *unit* is the interface unit number. If you do not provide values for the parameters *type* and *unit*, the command will display statistics for all the network interfaces.

Sample output of this command for Cisco's synchronous serial interfaces is provided below. Table 6-8 describes the fields seen.

```
UltraNet 0 is up, line protocol is up
  Hardware is cBus UltraNet, address is 8/32
  Internet address is 150.136.68.190, subnet mask is 255.255.255.0
  MTU 3500 bytes, BW 125000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
  Encapsulation ULTRANET, loopback not set, keepalive set (10 sec)
  ARP type: ULTRA
  Last input 0:00:09, output 0:00:09, output hang never
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 0 bits/sec, 0 packets/sec
     6 packets input, 236 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants
              0 parity, 0 rx disabled
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     6 packets output, 236 bytes, 0 underruns
     0 output errors, 0 applique, 1 interface resets, 0 restarts
```

Table 6-8 Show UltraNet Interface Field Descriptions

Field	Description
UltraNet is {up | down} ...is administratively down	Tells whether the interface is currently active and if it's been taken down by an administrator.
line protocol is {up | down | administratively down}	Tells whether the processes that handle the line protocol are active.
Hardware	Specifies the hardware type.
Internet Address	Specifies the Internet address.
Encapsulation	Encapsulation method assigned to interface.
loopback	Tells whether loopback is set or not.
keepalive	Tells whether keepalives are set or not.
Last input	The number of hours, minutes, and seconds since the last packet was successfully received by an interface. Useful for knowing when a dead interface failed.
output hang	The number of hours, minutes, and seconds (or never) since the interface was last reset because of a transmission that took too long. When the number of hours in any of the "last" fields exceeds 24 hours, the number of days and hours is printed. If that field overflows, asterisks are printed.
Output queue, Input Queue, drops	Number of packets in output and input queues. Each number is followed by a slash, the maximum size of the queue, and the number of packets dropped due to a full queue.

Five minute input rate, Five minute output rate	The average number of bytes and packets transmitted per second in the last five minutes.
packets input	The total number of error-free packets received by the system.
broadcasts	The total number of broadcast or multicast packets received by the interface.
runts	The number of packets which are discarded because they are smaller than the medium's minimum packet size.
giants	The number of packets which are discarded because they exceed the medium's maximum packet size.
parity	Report of the parity errors on the UltraNet interface.
rx disabled	Indicates inability to get a buffer when accessing a packet.
CRC	The Cyclic Redundancy Checksum generated by the originating LAN station or far-end device does not match the checksum calculated from the data received. On a LAN, this usually indicates noise or transmission problems on the LAN interface or the LAN bus itself. A high number of CRCs is usually the result of collisions or a station transmitting bad data. On a serial link, CRCs usually indicate noise, gain hits, or other transmission problems on the data link.
frame	The number of packets received incorrectly having a CRC error and a noninteger number of octets. On a serial line, this is usually the result of noise or other transmission problems.
overrun	The number of times the serial receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeded the receiver's ability to handle the data.
ignored	The number of received packets ignored by the interface because the interface hardware ran low on internal buffers. These buffers are different than the system buffers mentioned previously in the buffer description. Broadcast storms and bursts of noise can cause the ignored count to be increased.
abort	An illegal sequence of one bits on a serial interface. This usually indicates a clocking problem between the serial interface and the data link equipment.
packets output	Total number of messages transmitted by the system.
bytes output	Total number of bytes, including data and MAC encapsulation, transmitted by the system.

underruns	Number of times that the transmitter has been running faster than the router can handle. This may never happen (be reported) on some interfaces.
output errors	The sum of all errors which prevented the final transmission of datagrams out of the interface being examined. Note that this may not balance with the sum of the enumerated output errors, as some datagrams may have more than one error, and others may have errors that do not fall into any of the specifically tabulated categories.
applique	Indicates an unrecoverable error has occurred on the ULA applique.
interface resets	The number of times an interface has been completely reset. This can happen if packets queued for transmission were not sent within several seconds time. This can be caused by a malfunctioning modem which is not supplying the transmit clock signal, or by a cable problem. If the system notices that the carrier detect line of a serial interface is up, but the line protocol is down, it periodically resets the interface in an effort to restart it. Interface resets can also occur when an interface is looped back or shut down.
restarts	The number of times the controller was restarted because of errors.

UltraNet Special Command

Sometimes it is necessary to statically assign the UltraNet MAC layer address to the interface. This mechanism is needed if dynamic address assignment is not implemented on the Ultra Network Technologies interface on the network. Use the **ultranet address** interface subcommand to assign the MAC layer address of the interface:

ultranet address *ultranet-mac-address*

The argument *ultranet-mac-address* is the MAC address of the UltraNet service line.

Example:

These commands assign address *8/32* to the UltraNet interface.

```
interface ultranet 0
ultranet address 8/32
```

Configuring Dial Backup Service

The dial backup service provides protection against WAN down time by allowing you to configure a backup serial line via a circuit-switched connection.

To configure dial backup, associate a secondary serial interface as a backup to a primary serial interface. This feature requires that an external modem, CSU/DSU device, or ISDN terminal adapter (TA) attached to a circuit-switched service be connected on the secondary serial interface. The external device must be capable of responding to a DTR signal (DTR raised) by auto-dialing a connection to a preconfigured remote site.

The dial backup software keeps the secondary line inactive (DTR low) until one of the following conditions is met:

- The primary line goes down.
- The transmitted traffic load on the primary line exceeds a defined limit.

These conditions are defined using the interface subcommands described later in this section.

When the software detects either a lost Carrier Detect signal from the primary line device, or that the line protocol is down, it causes DTR to be raised on the secondary line, thereby activating it. At that time, the modem, CSU/DSU, or ISDN Terminal Adapter (TA) must be set to dial the remote site. When that connection is made, the routing protocol defined for the serial line will continue the job of transmitting traffic over the dialup line.

You may also configure the dial backup feature to activate the secondary line based upon traffic load on the line.

The software monitors the traffic load and computes a five-minute moving average. If this average exceeds the value you set for the line, then the secondary line is activated, and depending upon how the line is configured, some or all of the traffic will flow onto the secondary dialup primary line.

You may also specify a value that defines when the secondary line should be disabled, and the amount of time the secondary line can take going up or down.

Configuring the Dial Backup Line

Use the **backup interface** interface subcommands to configure the serial interface as a secondary, or dial backup, line. The commands have this syntax:

backup interface *interface-name*

no backup interface *interface-name*

The argument *interface-name* specifies the serial port to be set as the secondary interface line.

Use the **no backup** interface command with the appropriate serial port designation to turn this feature off.

Example:

These commands set serial 1 as the backup line to serial 0.

```
interface serial 0
backup interface serial 1
```

Defining the Traffic Load Threshold

Use the **backup load** interface subcommands to set the traffic load thresholds. The commands have this syntax:

backup load {*enable-threshold* |**never**} {*disable-load* |**never**}

no backup load {*enable-threshold* |**never**} {*disable-load* |**never**}

Enter the arguments *enable-threshold* and *disable-load* using percentage numbers representing the transmitted load as a percentage of the primary line's available bandwidth.

When the transmitted load on the primary line is greater than the value assigned to the *enable-threshold* argument, the secondary line is enabled.

When the transmitted load on the primary line plus the transmitted load on the secondary line is less than the value entered for the *disable-load* argument, then the secondary line is disabled.

If the **never** keyword is used instead of an *enable-threshold* value, the secondary line is never activated due to load. If the **never** keyword is used instead of an *disable-load* value, the secondary line is never deactivated due to load.

By default, no backup loads are defined.

Example:

This example sets the traffic load threshold to 60% on the primary line. When that load is exceeded, the secondary line is activated, and will not be deactivated until the combined load is less than 5% of the primary bandwidth.

```
interface serial 0
backup load 60 5
```

Defining the Backup Line Delay

Use the **backup delay** interface subcommands to define how much time should elapse before a line is set up or taken down. The commands have the following syntax:

backup delay {*enable-delay* |**never**} {*disable-delay* |**never**}

no backup delay {*enable-delay* |**never**} {*disable-delay* |**never**}

The argument *enable-delay* is the delay in seconds after the primary line goes down before the secondary line is activated.

The argument *disable-delay* is the delay in seconds after the primary line goes up before the secondary line is deactivated.

When the primary line goes down, the router delays the amount of seconds defined by the *enable-delay* argument before enabling the secondary line. If, after the delay period, the primary line is still down, the secondary line is activated.

When the primary line comes back up, the router will delay the amount of seconds defined by the *disable-delay* argument. If the *disable-load* condition described above can be satisfied, or if the secondary is never activated due to load, then the secondary line is also deactivated.

Note: In cases where there are spurious signal disruptions that may appear as intermittent lost carrier signals, it is recommended that some delay be enabled before activating and de-activating a secondary.

Use the **never** keyword to prevent any delay on the line being activated or deactivated.

Example:

This example sets a ten-second delay on deactivating the secondary line; however, the line is activated immediately.

```
interface serial 0
backup delay never 10
```

Dial Backup Configuration Examples

This section contains three examples of different dial backup configurations.

Example 1:

The following example configures serial 1 as a secondary line that activates only when the primary line (serial 0) goes down. The secondary line will not be activated due to load of the primary.

```
interface serial 0
backup interface serial 1
backup delay 30 60
```

The secondary line is configured to activate 30 seconds after the primary line goes down and to remain on for 60 seconds after the primary line is reactivated.

Example 2:

The following example configures the secondary line (serial 1) to be activated only when the load of the primary line reaches a certain threshold.

```
interface serial 0
backup interface serial 1
backup load 75 5
```

In this case, the secondary line will not be activated when the primary goes down. The secondary line will be activated when the load on the primary line is greater than 75% of the primary's bandwidth. The secondary line will then be brought down when the aggregate load between the primary and secondary lines fit within 5% of the primary bandwidth.

Example 3:

This example configures the secondary line to activate once the traffic threshold on the primary line exceeds 25%.

```
interface serial 0
backup interface serial 1
backup load 25 5
backup delay 10 60
```

Once the aggregate load of the primary and the secondary lines return to within 5% of the primary bandwidth, the secondary line is deactivated. The secondary line waits 10 seconds after the primary goes down before activating, and remains active for 60 seconds after the primary returns and becomes active again.

Configuring the Point-to-Point Protocol

The Point-to-Point Protocol (PPP), described in RFC 1134, is designed as a method of encapsulating Internet Protocol (IP) datagrams and other Network Layer protocol information over point-to-point links. The document "Point-to-Point Initial Configuration Options" defines the set of options that are negotiated during start up.

Cisco Systems' current implementation of PPP supports no configuration options. The software sends no options and any proposed options are rejected.

Of the possible upper layer protocols, only IP is supported at this time. Thus, the only upper-level protocol that can be sent or received over a point-to-point link using PPP encapsulation is IP. Refer to RFC 1134 for definitions of the codes and protocol states.

The Point-to-Point Protocol is enabled on an interface using the **encapsulation** interface subcommand followed by the **ppp** keyword.

encapsulation ppp

Note: PPP does not support the keepalive timer function (described in Chapter 14, section "Keepalive Timers," in Part Four).

Example:

These commands enable PPP encapsulation on serial interface 0 (zero).

```
interface serial 0
encapsulation ppp
```

Configuring the Null Interface

Cisco provides support for a null interface. This pseudo-interface functions similarly to the null devices available on most operating systems. This interface is always up and can never forward or receive traffic; encapsulation always fails.

The null interface provides an alternative method of filtering traffic. The overhead involved with using access lists can be avoided by directing undesired network traffic to the null interface.

To specify the null interface, specify "null 0" (or "null0") as the interface name and unit. The null interface may be used in any command that has an interface type as a parameter.

Example:

This command configures a null interface for IP route 127.0.0.0.

```
ip route 127.0.0.0 null 0
```

Interface Support Subcommand Summary

Following are alphabetically arranged summaries of the interface subcommands for interface support.

[no] backup delay {*enable-delay* | **never**} {*disable-delay* | **never**}

Defines how much time should elapse before a line is set up or taken down. The argument *enable-delay* is the delay in seconds after the primary line goes down before the secondary line is activated. The argument *disable-delay* is the delay in seconds after the primary line goes up before the secondary line is deactivated. The **never** keyword prevents any delay on the line being activated or deactivated.

[no] backup interface *interface-name*

Configures the serial interface as a secondary, or dial backup, line. The argument *interface-name* specifies the serial port to be set as the secondary interface line. Use the **no backup** interface command with the appropriate serial port designation to turn this feature off.

[no] backup load {*enable-threshold* | **never**} {*disable-load* | **never**}

Sets the traffic load thresholds. The arguments *enable-threshold* and *disable-load* are percentage numbers representing the transmitted load as a percentage of the primary line's available bandwidth. The **never** keyword prevents the secondary line from being activated due to load. By default, no backup loads are defined; the **no** form of the command restores this default.

[no] description *name-string*

Adds a descriptive name to an interface. The argument *name-string* is a comment to be put in the configuration.

encapsulation *encapsulation-type*

Assigns encapsulation method. The *encapsulation-type* argument is a keyword that identifies one of the following serial encapsulation types that Cisco Systems' software supports:

- **arpa**—Ethernet version 2.0 encapsulation
- **bfex25**—Blacker Front End Encryption X.25 operation
- **ddnx25-dce**—DDN X.25 DCE operation
- **ddnx25**—DDN X.25 DTE operation
- **frame-relay**—Frame Relay
- **hdh**—HDH Protocol
- **hdlc**—HDLC Protocol
- **iso1**—IEEE 802.3 encapsulation
- **lapb-dce**—X.25 LAPB DCE operation
- **lapb**—X.25 LAPB DTE operation
- **multi-lapb-dce**—X.25 LAPB multiprotocol DCE operation
- **multi-lapb**—X.25 LAPB multiprotocol DTE operation
- **ppp**—Point-to-Point Protocol (PPP)
- **snap**—IEEE 802.2 Ethernet media
- **smds**—SMDS service
- **x25-dce**—X.25 DCE operation
- **x25**—X.25 DTE operation

fddi cmt-signal-bits *signal-bits* **phy-a | phy-b**

Controls the information transmitted during the CMT signaling phase. The argument *signal-bits* is written as a hexadecimal number preceded by "0x," for example, "0x208." The keywords **phy-a** and **phy-b** select the Physical Sublayer (Physical A or Physical B station), for control of each fiber.

fddi tl-min-time *microseconds*

Controls the FDDI TL_MIN time (the minimum time to transmit a Physical Sublayer, or PHY line state before advancing to the next Physical Connection Management, or PCM state, as defined by the X3T9.5 specification). The specification defines the argument *microseconds* to be a value of 30. This value is used during the Connection Management (CMT) phase to ensure that signals are maintained for at least the value of TL_MIN so the remote station can acquire the signal.

Note: Interoperability tests have shown that some implementations of the FDDI standard need more than 30 microseconds to sense a signal.

---

fddi token-rotation-time *microseconds*

Controls ring scheduling during normal operation, and detects and recovers from serious ring error situations. The argument *microseconds* determines the token rotation time (TRT). The default value is 5,000 microseconds.

fddi valid-transmission-time *microseconds*

Recovers from a transient ring error. The argument *microseconds* sets the transmission valid timer (TVX) interval. The default valid transmission timer value is 2,500 microseconds.

interface *type unit*

Specifies a serial interface. The argument *type* specifies the interface type—**serial, ethernet, tokenring, fddi, or ultranet**—and the argument *unit* specifies the interface number or card number.

[no] shutdown

Disables and enables an interface.

ultranet address *ultranet-mac-address*

Assigns the MAC layer address of the interface. The argument *ultranet-mac-address* is the MAC address of the UltraNet service line.

Interface Support EXEC Command Summary

Following is an alphabetically arranged summary of the EXEC interface support commands.

clear interface *type unit*

Resets the hardware logic on an interface.

cmt connect [*interface* [**phy-a** | **phy-b**]]

Starts the FDDI CMT process.

cmt disconnect [*interface* [**phy-a** | **phy-b**]]

Stops the FDDI CMT process.

[un]debug broadcast

Enables you to log all Level 2 (MAC) broadcast packets received. This information is useful for finding the source of a broadcast storm.

[un]debug fddi-cmt-events

Enables logging of FDDI Connection Management (CMT) transactions.

[un]debug fddi-smt-packets

Enables logging of FDDI station management (SMT) packets.

[un]debug packet

The **debug packet** command enables logging of packets that the network server is unable to classify. Examples of this are packets with an unknown Ethernet link type, or IP packets with an unrecognized protocol field.

[un]debug serial-interface

Enables logging of serial-interface events for network servers equipped with serial network interfaces.

[un]debug token-ring

Enables logging of Token Ring interface activity. This command reports several lines of information for each packet sent or received and is intended for low traffic, detailed debugging.

show controller {**serial** | **token** | **mci** | **cbus** | **fddi**}

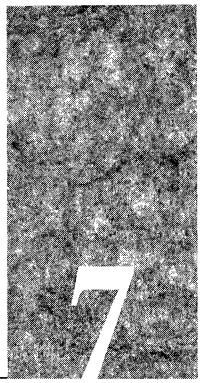
Displays current internal status information for different interface cards.

show interfaces [*type unit*]

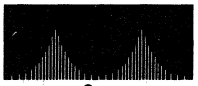
Displays statistics for the network interfaces on the network server. The optional argument *type* can be one of the following: **ethernet**, **serial**, **tokenring**, or **fddi**. The argument *unit* specifies the interface unit or card number.

show version

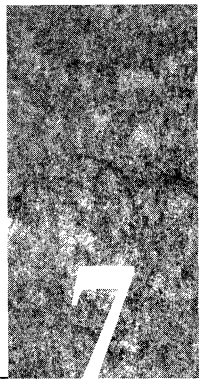
Displays the configuration of the system hardware, the software version, the names and sources of configuration files, and the boot images.



Chapter 7
Adjusting Interface Characteristics



Interface Configuration Subcommand Summary 7-18



Chapter 7
Adjusting Interface Characteristics

This chapter describes how to make adjustments to—or how to *fine tune*—the router interfaces, and how to enable loopback tests. The tasks in this chapter include:

- Switching and scheduling priorities
- Priority output queuing
- Interface hold queues
- Setting the bandwidth
- Setting delay
- Setting loopback testing on an interface

To adjust or test an interface, you must be in the configuration command collection mode. To enter this mode, type the EXEC command **configure** at the EXEC prompt; the **configure** command will place the system into the configuration command collection mode. Once in the command collection mode, start configuring the interface by entering the **interface** command.

A summary of the interface subcommands described in this chapter is included at the end of the chapter.

Adjusting Serial Interface Characteristics

The following sections describe adjustments that can be made to serial interfaces.

Specifying Transmit Delay

Since the MCI and SCI interface cards can send back-to-back data packets over serial interfaces faster than some hosts can receive them, you can specify a minimum dead-time after transmitting a datagram using the following subcommand which is especially useful for serial interfaces. Use the **transmitter-delay** interface subcommands to do this. The full syntax of the command follows.

transmitter-delay *microseconds*

no transmitter-delay

The argument *microseconds* specifies the approximate number of microseconds of minimum delay after transmitting a datagram. The default value is zero; the **no transmitter-delay** command restores this default.

Example:

This command specifies a delay of 300 microseconds on interface serial 0.

```
interface serial 0
transmitter-delay 300
```

For the serial interface on the IGS router and the High Speed Serial Interface (HSSI) on the AGS+, the command is specified as follows:

transmitter-delay *hdlc-flags*

The argument *hdlc-flags* causes a minimum of HDLC flags to be sent between each packet. The maximum number of flags between packets that can be transmitted on the IGS is 62; the minimum is 2.

Configuring DTR Signal Pulsing

The **pulse-time** interface subcommand enables pulsing DTR signals on the Cisco MCI and SCI serial interfaces. The full syntax of the command follows.

pulse-time *seconds*

no pulse-time

The argument *seconds* specifies the interval. When the serial line protocol goes down (for example, because of loss of synchronization) the interface hardware is reset and the DTR signal is held inactive for at least the specified interval. This function is useful for handling encrypting or other similar devices that use the toggling of the DTR signal to resynchronize.

The default interval is zero seconds, which is restored by the **no pulse-time** command.

Example:

This command enables DTR pulse signals for three seconds on interface serial 2.

```
interface serial 2
pulse-time 3
```

Configuring for DCE Appliques

You may configure the clock rate on the serial interface of the SCI and MCI cards to an acceptable bit rate using the **clockrate** interface subcommand. The full syntax is as follows:

clockrate *speed*

no clockrate

The argument *speed* is the desired clock rate, and may be one of any of the rates in Table 7-1.

Table 7-1 Legal Bits per Second Values

1200	2400	4800
9600	19200	34800
56000	64000	72000
125000	148000	500000
800000	1000000	1300000
2000000	4000000	

1269

Be aware that the fastest speeds may not work if your cable is too long, and that speeds faster than 148,000 bits per second are too fast for RS-232 signaling. Cisco recommends you only use the synchronous serial RS-232 signal at speeds up to 64,000 bits per second. To permit a faster speed, use an RS-449 or V.35 applique.

Use the **no clockrate** command to remove the clock rate if you change the interface from a DCE to a DTE device.

Example:

These sample commands set the clock rate on the first serial interface to 64,000 bits per second.

```
interface serial 0
clockrate 64000
```

Adjusting Characteristics That Apply to All Interface Types

The following sections describe adjustments that apply to all interface types.

Configuring Switching and Scheduling Priorities

The normal operation of the network server allows the switching operations to use as much of the central processor as is required. In the event that the network is running unusually heavy loads that do not allow the processor the time to handle the routing protocols, more priority can be given to the system process scheduler using the **scheduler-interval** global configuration command. The full command syntax follows.

scheduler-interval *milliseconds*

no scheduler-interval

The **scheduler-interval** command controls the maximum amount of time that may elapse without running the lowest priority system processes. The minimum interval that may be specified is 500 milliseconds; there is no maximum value. The default is to allow high-priority operations to use as much of the central processor as needed. The **no scheduler-interval** command restores that default.

Example:

This command changes the low-priority process schedule to an interval of 750 milliseconds.

```
scheduler-interval 750
```

Priority Output Queuing

Priority output queuing is a fine tuning mechanism allowing the administrator to set priorities on the type of traffic passing through the network. It is a mechanism for prioritizing datagrams transmitted on an interface. Datagrams are classified according to various criteria, including protocol and subprotocol type, and then queued on one of four output queues.

When the network server is ready to transmit a datagram, it scans the priority queues in order, from highest to lowest, to find the highest priority datagram. After that datagram is completely transmitted, the network server scans the priority queues again. If a priority output queue fills up, datagrams will be dropped and, for protocols such as IP and CLNS, quench indications will be sent to the original transmitter.

Although priority queuing may be enabled for any interface, the intended application was for low bandwidth, congested serial interfaces.

Cisco's priority output queuing mechanism allows traffic control based on protocol or interface type. Commands are also offered to set the size of the queue, and defaults for what happens to packets that are not defined by priority output queue rules.

The priority output queuing mechanism can be used to manage traffic from all networking protocols and from bridges. Additional fine tuning capability is available for IP, and for setting boundaries on the packet size.

Note: Priority queuing introduces extra overhead that is acceptable for slow interfaces, but may not be acceptable for higher speed interfaces such as Ethernet, Token Ring, or FDDI.

There are four priority queues—high, medium, normal, low—listed in order from highest to lowest priority.

Management traffic sourced by the network server is always assigned to the high priority queue to ensure the correct operation of the network. Datagrams that are not classified by the priority list mechanism are assigned to the normal queue.

A priority list is a set of rules that describes how datagrams should be assigned to priority queues. A priority list may also describe a default priority or the queue size limits of the various priority queues.

Assigning Priority by Protocol Type

Use the **priority-list** command to establish queuing priorities based upon the protocol type. The full syntax of this command follows:

priority-list *list* **protocol** *protocol-name queue-keyword* [*args*]

no priority-list *list* **protocol** *protocol-name queue-keyword*

The argument *list* is an arbitrary integer between 1 and 10 that identifies the priority list selected by the user.

The keyword **protocol** is used with the argument *protocol-name* to specify the protocol you are using, and is one of the following: **ip**, **pup**, **chaos**, **xns**, **decnet**, **appletalk**, **clns**, **novell**, **apollo**, **vines**, **stun** (for Serial Tunneling), or **bridge** (for transparent bridging traffic).

The argument *queue-keyword* is a priority queue name, one of **high**, **medium**, **normal**, or **low**.

Optional arguments (*args*) may be specified, depending on the *protocol-name* keyword, as follows.

- **gt** *byte-count*—Specifies a greater-than count. The priority level assigned goes into effect when a packet exceeds the value entered for the argument *byte-count*. The size of the packet must also include additional bytes due to MAC encapsulation on the outgoing interface.
- **lt** *byte-count*—Specifies a less-than count. The priority level assigned goes into effect when a packet size is less than the value entered for *byte-count*. The size of the packet must also include additional bytes due to MAC encapsulation on the outgoing interface.
- **bridge list** *list-number*—Assigns the priority level to bridged traffic according to access list number using the bridge and **list** keywords. The *list-number* argument is the Ethernet-type code access list number assigned by the **access-list** global configuration command and the access-group list interface subcommand.
- **ip list** *list-number*—Assigns traffic priorities according to a specific list. The *list-number* argument is the IP access list number assigned by the access-group list interface subcommand. (For use with the IP protocol, only.)
- **ip tcp** *port*—Assigns the priority level defined to TCP packets originating from or destined to a specified port. (For use with the IP protocol only.) Table 7-2 lists common TCP services and their port numbers.

Table 7-2 Common TCP Services and Their Port Numbers

Service	Port
Telnet	23
SMTP	25
FTP	20, 21

- **ip udp** *port*—Assigns the priority level defined to UDP packets originating from or destined to the specified port. (For use with the IP protocol, only.) The following table lists common UDP services and their port numbers:

Table 7-3 Common UDP Services and Their Port Numbers

Service	Port
TFTP	69
NFS	2049
SNMP	161
RPC	111
DNS	53

Use the **no priority-list** command with the appropriate arguments to remove an entry from the list.

Examples:

This command assigns one as the arbitrary priority list number, specifies DECnet as the protocol type, and assigns a high priority to the datagrams transmitted on this interface:

```
priority-list 1 protocol decnet high
```

When classifying a datagram, the system searches the list of rules specified by **priority-list** commands for a matching protocol type. When a match is found, the datagram is assigned to the appropriate queue. The list is searched in the order it is specified and the first matching rule terminates the search.

This command assigns a medium priority level to every DECnet packet with a size greater than 200 bytes.

```
!
priority-list 2 protocol decnet medium gt 200
!
```

This command assigns a medium priority level to every DECnet packet with a size less than 200 bytes.

```
!
priority-list 4 protocol decnet medium lt 200
!
```

This command assigns a high priority to traffic that matches IP access list 10:

```
!
priority-list 1 protocol ip high list 10
!
```

This command assigns a medium priority level to Telnet packets:

```
!
priority-list 4 protocol ip medium tcp 23
!
```

This command assigns a medium priority level to UDP Domain Name service packets:

```
!
priority-list 4 protocol ip medium udp 53
!
```

This command assigns a high priority to traffic that matches Ethernet type code access list 201.

```
!
priority-list 1 protocol bridge high list 201
!
```

When using multiple rules for a single protocol, remember that the order of the rules matters.

Assigning Priority by Interface Type

Use this variation of the **priority-list** command to establish queuing priorities on packets entering from a given interface (full syntax follows):

priority-list *list* **interface** *interface-name queue-keyword*

no priority-list *list* **interface** *interface-name queue-keyword*

The argument *list* is an arbitrary integer between one and ten that identifies the priority list selected by the user.

The keyword **interface** applies to the priority queuing mechanism to packets arriving from an interface. The argument *interface-name* specifies the name of the interface.

The argument *queue-keyword* is a priority queue name, one of **high**, **medium**, **normal**, or **low**.

Use the **no priority-list** command with the appropriate arguments to remove an entry from the list.

Example:

The following sample command sets any packet type entering on interface Ethernet 2 to a medium priority:

```
!
priority-list 3 interface ethernet 2 medium
!
```

Assigning a Default Priority

Use the following command to assign a priority queue for those datagrams that did not match any other rule in the priority list:

priority-list *list* **default** *queue-keyword*

no priority-list *list* **default**

If no default is specified, or the **no** form of the command is specified, the **normal** queue is assumed.

Specifying the Maximum Packets in the Priority Queues

Use this variation of the **priority-list** command to specify the maximum number of packets that may be waiting in each of the priority queues (full syntax shown):

priority-list *list* **queue-limit** *high-limit medium-limit normal-limit low-limit*

no priority-list *list* **queue-limit**

If a priority queue overflows, excess datagrams are discarded and quench messages may be sent, if appropriate, for the protocol. The default **queue-limit** arguments are 20 datagrams for the **high** priority queue, 40 for **medium**, 60 for **normal**, and 80 for **low**.

The **no priority-list** command with the appropriate keywords and arguments restores these defaults.

Example:

The following is a sample of a priority list, including the access list referenced by one of the priority list rules.

```
priority-list 1 protocol bridge high list 201
priority-list 1 protocol ip medium
priority-list 1 protocol decnet medium
priority-list 1 default low
priority-list 1 queue-limit 20 20 20 10
!
access-list 201 permit  0x6004  0x0000
!
```

Bridged traffic that matches access list 201 is given high priority. Since type code 6004 is the Local Area Transport (LAT), this rule has the effect of making LAT traffic have high priority. IP and DECnet traffic are given medium priority, and everything else is given low priority. In this instance, the network administrator has assigned queue limits of 20 packets for every queue except the low priority queue, which is limited to 10 untransmitted packets.

Assigning a Priority Group to an Interface

Use the **priority-group** interface subcommand to assign the specified priority group to an interface.

priority-group *list*

The argument *list* is the priority list number assigned to the interface.

Example:

This example causes packets exiting interface serial 0 to be classified by priority list 1.

```
interface serial 0
priority-group 1
```

Monitoring the Priority Queuing Lists

When priority queuing is enabled on an interface, the EXEC command **show interfaces** displays information about the input and output queues. This information is a triplet of numbers: the first is the number of packets currently in the queue, the second is the maximum number of packets permitted in the queue, and the third is the number of packets discarded because the queue is full.

```
Input queue: 0/75/4 (size/max/drops); Total output drops: 0
 Output queue: high 0/20/0, medium 0/40/00, normal 0/60/0, low 0/80/0
```

The total `output drops` number is the sum of the discarded numbers from the output queues, plus the count of packets discarded before being prioritized, or because of fast-switching activity.

The **show priority-lists** command gives a brief summary of the rules in the priority lists. Enter this command at the EXEC prompt:

show priority-lists

A sample display follows:

```
Group  Protocol   Priority         List
    1  ip         medium
    1  decnet     medium
    1  bridge     high             201
```

Controlling Interface Hold Queues

Each network interface in a network server has a hold-queue limit. This limit is the number of data packets that the interface can store in its hold queue before rejecting new packets. When the interface empties the hold queue by one or more packets, the interface can accept new packets again.

To specify the hold-queue limit of an interface, use the **hold-queue** interface subcommand. Its full syntax is as follows:

hold-queue *length* {**in** | **out**}

no hold-queue {**in** | **out**}

The argument *length* is the maximum number of packets in the queue. The **in** keyword specifies the input queue; the **out** keyword specifies the output queue.

The **no hold-queue** command with the appropriate keyword restores the default values for an interface. There is no fixed upper limit to a queue size.

The input hold queue prevents a single interface from flooding the network server with too many input packets. Further input packets are discarded if the interface has too many input packets outstanding in the system. The default input hold queue is 75 packets. The default output hold-queue limit is 100 packets. This limit prevents a malfunctioning interface from consuming an excessive amount of memory.

If priority output queueing is being used, the length of the four output queues is set using the **priority-list** configuration command. The **hold-queue** command cannot be used to set an output hold queue length in this situation.

For slow links, use a small output hold-queue limit. This approach prevents storing packets at a rate that exceeds the transmission capability of the link. For fast links, use a large output hold-queue limit. A fast link may be busy for a short time (and thus require the hold queue), but can empty the output hold queue quickly when capacity returns.

To display the current hold queue setting and the number of packets discarded because of hold queue overflows, use the EXEC command **show interfaces**.

Setting Bandwidth

Higher-level protocols may use bandwidth information to make operating decisions. For example, IGRP uses the minimum path bandwidth to determine a routing metric. The TCP protocol adjusts initial retransmission parameters based on the apparent bandwidth of the outgoing interface.

To set a bandwidth value for an interface, use the **bandwidth** interface subcommand. Its full syntax follows:

bandwidth *kilobits*

no bandwidth

The argument *kilobits* specifies the intended bandwidth in kilobits per second. For a full bandwidth DS3, enter the value 45045. Default bandwidth values are set during start up and can be displayed with the EXEC command **show interfaces.**

The **bandwidth** subcommand sets an informational parameter only; you cannot adjust the actual bandwidth of an interface with this subcommand. For some media, such as Ethernet, the bandwidth is fixed; for other media, such as serial lines, you can change the actual bandwidth by adjusting hardware. For both classes of media, you can use the **bandwidth** subcommand to communicate the current bandwidth to the higher-level protocols.

Use the **no bandwidth** command to restore the default values.

Setting Delay

Higher-level protocols may use delay information to make operating decisions. For example, IGRP can use delay information to differentiate between a satellite link and a land link.

To set a delay value for an interface, use the **delay** interface subcommand. The full syntax of this command follows.

delay *tens-of-microseconds*

no delay

The argument *tens-of-microseconds* specifies the delay for an interface or network segment in tens of microseconds. Default delay values may be displayed with the EXEC command **show interfaces**. The **no delay** command restores the default.

Note: The **delay** subcommand sets an informational parameter only; you cannot adjust the actual delay of an interface with this subcommand.

Setting Error Count Reset Frequency

The Cisco interface software provides a mechanism for protection against packet overload and resultant recount errors on the MCI interface cards. This mechanism is set using the **error-threshold** interface subcommand, as follows:

error-threshold *milliseconds*

The argument *milliseconds* is the frequency at which the error recount will be set. The default value is 1000 milliseconds.

Example:

These commands set the error recount threshold on Ethernet interface 2 to 10,000 milliseconds.

```
interface ethernet 2
error-threshold 10000
```

Setting and Adjusting Packet Sizes

Each interface has a default maximum packet size or maximum transmission unit (MTU) size. This number generally defaults to the largest size possible for that type interface. For example, the Ethernet MTU size defaults to 1500 bytes. On serial interfaces, the MTU size varies, but cannot be set smaller than 64 bytes. You can adjust the MTU using the **mtu** interface subcommand. Its full syntax is:

mtu *bytes*

no mtu

The arguments *bytes* is the desired size in bytes. The **no mtu** command restores this value to its original, default value.

Enabling the Loopback Test

The Cisco software provides a loopback test to detect and distinguish equipment malfunctions between line and modem or CSU/DSU (Channel Service Unit/Digital Service Unit) problems on the network server. If correct data transmission is not possible when an interface is in loopback mode, the interface is the source of the problem. The DSU may have similar loopback functions you can use to isolate the problem, if the interface loopback tests passes. If the device does not support local loopback, then this function will have no effect.

You can specify hardware loopback tests on the Cisco Ethernet and serial interfaces, and all Token Ring interfaces except the CSC-R 4 megabit card that are attached to CSU/DSUs, and that support the local loopback signal. The CSU/DSU acts as a DCE device; the Cisco router as a DTE device. The local loopback test generates a CSU loop—a signal that goes through the CSU/DSU to the line, then back through the CSU/DSU to the router.

The **ping** command can also be useful during loopback operation; it is described in the section on "Testing Connectivity with the Ping Command" in Chapter 5.

The following sections describe the various loopback tests available for the supported interfaces.

Note: The loopback tests do not work on the Ethernet interface of the Cisco IGS router product.

Enabling Loopback on the HSSI

The HSSI supports the loopback test from the router to any of the following:

- Applique
- CSU/DSU (DTE side of the CSU/DSU)
- Line side of the CSUDSU
- Remote CSU/DSU

These concepts are illustrated in Figure 7-1.

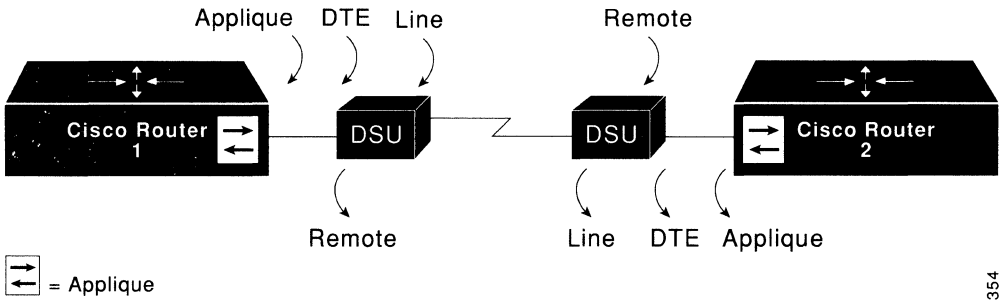


Figure 7-1 HSSI Loopback Test

The commands to enable these tests are described next. In the following commands, the **loopback** command with no argument defaults to the **loopback applique** command.

To start loopback operation, use the **loopback** interface subcommand with the appropriate keyword. To restore an interface in loopback mode to normal operation, use the **no loopback** interface subcommand with the appropriate keyword. Each keyword variation of the **loopback** command is described in the following sections.

Internal Loop to the Applique

Use the following command to configure an internal loop on the HSSI applique. The full syntax of this command follows:

loopback applique

no loopback applique

Once enabled, the command loops the packets on the applique, thereby establishing a loopback inside the Cisco router. This command is useful for "sending pings to yourself" to check functionality of the applique. The HSSI applique (HSA card) uses an internal 44.736 MHz crystal clock during the applique loopback to drive its internal circuits. See the document *HSSI Specification*, available from Cisco Systems, for more information.

This command is functionally equivalent to entering the loopback command with no arguments; however, when the HSCI card is installed, the configuration displayed after the write terminal command is entered will show loopback applique set.

Loopback to the DTE

Use the **loopback dte** command to loop packets to DTE within the CSU/DSU at the DTE interface, when the device supports this function. The full syntax of this command is:

loopback dte

no loopback dte

This command is useful for testing the DTE to DCE cable.

Loopback to the Line

Use the **loopback line** subcommand to loop the packets completely through the CSU/DSU to configure a CSU loop, when the device supports this feature. The full syntax of this command follows:

loopback line

no loopback line

This command is used for testing the DCE device (CSU/DSU) itself.

Loopback to the Remote CSU/DSU

If you want to loop the packets through the CSU/DSU, over the DS3 link and to the remote CSU/DSU and back, use the **loopback remote** subcommand. Its full syntax follows:

loopback remote

no loopback remote

This is applicable only when the device supports the remote function and it is used for testing the data communications channels. The loopback is usually performed at the line port, as opposed to the DTE port, of the remote CSU/DSU.

To show interfaces currently in loopback operation, use the EXEC command **show interfaces.**

HSSI Externally Requested Loopback

The HSA applique (on the HSSI), contains an LED that indicates the LA, LB, and LC signals transiting through the devices. The CSU/DSU uses the LC signal to request a loopback from the router. The CSU/DSU may want to do this so that its own network management diagnostics can independently check the integrity of the connection between the CSU/DSU and the Cisco router.

By default, this feature is enabled on the Cisco router, to support those CSU/DSUs that support this function. To enable and disable an external loopback request on HSSI from the CSU/DSU, use these commands:

hssi external-loop-request

no hssi external-loop-request

When the CSU/DSU asserts the LC signal and the Cisco router enables the external loopback, the connection is blocked by the loopback, and the router no longer has access to the data communication channel. Figure 7-2 illustrates the extent of the signal during an external loopback request.

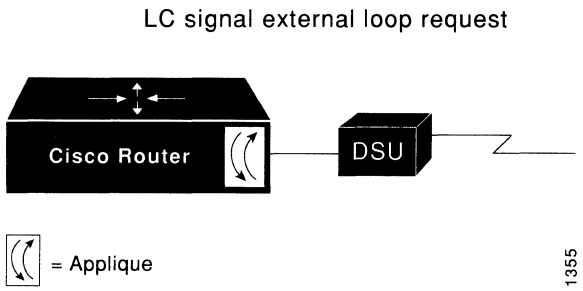


Figure 7-2 HSSI External Loopback Request

If your CSU/DSU does not support this feature, it should be disabled in the Cisco router. This prevents spurious line noise from accidentally tripping the external loopback request line, which would interrupt the normal data flow.

HSCI Card Ribbon Cable Loopback Test

Cisco provides a useful diagnostic that allows fault isolation of possible defects on the HSCI card. This diagnostic is not part of the normal system diagnostics, but is offered to help technicians test for controller defects at installation or when the system is upgraded. The diagnostic involves recabling of the HSCI card and then entering a diagnostic script. The procedures to perform this diagnostic are described in the Cisco publication *Modular Products Hardware Installation and Reference.*

Enabling Loopback on an UltraNet Connection

For users with Ultranet connections, the loopback function is enabled with the **loopback** command (full syntax follows):

loopback

no loopback

Enabling the function places an internal and external loopback on the ULA applique, as seen in the following illustration.

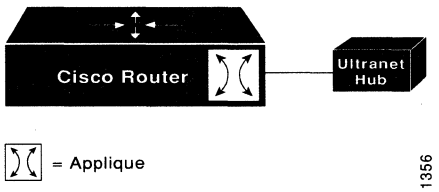


Figure 7-3 UltraNet Loopback Test

Data being transmitted from the router is returned to the router, and incoming data from outside is transmitted back outside.

Enabling Loopback on MCI and SCI Serial Cards

The MCI and SCI serial interface cards support the loopback function when a CSU/DSU or equivalent device is attached to the router. Use the **loopback** command to enable loopback on the interface. The full syntax of this command follows:

loopback

no loopback

The **loopback** interface subcommand loops the packets through the CSU/DSU to configure a CSU loop, when the device supports this feature. The **no loopback** command disables the function.

Enabling Loopback on MCI and MEC Ethernet Cards

Cisco-designed Ethernet interfaces may also be placed into loopback mode. During loopback operation, the interface receives back every packet it sends. Loopback operation has the additional effect of disconnecting network server functionality from the network. Use these commands to enable or disable the loopback test:

loopback

no loopback

Configuring the Ethernet Loopback Server

The Cisco network server provides an Ethernet loopback server that supports Xerox, Intel, and DEC systems specified by the "blue book," a joint specification written by Xerox, Digital Equipment Corporation, and Intel that defines the Ethernet protocol. The loopback server responds to forward data loopback messages sent either to the server's MAC address, or to the broadcast address. Currently, the Ethernet loopback server does not respond to the loopback assistance multicast address.

Use the Ethernet loopback server to test communications between Cisco internetworking products and DEC systems that do not support the IP **ping** command, such as DECnet-only VMS systems.

To originate a loop test on your VMS system with a Cisco server, use the DEC Network Control Program (NCP) command **Loop Circuit.** For more information about the **Loop Circuit** command, consult the DECnet VAX documentation. Cisco network servers support all options that can be specified by the VMS hosts.

Enabling Loopback on the CSC-FCI FDDI Card

The Cisco FDDI (CSC-FCI) may also be placed into loopback mode. During loopback operation, the interface receives back every packet it sends. Loopback operation has the additional effect of disconnecting network server functionality from the network. Use these commands to enable or disable the loopback test:

loopback

no loopback

Enabling Loopback on Token Ring Cards

All of Cisco's Token Ring interface cards (except the 4 megabit CSC-R card) may also be placed into loopback mode. During loopback operation, the interface receives back every packet it sends. Loopback operation has the additional effect of "disconnecting" network server functionality from the network. Use this command to enable the loopback test: Use these commands to enable or disable loopback on the interface:

loopback

no loopback

Interface Configuration Subcommand Summary

This section provides an alphabetical list of all the interface commands described in this chapter.

[no] bandwidth *kilobits*

Sets a bandwidth value for an interface. The argument *kilobits* specifies the intended bandwidth in kilobits per second. Default bandwidth values are set during start up and can be displayed with the EXEC command **show interfaces.** The **no** form of the command restores the default.

[no] clockrate *speed*

Configures the clock rate on the serial interface of the SCI and MCI cards to an acceptable bit rate. The argument *speed* is the desired clock rate in bits per second. The **no** form removes the command from the configuration.

[no] delay *tens-of-microseconds*

Sets a delay value for an interface. The argument *tens-of-microseconds* specifies the delay for an interface or network segment in tens of microseconds. Default delay values may be displayed with the EXEC command **show interfaces**. The **no** form of the command restores the default.

Note: The **delay** subcommand sets an informational parameter only; you cannot adjust the actual delay of an interface with this subcommand.

---

hold-queue *length* {**in** | **out**}
no hold-queue {**in** | **out**}

Specifies the hold-queue limit of an interface. The argument *length* is the maximum number of packets in the queue. The **in** keyword specifies the input queue; the **out** keyword specifies the output queue. The **no** keyword restores the default values for an interface. There is no fixed upper limit to a queue size.

[no] hssi external-loopback-request

Enables and disables an external loopback request on HSSI from the CSU/DSU. When the CSU/DSU asserts the LC signal and the Cisco router enables the external loopback, the connection is blocked by the loopback, and the router no longer has access to the data communication channel.

[no] loopback

On HSSI—Configures an internal loopback loop on the specified interface. This makes the interface loop the packets to the applique, thereby establishing loopback inside the Cisco router. This is the same as the **loopback applique** command.

On UltraNet—Configures a two-way, internal and external loopback on the ULA applique.

On MCI and SCI serial cards—Loops the packets through the CSU/DSU to configure a "CSU loop," when the device supports this feature. This is similar to the **loopback line** command on the HSSI.

On MCI and MEC Ethernet cards—Loops the packets at the interface within the router.

On the CSC-R16 card—Loops the packets at the interface within the router.

The **no** form of the command disables the loopback test.

[no] loopback applique

Configures an internal loop on the HSSI. This makes the interface loop the packets to the applique, thereby establishing loopback inside the Cisco router. This command is useful for sending pings to yourself to check functionality of the applique. For use with HSSI connections only.

[no] loopback dte

Loops packets from the router out to the CSU/DSU and back. For use with HSSI connections only to check connectivity between the router and the CSU/DSU.

[no] loopback line

Loops the packets through the CSU/DSU to configure a CSU loop. Loopback occurs on the line side of the CSU/DSU to test functionality of the CSU/DSU. For use with HSSI connections only.

[no] loopback remote

Loops the packets through the CSU/DSU, over the DS3 link and to the remote CSU/DSU. This is applicable only when the device supports the remote function. For use with HSSI connections only.

[no] priority-list *list* **default** *queue-keyword*

Assigns a priority queue for those datagrams that did not match any other rule in the priority list. If no default or the **no** form of the command is specified, the **normal** queue is assumed.

[**no**] **priority-list** *list* **interface** *interface-name queue-keyword*

Sets up priority queuing on the specified interface. The keyword **interface** applies to the priority queuing mechanism to an interface. The argument *interface-name* specifies the name of the interface. The arguments *list* and *queue-keyword* are described in the section "Assigning Priority by Interface Type." The **no** form of the command removes the item from the list.

[**no**] **priority-list** *list* **protocol** *protocol-name queue-keyword* [*args*]

Sets up priority queuing on the specified interface. The keyword **protocol** applies to the priority queuing mechanism to a protocol. The argument *protocol-name* specifies the name of the protocol. The arguments *list*, *queue-keyword*, and the optional *args* are described in the section "Assigning Priority by Protocol Type." The **no** form of the command removes the item from the list.

[**no**] **priority-list** *list* **queue-limit** *high-limit medium-limit normal-limit low-limit*

Specifies the maximum number of packets that may be waiting in each of the priority queues. If a priority queue overflows, excess datagrams are discarded and quench messages may be sent, if appropriate, for the protocol. The **no** form of the command resets all four queue sizes to their default values as follows: *high-limit* = 20; *medium-limit* = 40; *normal-limit* = 60; *low-limit* = 80.

[**no**] **pulse-time** *seconds*

Enables pulsing DTR signals on the Cisco MCI and SCI serial interfaces for a minimum interval of *seconds*.

[**no**] **scheduler-interval** *milliseconds*

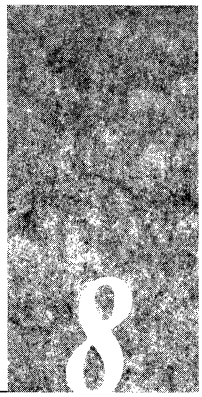
Controls the maximum amount of time that may elapse without running the lowest priority system processes. The minimum interval that may be specified is 500 milliseconds; there is no maximum value. The default is to allow high priority operations to use as much of the central processor as needed. The **no scheduler-interval** command restores that default.

[**no**] **transmitter-delay** *microseconds*

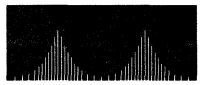
Specifies a minimum dead-time after transmitting a datagram. The argument *microseconds* specifies the approximate number of microseconds of minimum delay after transmitting a datagram. For MCI and SCI serial cards only. The **no** form of the command restores the default value of zero microseconds.

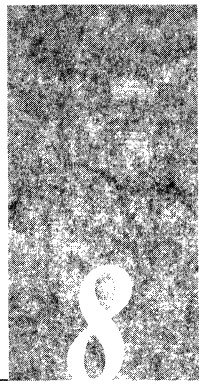
[**no**] **transmitter-delay** *hdlc-flags*

Specify this command syntax for the IGS router and the High Speed Serial Interface (HSSI) connector. The argument *hdlc-flags* causes a minimum of HDLC flags to be sent between each packet. The **no** form of the command restores the default value of zero microseconds.



Chapter 8
Configuring Packet-Switched Software





Chapter 8
Configuring Packet-Switched Software

This chapter describes the configuration tasks for the Cisco packet-switched software. These tasks include configuring the following protocols and services:

- Recommendation X.25, including Link Access Procedure, Balanced (LAPB)
- Frame relay service
- Switched Multi-Megabit Data Services (SMDS)

Summaries of the commands to configure and maintain each service are listed at the end of each section.

Configuring LAPB

It is possible to only use LAPB as a serial encapsulation method. This can be done using a leased serial line. You must use one of the X.25 packet-level encapsulations when attaching to an X.25 network.

Using LAPB under noisy conditions can result in greater throughput than HDLC encapsulation. When LAPB detects a damaged frame, the router immediately retransmits the frame instead of waiting for host timers to expire. This behavior is good only if the host timers are relatively slow. In the case of quickly expiring host timers, however, you will discover that LAPB is spending much of its time retransmitting host retransmissions.

However, if the line is not noisy, the lower overhead of HDLC encapsulation is more efficient than LAPB. When using long delay satellite links, the lock step behavior of LAPB makes the use of HDLC encapsulation the better choice.

The X.25 Recommendation distinguishes between two types of X.25 hosts: data terminal equipment (DTE) in LAPB encapsulation hosts, and data communications equipment (DCE) in LAPB encapsulation hosts.

A router using LAPB encapsulation can act as a DTE or DCE device at the protocol level.

Running a Single Network Protocol

To run datagrams of a single protocol over a serial interface using the LAPB encapsulation, use the interface subcommand:

encapsulation {**lapb** | **lapb-dce**}

The keyword **lapb** sets DTE operation; the keyword **lapb-dce** sets DCE operation. One end of the link must be DTE and the other must be DCE. By default, the single protocol is IP.

To configure another protocol, use the interface subcommand:

lapb protocol *keyword*

Possible protocol keywords include **ip**, **xns**, **decnet**, **appletalk**, **vines**, **clns**, **novell**, and **apollo**.

Running Multiple Network Protocols

To enable use of multiple network protocols on the same line at the same time, use the keyword **multi-lapb** or **multi-lapb-dce** for DTE or DCE operation, respectively:

encapsulation {**multi-lapb** | **multi-lapb-dce**}

For example, with the **multi-lapb** or **multi-lapb-dce** keyword, you can use IP, DECnet, and XNS at the same time. Both ends of the line must use the same encapsulation: either **lapb** or **multi-lapb**. One end of each line must be DCE.

Sample Configuration of LAPB Encapsulation

In the following example of LAPB encapsulation configuration, the frame size (N1), window size (K), hold timer (TH), and maximum retransmission (N2) parameters retain their default values. The **encapsulation** subcommand sets DCE operation for IP packets only, and the **lapb t1** subcommand sets the retransmission timer to 4,000 milliseconds (4 seconds).

Example:

```
interface serial 3
encapsulation lapb-dce
lapb t1 4000
```

For more information on LAPB parameters, see the next section, "Setting the X.25 Level 2 (LAPB) Parameters."

Setting the X.25 Level 2 (LAPB) Parameters

X.25 Level 2, or LAPB (Link Access Procedure, Balanced), is a data encapsulation protocol that operates at Level 2 (the data link level) of the OSI reference model. LAPB specifies methods for exchanging data (in units called *frames*), detecting out-of-sequence or missing frames, retransmitting frames, and acknowledging frames.

LAPB parameters are set with the **lapb** interface subcommand. The interface must be running with either a LAPB or X.25 encapsulation method specified by the **encapsulation** interface subcommand. This subcommand takes two required arguments, *parameter* and *value*. The argument *parameter* is one of several keywords described in the following text, and the argument *value* is a decimal number representing a period of time, a bit count, or a frame count, depending on parameter. Table 8-1 summarizes the LAPB parameters.

Table 8-1 LAPB Parameters

Parameter	Value	Value Range	Default
k	*frames*	1-7	7
n1	*bits*	1-16384	12000
n2	*times*	1-255	20
t1	*milliseconds*	1-64000	3000

1246

Note: The LAPB TH value is not included as a configurable parameter; the value is always 0.

Setting the Retransmission Timer

The retransmission timer determines how long a transmitted frame can remain unacknowledged before the router polls for an acknowledgment. To set the limit for the retransmission timer (the LAPB T1 parameter), use this subcommand:

lapb t1 *milliseconds*

The argument *milliseconds* is the number of milliseconds from 1 through 64000. The default value is 3,000 milliseconds.

For X.25 networks, the router retransmission timer setting should match that of the network. Mismatched retransmission timers can cause excessive retransmissions and an effective loss of bandwidth.

For leased-line circuits, the retransmission timer setting is critical. The timer setting must be large enough to permit several maximum-sized frames to complete one round trip on the link. If the timer setting is too small, the router will poll before the acknowledgment frame can return, which results in an effective loss of bandwidth. If the timer setting is too large, the router waits longer than necessary before requesting an acknowledgment, which also reduces bandwidth.

To determine an optimal value for the retransmission timer, use the privileged EXEC command **ping** to measure the round-trip time of a maximum-sized frame on the link. Multiply this time by a safety factor that takes into account the speed of the link, the link quality, and the distance. A typical safety factor is four. Choosing a larger safety factor can result in slower data transfer if the line is noisy. However, this disadvantage is minor compared to the excessive retransmissions and effective bandwidth reduction caused by a timer setting that is too small.

Setting Frame Parameters

To specify the maximum number of bits a frame can hold, use the **lapb n1** interface subcommand:

lapb n1 *bits*

The **n1** keyword specifies the maximum number of bits (N1) a frame can hold. The argument *bits* is the number of bits from 1 through 16,384, and must be a multiple of eight. The default value is 12,000 bits (1500 bytes).

When connecting to an X.25 network, use the N1 parameter value set by the network administration, which is the maximum size of an X.25 packet. When using LAPB over leased lines, the N1 parameter should be eight times the MTU.

To specify the maximum number of times an acknowledgment frame can be retransmitted, use the **lapb n2** interface subcommand:

lapb n2 *retries*

The argument *retries* is the retransmission count from 1 through 255. The default value is 20 retransmissions.

To specify the maximum permissible number of outstanding frames, called the *window size*, use the **lapb k** interface subcommand:

lapb k *window-size*

The argument *window-size* is a packet count from one to seven. The default value is seven packets.

Monitoring and Troubleshooting LAPB

To display operation statistics for an interface using LAPB encapsulation, use the EXEC command **show interfaces.**

The following example output shows the state of the LAPB protocol, the current parameter settings, and a count of the different types of frames. Each frame count is displayed in the form sent/received.

```
LAPB state is DISCONNECT, T1 3000, N1 12000, N2 20, K 7,TH 3000
IFRAMEs 12/28 RNRs 0/1 REJs 13/1 SABMs 1/13 FRMRs 3/0 DISCs 0/11
```

For a description of the variable names in the **show interface** output, see the X.25 recommendation.

To debug LAPB problems, you must have a good understanding of the X.25 recommendation.

To enable the logging of all packets received and generated, use the privileged EXEC command **debug lapb**. Note that this command slows down processing considerably on heavily loaded links. The following shows example output:

```
Serial0: LAPB 0 CONNECT (5) IFRAME 0 1
Serial0: LAPB I CONNECT (2) RR 1 (R)
Serial0: LAPB I CONNECT (5) IFRAME P 2 1 (C)
Serial0: LAPB 0 REJSENT (2) REJ P/F 1
Serial0: LAPB I REJSENT (2) DM F (C)
Serial0: LAPB I DISCONNECT (2) SABM (C)
Serial0: LAPB 0 CONNECT (2) UA
.
.
.
Serial0: LAPB T SABMSENT 357964 0
Serial0: LAPB 0 SABMSENT (2) SABM P
```

In the example output, each line represents a LAPB frame entering or exiting the router. The first field shows the interface type and unit number of the interface reporting the frame event. The second field is the protocol that provided the information.

The third field is `I`, `O`, or `T` for "frame input," "frame output," or "T1 timer expired," respectively. The fourth field indicates the state of the protocol when the frame event occurred. In a timer event, the state name is followed by the current timer value and the number of retransmissions.

In a packet input or output event, the state name is followed by the size of the frame in bytes (in parentheses) and the frame type name. The next field is an optional indicator: `P/F`, `P`, or `F`, which stand for "Poll/Final," "Poll," and "Final," respectively. For IFRAME frames only, the next two numbers are the receive and send sequence numbers, respectively. For RR, RNR, and REJ frames, the next number is the receive sequence number. For FRMR frames, the next three numbers are three bytes of error data. The last optional indicator is `(C)` or `(R)` for "command" or "response," respectively.

Configuring X.25

The software for the Cisco network server products supports the 1980 and 1984 Recommendation X.25 published by the French International Telegraph and Telephone Consultative Committee (CCITT). The Recommendations specify connections between data terminal equipment (DTE) and data communications equipment (DCE). The Defense Data Network (DDN) and the International Standards Organization (ISO) specify the use of X.25 protocol for computer communications. Many public and private networks also use Recommendation X.25 as their interface technology.

The X.25 model is a telephone network for computer data communications. To start data communications, one computer system calls another to request a communications session. The called computer system can accept or refuse the call. If the called system accepts the call, the two computer systems can begin transferring data in both directions; either system can terminate the call.

In addition to providing remote terminal access, X.25 networks provide bridging capability using a growing list of protocols—the Internet Protocol (IP), DECnet, XNS, ISO CLNS, AppleTalk, Novell IPX, Banyan VINES, and Apollo Domain.

The following sections provide an overview of the Cisco X.25 implementation, describing the different encapsulation methods supported by X.25, and X.25 as a datagram transport, with special attention to the IP protocol. This is followed by descriptions of the DDN X.25 support provided by the Cisco routers, and descriptions of how to configure X.25 switching and bridging. Finally, the configuration of the X.25 Level 3 parameters, and the X.25 Level 2 facilities are discussed. A summary of the interface subcommands are provided at the end of the section.

Note: The default values provided by the software are sufficient for most X.25 networks; however, some parameters may need to be configured, depending on the network.

Overview of Cisco X.25 Support

Cisco Systems' X.25 support can be used in two different ways:

- As a transport for datagram traffic—This entails encapsulating datagrams of IP, DECnet, AppleTalk, and so forth inside packets on an X.25 virtual circuit. Mappings between X.25 addresses and protocol addresses allow these datagrams to be routed through an X.25 network.
- As an X.25 switch—X.25 calls can be routed based on their X.25 addresses either between serial interfaces on the same router (local switching) or across an IP network to another Cisco router (remote switching).

Remote X.25 switching encapsulates the X.25 packet-level inside a TCP connection, allowing disjoint X.25 networks to be connected via a TCP/IP-based network.

X.25 Encapsulation Methods

This section describes the different encapsulation methods and commands that Cisco supports for commercial and private X.25 networks.

Methods of encapsulation for DDN networks are described in the section "Configuring the Datagram Transport on DDN Networks" later in this chapter.

Configuring X.25 DTE and DCE Operation

A router using X.25 Level 3 encapsulation can act as a DTE or DCE device on general X.25 networks.

To set X.25 DTE operation, use the **encapsulation x25** interface subcommand:

encapsulation x25

To set X.25 DCE operation, use the **encapsulation x25-dce** interface subcommand:

encapsulation x25-dce

Configuring the Datagram Transport on Commercial X.25 Networks

Cisco Systems' X.25 support is most commonly configured as a transport for datagrams across an X.25 network. This is accomplished by first establishing a mapping between protocol addresses (for example, IP or DECnet) and the X.121 addresses of the X.25 network. When datagrams for a particular destination are routed for the first time, a virtual circuit is set up to the appropriate X.121 address. The Call User Data portion of the initial Call Request identifies the protocol of the datagrams being carried by a particular virtual circuit. If multiple protocols are in use, multiple virtual circuits will be opened.

Address Mapping Issues

To transport datagrams using X.25 Level 3, the router must map network-protocol addresses to X.121 addresses and vice versa. For example, Figure 8-1 illustrates Hosts A and B that want to communicate via Routers X and Y, which have an X.25 link between them.

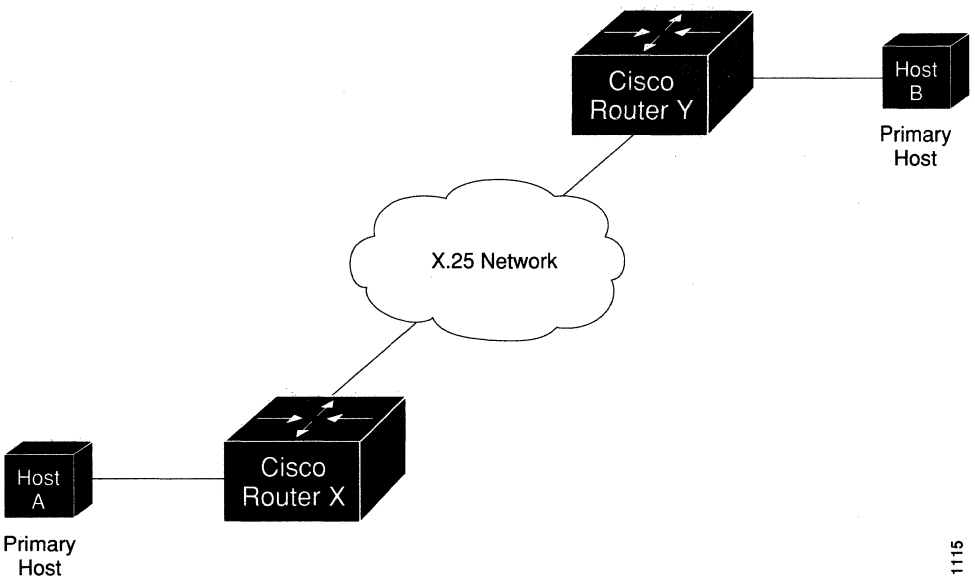


Figure 8-1 Communicating via Routers Through an X.25 Network

To send a packet to Host B, Host A must first send the packet to Router X. From the destination address in the packet and from its routing information, Router X determines that it must send the packet to Router Y over the X.25 network. Router X must then determine the X.121 address for Router Y to open a virtual circuit. Router X uses its network-protocol-to-X.121 address map to convert the protocol address of Router Y to the equivalent X.121 address. Router X can now make the call to create a virtual circuit.

Except in the case of IP using DDN X.25 encapsulation, there is no standard method for dynamically determining these mappings. Instead, static mapping tables must be configured into each router.

To display the network-protocol-to-X.121 address mapping, use the EXEC command **show x25 map.**

Setting Address Mappings

To specify a network-protocol-to-X.121 address mapping such as Internet-to-X.121 or DECnet-to-X.121, use the **x25 map** interface subcommand:

x25 map *protocol-keyword protocol-address X.121-address* [*option1... option6*]

no x25 map *protocol-keyword protocol-address X.121-address*

Note: For bridging over X.25, there is no protocol address; however, the broadcast option is required.

The argument *protocol-keyword* is one of these keywords:

- **ip**—IP
- **decnet**—DECnet
- **chaos**—CHAOSnet
- **xns**—XNS
- **novell**—Novell IPX
- **appletalk**—AppleTalk
- **vines**—VINES
- **apollo**—Apollo Domain
- **pup**—PUP

The *address* arguments specify the network-protocol-to-X.121 mapping.

The *option* arguments add certain features to the mapping specified, and can be any of the following, up to six. They must be specified in the order listed.

- **reverse**—Specifies reverse charging for outgoing calls.
- **accept-reverse**—Causes the router to accept incoming reverse-charged calls. If this option is not present, the router clears reverse charge calls.
- **broadcast**—Causes the router to direct any broadcasts sent through this interface to the specified X.121 address. This option is needed when dynamic routing protocols are being used to access the X.25 network, and is required for bridging X.25.
- **cug** *number*—Specifies a Closed User Group number (from 1 to 99) for the mapping in the outgoing call.
- **nvc** *count*—Sets the number of virtual circuits (VCs) for this protocol/host. The default *count* is the **x25 nvc** setting of the interface. A maximum number of eight VCs can be configured for a single protocol/host.
- **packetsize** *in-size out-size*—Specifies input packet size *in-size* and output packet size *out-size* for the mapping in the outgoing call.
- **windowsize** *in-size out-size*—Specifies input window size *in-size* and output window size *out-size* for the mapping in the outgoing call.
- **throughput** *in out*—Requests the amount of bandwidth through the X.25 network.
- **modulo** *size*—Specifies the maximum window size for this map. The argument *size* permits windows of 8 or 128 on the same interface.
- **nuid** *username password*—Specifies that a network ID facility be sent in the outgoing call with the specified user name and password.

To retract a network-protocol-to-X.121 mapping, use the **no x25 map** interface subcommand with the appropriate network protocol and X.121 address arguments.

Bridging on X.25

Cisco's transparent bridging software supports bridging of X.25 frames. To configure this capability, add this variation of the **x25 map** interface subcommand to the bridging configuration file:

x25 map bridge *X.121-address* **broadcast** [*options-keywords*]

The command specifies Internet-to-X.121 address mapping. The keyword **bridge** specifies bridging over X.25. The argument *X.121-address* is the X.121 address. The keyword **broadcast** is required for bridging X.25 frames. The argument *options-keywords* are the are services that may be added to this map as listed in the section "Setting Address Mappings" earlier in this chapter. For further information about bridging on X.25, and for an example configuration, refer to Chapter 20, "Configuring Transparent Bridging."

Configuring the X.25 Parameters

The Cisco router software provides subcommands to configure the standard Level 2 and Level 3 X.25 parameters and user facilities.

This section discuses the commands and procedures needed to set these parameters, including interface addresses, virtual circuit channel sequence, and addresses.

Setting Permanent Virtual Circuits

Permanent Virtual Circuits (PVCs) are the X.25 equivalent of leased lines; they are never disconnected. To establish a PVC, use the **x25 pvc** interface subcommand:

x25 pvc *circuit protocol-keyword protocol-address*

no x25 pvc *circuit protocol-keyword protocol-address*

The argument *circuit* is a virtual-circuit channel number and it must be less than the lower limit of the incoming call range in the virtual circuit channel sequence.

The argument *protocol-keyword* can be one of these keywords:

- **ip**—IP
- **decnet**—DECnet
- **chaos**—CHAOSnet
- **xns**—XNS
- **novell**—Novell IPX
- **appletalk**—AppleTalk
- **vines**—VINES
- **apollo**—Apollo Domain
- **pup**—PUP
- **bridged**—Bridged

The argument *protocol-address* is that of the host at the other end of the PVC.

To delete a PVC, use the **no x25 pvc** interface subcommand with the appropriate channel number, protocol keyword, and protocol address.

Note: You must specify the required network-protocol-to-X.121 address mapping with an **x25 map** subcommand before you can set up a PVC.

Note: When you are configuring X.25 to use a PVC, you must ensure that no traffic is sent towards the remote router between the time the X.25 map command is issued and the time that the X.25 PVC command is issued. Otherwise, the local router will create an SVC, and then the PVC command will not be allowed.

Note: Map entries with Broadcast attributes are particularly likely to get traffic. The simplest way to ensure that no traffic is sent while configuring is to shut down the interface while configuring it for PVC.

Protocol-to-Virtual-Circuit Mapping

The Call Request packet that sets up a virtual circuit contains a field called the Call User Data field. Typically, the first byte of Call User Data is used by Cisco software to distinguish which high-level protocol will be carried by a particular virtual circuit.

Table 8-2 lists the hexadecimal values of the initial byte of Call User Data and its corresponding network level protocol. The use of 0x81 for ISO 8473 (CLNS) is an ISO standard. The use of 0xCC for Department of Defense IP is defined by RFC 877. The use of C0 00 80 C4 is defined by Banyan. The other values are meaningful only to Cisco software. All the values are padded with three bytes of 0x00, except for VINES, the BFE X.25 encapsulation and CLNS, which follow ISO 8473 requirements.

Table 8-2 Protocols and Initial Byte of Call User Data

Protocol	Initial CUD Byte
ISO CLNS	0x81
DOD IP	0xCC
PUP	0xCE
Chaosnet	0xCF
DECnet	0xD0
XNS	0xD1
AppleTalk	0xD2
Novell	0xD3
Apollo Domain	0xD4
VINES	0xC0 0x00 0x800xC4
Bridges	0xD5

1243

To set the default protocol, use the x25 default command. The full syntax follows:

x25 default *protocol*

no x25 default *protocol*

The **x25 default** command specifies the protocol assumed by the router to interpret calls with unknown Call User Data. The argument *protocol* sets the default protocol and is either **ip** or **pad**. Use this subcommand to change the action taken when an incoming call is received without identifying Call User Data. Normally, the call is cleared. When you use this subcommand, the incoming call is assumed to contain the specified default protocol.

The **no x25 default** subcommand removes the protocol specified.

Displaying Address Mappings

To display the network-protocol-to-X.121 address mappings, enter this command at the EXEC prompt:

show x25 map

The following is a sample output:

```
Serial0: novel 10.0.0c00.7b22 000000220200 PERMANENT, 1 LCN: 4*
Serial0: IP 10.1.0.1 000000010100 CONSTRUCTED
Serial1: appletalk 128.1 000000010000 PERMANENT
Serial1: decnet 28.1 000000020000 PERMANENT BROADCAST
Serial1: ip 128.1.0.3 000000030000 PERMANENT, 2 LCN: 1023*, 1024
```

Each line of output shows the interface name, the protocol type, the protocol address, the X.121 address, and the address-mapping type. The address-mapping types are PERMANENT, entered with the **x25 map** interface subcommand, INTERFACE, indicating the address of a router network interface, and CONSTRUCTED (derived using the DDN address conversion scheme).

If broadcasts are enabled for an address mapping, the word BROADCAST also appears on the output line. Finally, each line also shows the number of Logical Circuit Numbers (LCNs) and, if greater than zero, a list of the LCNs for the protocol/X.121 address. An asterisk marks the current LCN.

Example X.25 Configuration

The following example shows the complete configuration for a serial interface connected to a commercial X.25 PDN for routing the IP protocol. The IP subnetwork address *131.108.9.0* has been assigned for the X.25 network.

Note: When routing IP over X.25, the X.25 network must be treated as a single IP network or subnetwork. Map entries for routers with addresses on subnetworks, other than the one on which the interface's IP address is stored, are ignored by the routing software. Additionally, all routers using the subnet number should have map entries for all others. There are also issues with the broadcast flag, which apply both to IP and to other protocols with dynamic routing.

```
interface serial 2
ip address 131.108.9.1 255.255.255.0
!
encapsulation X25
!
! The "bandwidth" command is not part of the X.25
! configuration; it's especially important to understand that it doesn't
! have any connection with the X.25 entity of the same name. cisco
!"bandwidth" commands are used by IP routing processes (currently only IGRP),
! to determine which lines are the best choices for traffic.
! Since the default is 1544, and X.25 service at that rate isn't generally
! available, most X.25 interfaces that are being used with IGRP in a
! real environment will have "bandwidth" settings.
!
! This is a 9.6 Kbaud line:
!
bandwidth 10
!
```

```
! These Level 3 parameters are defaults; they need to
! match PDN defaults. They are negotiable on a per-call basis:
!
x25 win 7
x25 wout 7
x25 ips 512
x25 ops 512
!
! You must specify an X.25 address, which you get from the PDN:
!
x25 address 31370054065
!
! The following Level 3 parameters have been set to match the network.
! You generally need to change some Level 3 parameters, most often
! those listed below. You may not need to change any Level 2
! parameters, however.
!
x25 hic 32
x25 htc 32
x25 hoc 32
x25 idle 5
x25 nvc 2
!
! The following commands configure the X.25 map. If you want to exchange
! routing updates with any of the routers, they would need "broadcast" flags.
! If the X.25 network is the only path to them, static routes are
! generally used to save on packet charges. If there is a redundant path,
! it might be desirable to run a dynamic routing protocol.
!
x25 map IP 131.108.9.3 31370019134 ACCEPT-REVERSE
! (ACCEPT-REVERSE allows collect calls)
x25 map IP 131.108.9.1 31370054065
x25 map IP 131.108.9.2 31370053087
!
! The PDN cannot handle fast back-to-back packets, so the
!"transmitter-delay" command is used to slow down the MCI card:
!
transmitter-delay 1000
```

Configuring the Datagram Transport on DDN Networks

The DDN X.25 protocol has two versions: Basic Service and Standard Service. Using the DDN X.25 Basic Service, network devices send raw X.25 data across the DDN and assume no structure within the data portion of the X.25 packet. Basic Service users can interoperate only with other Basic Service users.

DDN X.25 Standard Service requires that the X.25 data packets carry IP datagrams. Because the DDN Packet Switch Nodes (PSNs) can extract the Internet packet from within the X.25 packet, they can pass data to either an 1822-speaking-host or to another Standard Service host. Thus, hosts using the older 1822 network interface can interoperate with hosts using Standard Service.

The DDN X.25 Standard is the required protocol for use with DDN PSNs. The Defense Communications Agency (DCA) has certified the Cisco Systems DDN X.25 Standard implementation for attachment to the Defense Data Network.

Enabling DDN X.25

A router using the DDN X.25 Basic Service can act as a DTE or DCE device. To set operation type, use the **encapsulation** interface subcommand:

encapsulation {ddnx25 | ddnx25-dce}

These keywords cause the router to specify the Standard Service facility in the Call Request packet, which notifies the PSNs to use Standard Service.

Using Standard Service, the DDN can provide better service for virtual circuits with higher precedence values. If the router receives an Internet packet with a nonzero Internet precedence field, it opens a new virtual circuit and sets the precedence facility request to the DDN-specified precedence mapping in the Call Request packet. Different virtual circuits are maintained based on the precedence mapping values and the permitted number of virtual circuits.

For situations requiring a high degree of security, the Defense Data Network Blacker Front End Encryption (BFE) device is supported. If the router is attached to such a device, the **bfex25** keyword must be used with the **encapsulation** subcommand:

encapsulation bfex25

This encapsulation provides a mapping from Class A IP addresses to the type of X.121 addresses expected by the BFE encryption device.

DDN X.25 Dynamic Mapping

The DDN X.25 standard implementation includes a scheme for dynamically mapping all classes of Internet addresses to X.121 addresses without a table. This scheme requires that the Internet addresses conform to the formats shown in Table 8-3. These formats segment the Internet addresses into network (N), host (H), logical address (L), and IMP (I) portions. (The acronym IMP, which stands for Interface Message Processor, is the predecessor of PSN, which stands for Packet Switch Node.) For the BFE encapsulation, the Internet address is segmented into Port (P), Domain (D), and BFE ID number (B).

Table 8-3 DDN Internet/X.121 Address Conventions

Class A:	Net.Host.LH.PSN→0000 0 PPPHH00
Bits:	8 8 8 8
Class B:	Net.Net.Host.LH.PSN→0000 0 PPPHH00
Bits:	8 8 8 8
Class C:	Net.Net.Net.Host.PSN→0000 0 PPPHH00
Bits:	8 8 8 4 4
Class BFE:	Net.unused.Port.Domain.BFE →0000 0 PDDDBBB
Bits:	8 1 3 10 10

1266

The DDN conversion scheme uses the host and IMP portions of an Internet address to create the corresponding X.121 address. Strictly speaking, the DDN conversion mechanism is limited to Class A Internet addresses. However, the router can convert Class B and Class C addresses as well. As indicated in Table 8-3, this method uses the last two octets of a Class B address as the host and IMP identifiers, and the upper and lower four bits in the last octet of a Class C address as the host and IMP identifiers, respectively.

The DDN conversion scheme uses a physical address mapping if the host identifier is numerically less than 64. (This limit derives from the fact that a PSN cannot support more than 64 nodes.) If the host identifier is numerically larger than 64, the resulting X.121 address is called a logical address. The DDN does not use logical addresses.

The format of physical DDN X.25/X.121 addresses is ZZZZFIIIHHZZ(SS), where each character represents a digit. ZZZZ represents four zeros, F is zero to indicate a physical address, III represents the IMP (PSN) octet from the Internet address padded with leading zeros, HH is the host octet from the Internet address padded with leading zeros, and ZZ represents two zeros. (SS) represents the optional and unused subaddress.

The physical and logical mappings of the DDN conversion scheme always generate a 12-digit X.121 address. Subaddresses are optional; when added to this scheme, the result is a 14-digit X.121 address. The DDN does not use subaddressing.

Packets using routing and other protocols that require broadcast support can successfully traverse X.25 networks, including the DDN. This traversal requires the use of network-protocol-to-X.121 maps because the router must know explicitly where to deliver broadcast datagrams. (X.25 does not support broadcasts.) You can mark network-protocol-to-X.121 map entries to accept broadcast packets; the router then sends broadcast packets to hosts with marked entries. If you do not specify the address for an interface configured for DDN X.25, the router uses the DDN mapping technique to obtain the X.121 address of an interface.

To display the network-protocol-to-X.121 address mapping, enter this command at the EXEC prompt:

show x25 map

This command is described in more detail later in this chapter. For more information on the BFE addressing, refer to the document *Blacker Interface Control Document (ICD)*, available by copying the file *blacker.doc* from Cisco's *ftp.cisco.com* public directory.

DDN X.25 Configuration Subcommands

Normally the X.25 parameters of a DDN connection are configured using the **x25** and **lapb** interface subcommand described in the section "Configuring the X.25 Parameters," earlier in this chapter. There are a few DDN-specific subcommands, however.

The Cisco X.25 implementation allows you to enable or disable the ability to open a new virtual circuit based on the IP Type of Service (TOS) field.

To do this, use the **x25 ip-precedence** interface subcommand. The full syntax of this command follows:

x25 ip-precedence

no x25 ip-precedence

By default, Cisco routers open one virtual circuit for each type of service. There is a problem associated with this in that some hosts send nonstandard data in the TOS field, thus causing multiple, wasteful virtual circuits to be created. The command **no x25 ip-precedence** causes the TOS field to be ignored when opening virtual circuits.

Using the HDH Protocol

As mentioned earlier, the HDH protocol (also known as the HDLC Distant Host or 1822-J protocol) provides a method for running the 1822 protocol over synchronous serial lines instead of over special-purpose 1822 hardware. HDH packets consist of 1822-LH/DH leaders and data encapsulated in LAPB (X.25 Level 2) format. The HDH hardware is on the Multiport Communications Interface (MCI) card. Strictly speaking, HDH is not X.25, however, it is still commonly used for attaching to the Defense Data Network.

The router supports both the packet and message modes of HDH. It is enabled with the **hdh** interface subcommand with the appropriate keyword. The syntax follows:

hdh {**packet** | **message**}

The **packet** keyword specifies the packet mode; the **message** keyword specifies the message mode.

To enable the HDH protocol, use the interface subcommand:

encapsulation hdh

To enable logging of HDH transactions, use the privileged EXEC command **debug hdh.** To enable logging of the underlying LAPB protocol transactions, use the privileged EXEC command **debug lapb.**

To display information about HDH, use the EXEC command **show imp-hosts.** This command displays information about the hosts with which the network server has communication during the past five minutes via its CSC-A (1822-LH/DH) interfaces or serial interface using HDH (1822-J) encapsulation.

Use the EXEC command **debug psn** to log information about the Packet Switch Node. This command enables logging of noteworthy Packet Switch Node (PSN) events for DDN network servers that are equipped with CSA-A (1822-LH/DH) interfaces or serial interfaces using HDH (1822-J) encapsulation.

The **debug psn-events** EXEC command enables logging of a subset of the PSN and 1822 debugging messages.

Configuring X.25 Switching

In addition to transporting datagrams, the Cisco X.25 software implementation allows switched virtual circuits to be forwarded from one X.25 interface to another and from one Cisco router to another. The forwarding behavior can be controlled based on a locally built table.

The X.25 switching subsystem supports the following facilities and parameters:

- The D-bit ignored but passed through transparently
- Variable length interrupt data
- Flow Control Parameter Negotiation
  - — Window size up to 7
  - — Packet size up to 2048
- The basic Closed User Group
- Throughput class negotiation
- Reverse charging and fast select
- Local facilities are stripped

Higher-level protocols may share an X.25 encapsulated serial interface with the X.25 switching support. The ability to switch or forward X.25 virtual circuits can be done in two different ways:

- Incoming calls received from a local serial interface running X.25 can be forwarded to another local serial interface running X.25. This is known as *local X.25 switching*, as the complete path is handled by the router itself. It does not matter whether the interfaces are configured as DTE or DCE, since software will take the appropriate actions.
- An incoming call can also be forwarded to another Cisco router using the TCP/IP protocols. Upon receipt of an incoming call, a TCP stream connection will be established to the Cisco router which is acting as the switch for the destination. All X.25 packets will be sent and received over this reliable data stream. Flow control is maintained from local DTE to remote DTE. This is known as *remote X.25 switching.*

Running X.25 over TCP/IP provides a number of benefits. The IP datagram containing the X.25 packet can be switched by other routers using the Cisco high-speed switching abilities. It also allows X.25 connections to be sent over networks running only the TCP/IP protocols. The TCP/IP protocol suite runs over many different networking technologies including Ethernet, Token Ring, T1 serial, and FDDI. Thus X.25 data can be forwarded over these media to another Cisco router where it can be output to an X.25 interface.

Enabling X.25 Switching

To enable X.25 switching, use the **x25 routing** global configuration command. The full syntax of this command follows:

x25 routing

no x25 routing

X.25 calls will not be forwarded until this command is issued. The command **no x25 routing** disables the forwarding of X.25 calls.

Constructing the X.25 Routing Table

The X.25 routing table is consulted when an incoming call is received that should be forwarded. The called (destination) X.121 address and Call User Data fields of the X.25 packet are used to determine the route.

An entry in the X.25 routing table is set up or removed with the **x25 rout**e global configuration commands. The full syntax and variations of these commands follows:

x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **interface** *interface-name*
no x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **interface** *interface-name*

x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **ip** *ip-address*
no x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **ip** *ip-address*

x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **alias** *interface-name*
no x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **alias** *interface-name*

The order in which X.25 routing table entries are specified is significant; the list is scanned linearly for the first match. The optional positional parameter (# followed by an integer) can be used to designate after which existing entry to insert or delete the new entry. If no positional parameter is given, the entry is appended to the end of the routing table.

The argument *pattern* is a regular expression that must match the called address and is required. Optional Call User Data corresponding to that X.121 address can be specified as a printable ASCII string. Both the X.121 address and Call User Data can be written using UNIX-style regular expressions. The Call User Data field specifies the data after the protocol identification field, which is four bytes. Use the **show x25 route** command to display the X.25 routing table.

The **alias** keyword permits a way for other X.121 addresses to be treated as local. An alias accepts calls for the router.

Enter the **no x25 route** command with the appropriate arguments and keywords to remove the entry from the table.

Translating X.25 Called Addresses

When interconnecting two separate X.25 networks, it is sometimes necessary to provide for address translation. Cisco's X.25 switch supports translation of X.25 called and calling addresses using the **substitute-dest** or **substitute-source** keyword with the **x25 route** configuration subcommand. Addresses can be rewritten using regular expression replacement.

The **substitute-source** keyword allows substitution of the calling address. For backwards compatibility, the **substitute** keyword will be accepted as **substitute-dest** and written to nonvolatile memory in the new format. When used with the **x25 use-source-address** command, this option allows the calling address to be modified.

x25 route [*#position*]*x121-pattern* [**substitute-source** *rewrite-pattern*] [**substitute-dest** *rewrite-pattern*] [**cud** *pattern*] *interface destination*

For typographical reasons, this command is shown on two lines. When using the optional keywords in this variation of the **x25 route** subcommand, the **substitute-source** keyword must precede the **substitute-dest** keyword, which both must precede the **cud** keyword, and the entire command must be on one line.

The argument *rewrite-pattern* will replace the called or calling X.121 address in routed X.25 packets, as appropriate. The backslash (\) character is treated specially in the argument *rewrite-pattern*; it indicates that the digit immediately following it selects a portion of the original called address to be inserted in the new called address. The characters \0 are replaced with the entire original address. The characters \1 through \9 are replaced with the strings that matched the first through ninth parenthesized parts of *X121-pattern*. See Table 8-4 for a summary of pattern and character matching. A more complete description of the pattern matching characters is found in Appendix D.

Table 8-4 Pattern and Character Matching

Pattern Matching	
\0	Replaces entire original address.
\1..9	Replaces strings that match first through ninth parenthesized part of X.121 address.
*	Matches 0 or more sequences of the regular expressions.
+	Matches 1 or more sequences of the regular expressions.
?	Matches the regular expression of the null string.
Character Matching	
^	Matches the null string at the beginning of the input string.
$	Matches the null string at the end of the input string.
char	Matches *char*.
.	Matches any single character.

1267

Example:

This example indicates that X.25 calls to addresses whose first four Data Network Identification Code (DNIC) digits are 1111 should be routed through interface serial 3, but that the DNIC field in the addresses presented to the equipment connected to that interface should be changed to 2222. The \1 characters in the rewrite pattern indicate that the portion of the original address matched by the characters .* should be inserted in the rewritten address.

```
x25 route ^1111(.*) substitute-dest 2222\1 interface serial 3
```

A more contrived example intended to illustrate the power of the rewriting scheme follows:

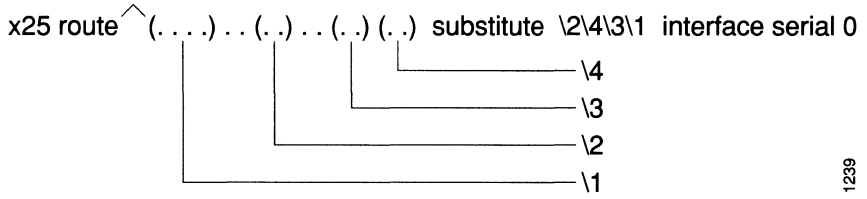


Figure 8-2 X.121 Address Translation Scheme

This sample would cause all X.25 calls with 14-digit called addresses to be routed through interface serial 0. The incoming DNIC field would be moved to the end of the address. The fifth, sixth, ninth, and tenth digits would be deleted, and the thirteenth and fourteenth would be moved before the eleventh and twelfth.

Configuring PVCs on an X.25 Switch

You may configure X.25 Permanent Virtual Circuits (PVCs) in the X.25 switching software. This means that DTEs that require permanent circuits can be connected to the Cisco router acting as an X.25 switch and have a properly functioning connection. X.25 RESETs will be sent indicating when the circuit comes up or goes down.

Use the **x25 pvc** interface subcommand to configure a PVC for a given interface. The syntax of this command follows.

x25 pvc *pvc-number* **interface** *interface-name pvc-number*

The argument *pvc-number* is the PVC number that will be used on the local interface (as defined by the primary interface command). The argument *interface-name* is the interface type and unit number (serial 0, for example), as specified by the **interface** command.

Example:

A PVC is linked across two serial interfaces on the same device. In this type, the alternate interface must be specified along with the PVC number on that interface.To make a working PVC connection, two commands must be specified, each pointing to the other as this example illustrates.

```
interface serial 0
encapsulation x25
x25 pvc 1 interface serial 1 1
interface serial 1
encapsulation x25
x25 pvc 1 interface serial 0 1
```

When you are configuring X.25 to use a PVC, you must ensure that no traffic is sent towards the remote router between the time the **X.25 map** command is issued and the time that **X25 pvc** command is issued. Otherwise, the local router will create an SVC, and then the **pvc** command will not be allowed.

Map entries with the Broadcast attribute are particularly likely to get traffic, due to routing protocol traffic.The simplest way to ensure that no traffic is sent while configuring is to shut down the interface while configuring it for a PVC.

X.25 Switching Configuration Examples

The following examples illustrate how to enable an X.25 switch, how to enable call forwarding, and how to configure a router on a TYMNET/PAD switch to accept and forward calls.

Example 1:

This configuration shows how to enable X.25 switching, as well as how to enter routes into the X.25 routing table.

```
!
! Enable X.25 forwarding
x25 routing
!
! Enter routes into the table. Without a positional parameter, entries
! are appended to the end of the table
x25 route ^100$ interface serial 0
x25 route 100 cud ^pad$ interface serial 2
x25 route 100 interface serial 1
x25 route ^3306 interface serial 3
x25 route .* ip 10.2.0.2
!
```

This routing table forwards calls for X.121 address 100 out the interface serial 0. Otherwise, if the X.121 address contains 100 anywhere within it and contains no Call User Data or the Call User Data is not the string pad, it is forwarded onto serial 1. If the X.121 address contains 100 somewhere within and the Call User Data is the string pad, the call is forwarded onto serial 2. All X.121 addresses that do not match the first three routes are checked for a DNIC of 3306 as the first four digits. If it does match, it is forwarded over serial 3. All other X.121 addresses will match the fifth entry which is a match-all pattern and will have a TCP

connection established to the IP address 10.2.0.2. The Cisco router at 10.2.0.2 will then route the call according to its X.25 routing table.

Example 2:

This configuration configures a Cisco router that sits on a Tymnet PAD/switch to accept calls and have them forwarded to a DEC VAX system. This feature permits running X.25 network over a generalized, already existing IP network, thereby making it unnecessary to get another physical line for one protocol.

The Cisco router positioned next to the DEC VAX system is configured with X.25 routes, as follows:

```
x25 route vax-x121-address interface serial 0
x25 route .* ip cisco-on-tymnet-ipaddress
```

This would route all calls to the DEC VAX X.121 address out to serial 0, where the VAX is connected running PSI. All other X.121 addresses would be forwarded to the *cisco-on-tymnet* address using its IP address. This would take all outgoing calls from the VAX and send them to *cisco-on-tymnet* for further processing.

On the router named *cisco-on-tymnet*, you would enter these commands.

```
x25 route vax-x121-address ip cisco-on-vax
x25 route .* interface serial 0
```

This forces all calls with the VAX X.121 address to be sent to the Cisco router with the VAX connected to it. All other calls with X.121 addresses will be forwarded out to Tymnet. If Tymnet can route them, then a CALL ACCEPTED packet will be returned and everything will proceed normally. If Tymnet can not handle it, it will clear the call and the CLEAR REQUEST packet will be forwarded back toward the VAX.

Setting the X.25 Level 3 Parameters

Once you establish X.25 Level 3 encapsulation, you can set X.25 Level 3 parameters. These parameters are described in the following sections.

Note: If you connect a router to an X.25 network, use the parameters set by the network administration. Also, note that the X.25 Level 2 parameters described in "Setting the X.25 Level 2 (LAPB) Parameters" earlier in this chapter affect X.25 Level 3 operations.

Setting the X.25 Interface Address

To set the X.121 address of a particular network interface, use the **x25 address** subcommand. The address is assigned by the X.25 network:

x25 address *X.121-address*

The argument *X.121-address* is a variable-length X.121 address.

Example:

The following subcommand sets the X.121 address of the current interface to address 2.

```
x25 address 2
```

The value must match that assigned by the X.25 network.

The section "DDN X.25 Dynamic Mapping," earlier in this chapter, describes the addressing scheme for DDN X.25 networks.

Configuring the Virtual Circuit Channel Sequence

An important part of X.25 operation is the virtual circuit channel sequence. This sequence is a range of virtual circuit channel numbers broken into four groups (listed here in numerically increasing order):

1. Permanent virtual circuits
2. Incoming calls
3. Incoming and outgoing (two-way) calls
4. Outgoing calls

Several X.25 parameters determine the numerical ranges of the last three groups; the range for permanent virtual circuits falls numerically below the incoming call range.

X.25 communications devices use the virtual circuit channel sequence when allocating virtual circuits. When initiating a call, these devices must search for an available channel in the sequence in one of two ways. For outgoing calls (made by a DTE device), the search starts at the upper end of the outgoing call range and proceeds in the direction of decreasing channel numbers. The search continues until the device finds an available channel or reaches the lower limit of the two-way call range.

For incoming calls (handled by a DCE device), the search for channel numbers to allocate starts at the lower end of the incoming call range, and proceeds in the direction of increasing channel numbers. The search continues until the device finds an available channel or reaches the upper limit of the two-way call range. The DTE and DCE devices fail to find an available channel only if the overall sequence range is very small or when all of the channels are in use.

To set the upper- and lower-limit parameters of the channel ranges, use the **x25** subcommand keywords listed in Table 8-5. Each keyword takes a channel number as its argument. Note that the values for these parameters must be the same on both ends of an X.25 link.

The default lower limit for all channel ranges on the router is 1; the default upper limit for all ranges is 1024. These defaults reflect a simple approach to allocating the channel ranges: assign the same low value to all lower limits, and assign the same high value to all upper limits. This approach causes the three ranges to overlap and become one large range.

Table 8-5 Range Limit Keywords for the Virtual Circuit Channel Sequence

Keyword	Limit Type	Range	Default
lic	Lower limit, incoming call range	1-4095	1
hic	Upper limit, incoming call range	1-4095	1024
ltc	Lower limit, two-way call range	1-4095	1
htc	Upper limit, two-way call range	1-4095	1024
loc	Lower limit, outgoing call range	1-4095	1
hoc	Upper limit, outgoing call range	1-4095	1024

1247

Setting the Highest Incoming Channel

The **hic** keyword sets the highest incoming channel (HIC).

x25 hic *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

Setting the Highest Outgoing Channel

The **hoc** keyword sets the highest outgoing channel (HOC).

x25 hoc *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

Setting the Highest Two-Way Channel

The **htc** keyword sets the highest two-way channel (HTC).

x25 htc *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

Setting the Lowest Incoming Channel

The **lic** keyword sets the lowest incoming channel (LIC).

x25 lic *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is one.

Setting the Lowest Outgoing Channel

The **loc** keyword sets the lowest outgoing channel (LOC).

x25 loc *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is one.

Setting the Lowest Two-Way Channel

The **ltc** keyword sets the lowest two-way channel (LTC).

x25 ltc *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is one.

Example:

The following commands set these channels.

```
01-20: — Incoming
01-20: — Either incoming or outgoing
01-20: — Outgoing


!
x25 hic 20
x25 htc 20
x25 hoc 20
!
```

Maintaining Virtual Circuits

The router can clear a switched virtual circuit (SVC) after a set period of inactivity. To set this period, use the **x25 idle** interface subcommands:

x25 idle *minutes*

no x25 idle

The argument *minutes* is the number of minutes in the period. The default value is zero, which causes the router to keep the SVC open indefinitely. Both calls originated and terminated by the router are cleared. The **no x25 idle** command returns the default.

To increase throughput across networks, you can establish up to eight SVCs to a host. To specify the maximum number of SVCs that can be open simultaneously to one host, use the **x25 nvc** interface subcommand:

x25 nvc *count*

The argument *count* is a circuit count from 1 to 8; the default is 1.

Note: This affects the default value for the number of SVCs. It does not affect the NVC value for any **x25 map** commands that have already been configured.

Configuring the Ignore VC Timer

Upon receiving a Clear Request for an outstanding Call Request, the X.25 support code immediately tries another Call Request, if it has more traffic to send. This can overrun some X.25 switches. To prevent this problem, use the **x25 hold-vc-timer** configuration commands:

x25 hold-vc-timer *minutes*

no x25 hold-vc-timer

The argument *minutes* is the number of minutes to prevent calls to a previously failed destination. Incoming calls will still be accepted. The default value is 0, and the **no x25 hold-vc-timer** command restores this default.

Configuring the X.25 Level 3 Retransmission Timers

The X.25 Level 3 retransmission timers determine how long the router must wait before retransmitting various Request packets. You can set these timers independently using the **x25** subcommand keywords listed in Table 8-6. Each keyword requires a time value in seconds as its argument. The last column shows the default timer values, in seconds. Four of the timers apply to DTE devices, and the other four apply to DCE devices.

Table 8-6 Retransmission Timer Keywords and Defaults

Keyword (DTE/DCE)	Affected Request Packet	Time(seconds) (DTE/DCE)
t20/t10	Restart Request	180/60
t21/t11	Call Request	200/180
122/t12	Reset Request	180/60
t23/t13	Clear Request	180/60

1249

Setting the DTE Restart Request Retransmission Timer

The **t20** keyword sets the limit for the Restart Request retransmission timer (T20) on DTE devices.

x25 t20 *seconds*

The argument *seconds* is a time value in seconds. The default is 180 seconds.

Setting the DCE Restart Request Retransmission Timer

The **t10** keyword sets the limit for the Restart Request retransmission timer (T10) on DCE devices.

x25 t10 *seconds*

The argument *seconds* is a time value in seconds. The default value is 60 seconds.

Setting the DTE Call Request Retransmission Timer

The **t21** keyword sets the limit for the Call Request retransmission timer (T21) on DTE devices.

x25 t21 *seconds*

The argument *seconds* is a time value in seconds. The default value is 200 seconds.

Setting the DCE Call Request Retransmission Timer

The **t11** keyword sets the limit for the Call Request retransmission timer (T11) on DCE devices.

x25 t11 *seconds*

The argument *seconds* is a time value in seconds. The default value is 180 seconds.

Setting the DTE Reset Request Retransmission Timer

The **t22** keyword sets the limit for the Reset Request retransmission timer (T22) on DTE devices.

x25 t22 *seconds*

The argument *seconds* is a time value in seconds. The default value is 180 seconds.

Setting the DCE Reset Request Retransmission Timer

The **t12** keyword sets the limit for the Reset Request retransmission timer (T12) on DCE devices.

x25 t12 *seconds*

The argument *seconds* is a time value in seconds. The default value is 60 seconds.

Setting the DTE Clear Request Retransmission Timer

The **t23** keyword sets the limit for the Clear Request retransmission timer (T23) on DTE devices.

x25 t23 *seconds*

The argument *seconds* is a time value in seconds. The default value is 180 seconds.

Setting the DCE Clear Request Retransmission Timer

The **t13** keyword sets the limit for the Clear Request retransmission timer (T13) on DCE devices.

x25 t13 *seconds*

The argument *seconds* is a time value in seconds. The default value is 60 seconds.

Updating the X.121 Address

Some X.25 calls, when forwarded by the X.25 switching support, need the calling (source) X.121 address updated to that of the outgoing interface. This is necessary when forwarding calls from private data networks to public data networks.

Outgoing calls forwarded over a specific interface can have their calling X.121 address updated by using the **x25 use-source-address** subcommand. The full syntax is:

x25 use-source-address

no x25 use-source-address

The **no x25 use-source-address** command prevents updating of the source address of outgoing calls.

Setting X.25 Packet Sizes

X.25 networks use maximum input and output packet sizes set by the network administration. You can set the router input and output packet sizes to match those of the network with the **x25** subcommand keywords **ips** and **ops,** respectively:

x25 ips *bytes*

x25 ops *bytes*

The argument *bytes* is a byte count in the range of 128 through 2048. The default value is 128 bytes.

Larger packet sizes are better, because smaller packets require more overhead processing.

Note: Set the **x25 ips** and **x25 ops** keywords to the same value unless your network supports asymmetry between input and output packets.

To send a packet larger than the X.25 packet size over an X.25 virtual circuit, a router must break the packet into two or more X.25 packets with the M-bit ("more data" bit) set. The receiving device collects all packets with the M-bit set and reassembles them.

Setting the Flow Control Modulus

X.25 supports flow control with a sliding window sequence count. The window counter restarts at zero upon reaching the upper limit, which is called the window modulus. To set the window modulus, use the **x25 modulo** interface subcommand:

x25 modulo *modulus*

The argument *modulus* is either 8 or 128. The default value is eight. The value of the modulo parameter must agree with that of the device on the other end of the X.25 link.

Configuring Packet Acknowledgment

To specify upper limits on the number of outstanding unacknowledged packets, use the commands **x25 win** (for input window) and **x25 wout** (for output window):

x25 win *packets*

x25 wout *packets*

The argument *packets* is a packet count. The packet count for **win** and **wout** can range from one to the window modulus. The default value is two packets.

The **x25 win** command determines how many packets the router can receive before sending an X.25 acknowledgment. The **wout** limit determines how many sent packets can remain unacknowledged before the router uses a hold queue. To maintain high bandwidth utilization, assign these limits the largest number that the network allows.

Note: Set **win** and **wout** to the same value unless your network supports asymmetry between input and output window sizes.

You can instruct the router to send acknowledgment packets when it is not busy sending other packets, even if the number of input packets has not reached the **win** count. This approach improves line responsiveness at the expense of bandwidth. To enable this option, use the **x25 th** subcommand:

x25 th *delay*

The argument *delay* must be between zero and the input window size. A value of one sends one Receiver Ready acknowledgment per packet at all times. The default value is zero, which disables the delayed acknowledgment strategy.

The router sends acknowledgment packets when the number of input packets reaches the count you specify, providing there are no other packets to send. For example, if you specify a count of one, the router can send an acknowledgment per input packet.

Suppressing the Calling Address

To omit the calling address in outgoing calls, use the **x25 suppress-calling-address** interface subcommands:

x25 suppress-calling-address

no x25 suppress-calling-address

The **suppress-calling-address** keyword omits the calling (source) X.121 address in Call Request packets. This option is required for networks that expect only subaddresses in the calling address field. The calling address is sent by default.

Use the **no x25 suppress-calling-address** subcommand to reset this subcommand to the default state.

Accepting Reverse Charge Calls

To instruct the router to accept all reverse charge calls, use the **x25 accept-reverse** interface subcommands:

x25 accept-reverse

no x25 accept-reverse

The **accept-reverse** keyword causes the interface to accept reverse charge calls by default. This behavior can also be configured on a per-peer basis using the **x25 map** subcommand.

The **no x25 accept-reverse** command disables this facility.

Forcing Packet-Level Restarts

To force a packet-level restart when the link level is restarted, use the **x25 linkrestart** interface subcommands:

x25 linkrestart

no x25 linkrestart

This command restarts X.25 Level 2 (LAPB) when errors occur. This behavior is the default and is necessary for networks that expect this behavior. Use the **no x25 linkrestart** to turn this feature off.

Setting the Packet Network Carrier

To set the packet network carrier, use the **x25 rpoa** interface subcommands:

x25 rpoa *name number*

no x25 rpoa *name*

The **x25 rpoa** interface subcommand specifies a list of transit RPOAs to use, referenced by name. The argument *name* must be unique with respect to all other RPOA names. It is used in the **x25 facility** and **x25 map** interface subcommands. The argument *number* is a number that is used to describe an RPOA. The **no x25 rpoa** command removes the specified name.

Setting X.25 Parameters on a Per-Call Basis

To override interface settings on a per-call basis, use the **x25 facility** interface subcommand. The full syntax of the command follows:

x25 facility *keyword argument*

no x25 facility *keyword argument*

The command enables X.25 facilities, which are sent between DTE and DCE devices to negotiate certain link parameters.

The argument *keyword* specifies one of the following keywords, followed by the required *argument*:

- **cug** *number*—Specifies a Closed User Group *number* from 1 through 99 to provide an extra measure of network security.
- **packetsize** *in-size out-size*—Sets the size in bytes of input packets *(in-size)* and output packets *(out-size)*. Both values should be the same.
- **reverse**—Reverses charges on all Call Request packets from the interface.
- **windowsize** *in-size out-size*—Sets the packet count for input windows (*in-size*) and output windows (*out-size*). Both values should be the same.
- **throughput** *in out*—Sets the requested throughput values for input and output throughput across the network.

- **rpoa** *name*—Specifies the list of transit Recognized Private Operating Agencies (RPOAs) to use in outgoing Call Request packets.

The **no x25 facility** command with the appropriate keyword and argument removes the facility.

Maintaining X.25

To clear all virtual circuits at once, use the privileged EXEC command **clear x25-vc.** This command takes an interface type keyword (usually **serial**) and a unit number as arguments to identify the interface with which the virtual circuits are associated.

To clear a particular virtual circuit, add the two arguments described above the **clear x25-vc** command, and include a logical circuit number (LCN) value as a third argument. The command syntax is:

clear x25-vc *interface unit* [*lcn*]

The **clear x25-vc** command clears all X.25 virtual circuits at once. The argument *interface* is the interface type. The argument *unit* is the interface unit number. The optional argument *lcn* clears the specified virtual circuit.

Monitoring X.25 Level 3 Operations

The router provides EXEC **show** commands to provide information on interface operation and virtual circuit operation. Use the EXEC command **show interfaces** to display interface parameters and statistics. Use the EXEC command **show x25 vc** to display virtual circuit parameters and statistics.

Displaying Interface Parameters and Statistics

To display parameter information for an interface using X.25 Level 3 protocol, use the EXEC command **show interfaces.** For X.25 interfaces, the following is an example of output:

```
X25 address 000000010100, state R1, modulo 8, idle 0, timer 0, nvc 1
Window size: input 2, output 2, Packet size: input 128, output 128
Timers: T20 180 T21 200 T22 180 T23 180 TH 0
Channels: Incoming 1-1024 Two-way 1-1024 Outgoing 1-1024
RESTARTs 1/20 CALLs 1000+2/1294+190/0+0 DIAGs 0/0
```

On the first line, the `address` field indicates the calling address used in the Call Request packet. The `state` field indicates the state of the interface: `R1` is the normal ready state, `R2` indicates the DTE not-ready state, and `R3` indicates the DCE not-ready state. If the state is `R2` or `R3`, the device is awaiting acknowledgment for a Restart Request packet. The `modulo` field displays the modulo value, which determines the sequence numbering scheme used. The `idle` field shows the number of minutes the router waits before closing idle virtual circuits. The `timer` field displays the value of the interface timer, which is zero unless the interface state is `R2` or `R3`. The `nvc` field displays the maximum number of simultaneous virtual circuits permitted to and from a single host.

On the second line, the `Window size` and `Packet size` fields show the default window

and packet sizes for the interface. Each virtual circuit can override these values using facilities specified with the **x25 map** or **x25 facility** subcommands.

The third line shows the values of the Request packet timers (T10 through T13 for a DCE device, and T20 through T23 for a DTE device). The fourth line shows the channel sequence ranges. The last line shows packet statistics for the interface using these formats:

```
sent/received\ successful+failed\ callssent+callssentfailed/\
callsreceived+callsreceivedfailed/\
callsforwarded+callsforwardedfailed\
```

Displaying Virtual Circuit Parameters and Statistics

The EXEC command **show x25 vc** displays the details of the active X.25 switched virtual circuits. To examine a particular virtual circuit, add an LCN argument to the **show x25 vc** command. The following is example output:

```
LC1: 1, State: D1, Interface: Serial0
Started 0:55:03, last input 0:54:56, output 0:54:56
Connected to IP [10.4.0.32] <->000000320400 Precedent: 0
Window size input: 7, output: 7
Packet size input: 1024, output: 1024
PS: 2 PR: 6 Remote PR: 2 RCNT: 1 RNR: FALSE
Retransmits: 0   Timer (secs): 0  Reassembly (bytes): 0
Held Fragments/Packets: 0/0
Bytes 1111/588 Packets 18/22 Resets 0/0 RNRs 0/0 REJs 0/0 INTs 0/0
```

On the first line, the `LCI` field displays the virtual circuit number. The `State` field displays the state of the virtual circuit (which is independent of the states of other virtual circuits); D1 is the normal ready state. (See the CCITT X.25 recommendation for a description of virtual circuit states.) The `Interface` field shows the interface used for the virtual circuit.

On the second line, the `Started` field shows the time elapsed since the virtual circuit was created, the `last input` field shows time of last input, and the `output` field shows time of last output.

On the third line, the `Connected to` field shows in brackets the network-protocol address and then the X.121 address. The `Precedent` field, which appears only if you have specified DDN encapsulation, indicates IP precedence.

The fourth and fifth lines show window and packet sizes. These sizes can differ from the interface default values if facilities were offered and accepted in the Call Request and Call Accepted packets.

On the sixth line, the `PS` and `PR` fields show the current send and receive sequence numbers, respectively. The `Remote PR` field shows the last `PR` number received from the other end of the circuit. The `RCNT` field shows the count of unacknowledged input packets. The `RNR` field shows the state of the `Receiver Not Ready` flag; this field is true of the network sends a receiver-not-ready packet.

On the seventh line, the `Retransmits` field shows the number of times a packet has been retransmitted. The `Timer` field shows a nonzero time value if a packet has not been acknowledged or if virtual circuits are being timed for inactivity. The `Reassembly` field shows the number of bytes received for a partial packet (a packet in which the more data bit is set).

On the eighth line, the fragments part of the `Held Fragments/Packets` field shows the number of X.25 packets being held. (In this case, `Fragments` refers to the X.25 fragmentation of higher-level data packets.) The Packets part of the `Held Fragments/Packets` field shows the number of higher-level Protocol packets currently being held pending the availability of resources, such as the establishment of the virtual circuit.

On the last line, the `Bytes` field shows the total numbers of bytes sent and received. The `Packets, Resets, RNRs, REJs,` and `INTs` fields show the total sent and received packet counts of the indicated types. (RNR is Receiver Not Ready, REJ is Reject, and INT is Interrupt).

Debugging X.25

When installing an X.25 link, you can use the EXEC commands **debug** with different keywords as follows:

debug x25

This command enables the monitoring of all X.25 traffic.

debug x25-events

This command enables the monitoring of all X.25 traffic but does not display information about X.25 data or acknowledgment packets.

debug x25-vc *number*

This command allows you to watch the specified port *number* with a virtual circuit using the **debug x25** and **debug x25-events** commands.

The following is example output:

```
Serial0: X25 I R1 RESTART (5) 8 lci 0 cause 7 diag 250
Serial0:  X25 0 R1 RESTART CONFIRMATION (3) 8 lci 0
Serial0: X25 0 P2 CALL REQUEST (19) 8 lci 1
From(14): 31250000000101 To(14): 31109090096101
Facilities (0)
Serial0: X25 0 P6 CLEAR REQUEST (5) 8 lci 1 cause diag 122
```

For each event, the first field identifies the interface on which the activity occurred, and the second field indicates that it was an X.25 event. The third field indicates whether the X.25 packet was input (`I`) or output (`O`). The fourth field is the state of the interface: `R1` is the normal ready state, `R2` indicates the DTE not-ready state, and `R3` indicates the DCE not-ready state.

The fifth field is the type of the X.25 packet that triggered the event, and the sixth field (in parentheses) gives the total length of the X.25 packet in bytes. The seventh field is the window modulus. The eighth field (labeled `lci`) shows the virtual circuit number. The ninth field (labeled `cause`) gives the cause code, and the tenth field (labeled `diag`) gives the diagnostic code.

For Call Request and Call Connected packets, the router shows additional information on separate lines. The `From` field shows the calling X.121 address and the `To` field shows the called X.121 address. The number in parentheses after the field name—`(14)` for example—specifies the number of digits in the address. The `Facilities` field indicates the length (in bytes) of the requested facilities and the facilities contents.

X.25 Global Configuration Command Summary

This section provides an alphabetically arranged summary of the X.25 global configuration commands.

[no] x25 route [*# position*]*x121-pattern* [**cud** *pattern*] **interface** *interface-name*
[no] x25 route [*# position*]*x121-pattern* [**cud** *pattern*] **ip** *ip-address*
[no] x25 route [*# position*]*x121-pattern* [**cud** *pattern*] **alias** *interface-name*
[no] x25 route [*#position*]*x121-pattern* [**substitute-source** *rewrite-pattern*]
[**substitute-dest** *rewrite-pattern*] [**cud** *pattern*] *interface destination*

Inserts or removes an entry in the X.25 routing table. The *x121-pattern pattern* parameter is the X.121 address of the called destination and is required. The **alias** keyword permits a way for other X.121 addresses to be treated as local. The **substitute-source** keyword allows substitution of the calling address. The argument *rewrite-pattern* replaces the called or calling X.121 address in routed X.25 packets.

[no] x25 routing

Enables or disables X.25 switching. X.25 calls will not be forwarded until this command is issued. The command **no x25 routing** disables the forwarding of X.25 calls.

X.25 Interface Subcommand Summary

This section provides an alphabetical list of all the interface subcommands used in the X.25 interface.

[no] x25 accept-reverse

Instructs the router to accept all reverse charge calls. This behavior can also be configured on a per-peer basis using the **x25 map** subcommand. The **no** form of the command resets the default state.

x25 address *X.121-address*

Sets the X.121 address of a particular network interface. The address is assigned by the X.25 network. The argument *X.121-address* is a variable-length X.121 address.

[no] x25 default *protocol*

Specifies or removes the protocol assumed by the CPT to interpret calls with unknown Call User Data. The argument *protocol* sets the default protocol and is either **ip** or **pad**.

[no] x25 facility *keyword argument*

Enables and disables the X.25 facilities, which are sent between DTE and DCE devices to negotiate certain link parameters. The argument *keyword* specifies one of the following keywords, followed by the required *argument*:

- **cug** *number*—Specifies a Closed User Group *number* from 1 through 99 to provide an extra measure of network security.
- **packetsize** *in-size out-size*—Sets the size in bytes of input packets *(in-size)* and output packets *(out-size)*. Both values should be the same.
- **reverse**—Reverses charges on all Call Request packets from the interface.
- **windowsize** *in-size out-size*—Sets the packet count for input windows (*in-size*) and output windows (*out-size*). Both values should be the same.
- **throughput** *in out*—Sets the requested throughput values for input and output throughput across the network.
- **rpoa** *name*—Specifies the list of transit RPOAs to use in outgoing Call Request packets.

x25 hic *channel*

Sets the highest incoming channel (HIC). The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

x25 hoc *channel*

Sets the highest outgoing channel (HOC). The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

[no] x25 hold-vc-timer *minutes*

Prevents overruns on X.25 switches for traffic through the VCs. The argument *minutes* is the number of minutes to prevent calls to a previously failed destination. Incoming calls will still be accepted.

x25 htc *channel*

Sets the highest two-way channel (HTC). The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

[no] x25 idle *minutes*

Sets the period of inactivity once an SVC has been cleared. The argument *minutes* is the number of minutes in the period. Both calls originated and terminated by the router are cleared. The default value is 0 (zero), which causes the router to keep the SVC open indefinitely. The **no** variation restores this default.

[no] x25 ip-precedence

Enables or disables the ability to open a new virtual circuit based on the IP Type of Service (TOS) field. By default, Cisco routers open one virtual circuit for each type of service.

x25 ips *bytes*
x25 ops *bytes*

Set the router packet size to match those of the network. The **ips** keyword specifies the router input packet size while the keyword **ops** specifies the router output packet size. The argument *bytes* is a byte count in the range of 128 through 1024. The default value is 128 bytes. Larger packet sizes are better, because smaller packets require more overhead processing.

Note: Set the **x25 ips** and **x25 ops** keywords to the same value unless your network supports asymmetry between input and output packets.

---

x25 lic *channel*

Sets the lowest incoming channel (LIC). The argument *channel* is a channel number from 1 through 4095. The default value is one.

[no] x25 linkrestart

Forces a packet-level restart when the link level is restarted and restarts X.25 Level 2 (LAPB) when errors occur. This behavior is the default and is necessary for networks that expect this behavior. The **no** form of the command resets the default state.

x25 loc *channel*

Sets the lowest outgoing channel (LOC). The argument *channel* is a channel number from 1 through 4095. The default value is one.

x25 ltc *channel*

Sets the lowest two-way channel (LTC). The argument *channel* is a channel number from 1 through 4095. The default value is one.

[**no**] **x25 map** *protocol-keyword protocol-address X.121-address* [*option1 ... option*]

Specifies a network-protocol-to-X.121 address mapping such as Internet-to-X.121 or DECnet-to-X.121. The argument *protocol-keyword* can be one of these protocol types: **ip**, **decnet**, **chaos**, **xns** , **novell**, **appletalk**, **vines**, **apollo**, **pup**. The *address* arguments specify the network-protocol-to-X.121 mapping. The *option* arguments add certain features to the mapping specified, and can be any of the following, up to six. They must be specified in the order listed.

- **reverse**—Specifies reverse charging for outgoing calls.
- **accept-reverse**—Causes the router to accept incoming reverse-charged calls. If this option is not present, the router clears reverse charge calls.
- **broadcast**—Causes the router to direct any broadcasts sent through this interface to the specified X.121 address. This option is needed when dynamic routing protocols are being used to access the X.25 network.
- **cug** *number*—Specifies a Closed User Group number (from 1 to 99) for the mapping in the outgoing call.
- **nvc** *count*—Sets the number of virtual circuits (VCs) for this map/host. The default *count* is the **x25 nvc** setting of the interface. A maximum number of eight VCs can be configured for a single map/host.
- **packetsize** *in-size out-size*—Specifies input packet size *in-size* and output packet size *out-size* for the mapping in the outgoing call.
- **windowsize** *in-size out-size*—Specifies input window size *in-size* and output window size *out-size* for the mapping in the outgoing call.
- **throughput** *in out*—Requests the amount of bandwidth through the X.25 network.
- **modulo** *size*—Specifies the maximum window size for this map. The argument *size* permits windows of 8 or 128 on the same interface.
- **nuid** *username password*—Specifies that a network ID facility be sent in the outgoing call with the specified user name and password.

x25 modulo *modulus*

Sets the modules. The argument *modulus* is either 8 or 128. The default value is eight. The value of the modulo parameter must agree with that of the device on the other end of the X.25 link.

x25 nvc *count*

Specifies the maximum number of switched virtual circuits that can be open simultaneously to one host. The argument *count* is a circuit count from 1 to 8; the default is 1.

[no] x25 pvc *circuit protocol-keyword protocol-address*

Establishes or deletes Permanent Virtual Circuits (PVCs). The argument *circuit* is a virtual circuit channel number and it must be less than the lower limit of the incoming call range in the virtual circuit channel sequence (set using the **lic** keyword). The argument *protocol-keyword* can be one of the protocol types mentioned above. The argument *protocol-address* is that of the host at the other end of the PVC.

Note: You must specify the required network-protocol-to-X.121 address mapping with an **x25 map** subcommand before you can set up a PVC.

---

x25 pvc *pvc-number* **interface** *interface-name pvc-number*

Configures a PVC for a given interface. The argument *pvc-number* is the PVC number that will be used on the local interface (as defined by the primary interface command). The argument *interface-name* is the interface type and unit number (serial 0, for example), as specified by the **interface** keywords.

[no] x25 rpoa *name number*

Specifies a list of transit RPOAs to use, referenced by name. The argument *name* must be unique with respect to all other RPOA names. It is used in the **x25 facility** and **x25 map** interface subcommands. The argument *number* is a number that is used to describe an RPOA.

[no] x25 suppress-calling-address

Omits the calling (source) X.121 address in Call Request packets. This option is required for networks that expect only subaddresses in the calling address field. The calling address is sent by default. The **no** form of the command resets the default state.

x25 t10 *seconds*

Sets the limit for the Restart Request retransmission timer (T10) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 60 seconds.

x25 t11 *seconds*

Sets the limit for the Call Request retransmission timer (T11) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 180 seconds.

x25 t12 *seconds*

Sets the limit for the Reset Request retransmission timer (T12) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 60 seconds.

x25 t13 *seconds*

Sets the limit for the Clear Request retransmission timer (T13) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 60 seconds.

x25 t20 *seconds*

Sets the limit for the Restart Request retransmission timer (T20) on DTE devices. The argument *seconds* is a time value in seconds. The default is 180 seconds.

x25 t21 *seconds*

Sets the limit for the Call Request retransmission timer (T21) on DTE devices. The argument *seconds* is a time value in seconds. The default value is 200 seconds.

x25 t22 *seconds*

Sets the limit for the Reset Request retransmission timer (T22) on DTE devices. The argument *seconds* is a time value in seconds. The default value is 180 seconds.

x25 t23 *seconds*

Sets the limit for the Clear Request retransmission timer (T23) on DTE devices. The argument *seconds* is a time value in seconds. The default value is 180 seconds.

x25 th *delay*

Instructs the router to send acknowledgment packets when it is not busy sending other packets, even if the number of input packets has not reached the **win** count, which improves line responsiveness at the expense of bandwidth. The argument *delay* must be between zero and the input window size. A value of one sends one Receiver Ready acknowledgment per packet at all times. The default value is zero, which disables the delayed acknowledgment strategy.

[no] x25 use-source-address

Updates the source address of outgoing calls forwarded over a specific interface use the following command. The **no** variation prevents the update.

x25 win *packets*
x25 wout *packets*

Set the upper limits on the number of outstanding unacknowledged packets. The **win** keyword specifies the upper limits of the number of outstanding unacknowledged packets in the input window., and determines how many packets the router can receive before sending an X.25 acknowledgment.

The **wout** keyword specifies the upper limits of the number of outstanding unacknowledged packets in the output window. The **wout** limit determines how many sent packets can remain unacknowledged before the router uses a hold queue.

To maintain high bandwidth utilization, assign these limits the largest number that the network allows. The argument *packets* is a packets count. The packet count for **win** and **wout** can range from one to the window modulus. The default value is two packets.

Note: Set **win** and **wout** to the same value unless your network supports asymmetry between input and output window sizes.

Configuring Frame Relay

This section describes frame relay configuration, Cisco Systems' implementation of frame relay, the protocols supported, and the hardware needed to operate with a frame relay network.

Frame relay is described as an encapsulation method and is directed mainly to users with large T1 network installations. Cisco's implementation meets the *Frame Relay Interface* specification produced by Northern Telecom, Digital Equipment Corporation, StrataCom, and Cisco Systems. Cisco's implementation also conforms to the Link Access Procedure (LAP-D) defined by the CCITT under its I-series (ISDN) recommendation as I122, "Framework for Additional Packet Model Bearer Services." Cisco's frame relay implementation currently supports routing on IP, DECnet, AppleTalk, Novell IPX, and VINES, and transparent bridging.

The Cisco network server supports the local management interface (LMI), as specified in the joint *Frame Relay Interface* specification. The LMI includes support for a keepalive mechanism, a multicast group, and a status message.

The keepalive mechanism provides an exchange of information between the network server and the switch to verify data is flowing. The multicast mechanism provides the network server with its local data link connection identifier (DLCI) and the multicast DLCI. The status mechanism provides an on-going status report on the DLCIs known by the switch.

Configuring the Hardware

The following hardware configuration is required for frame relay connections:

- Each router connects directly to the frame relay switch.
- Each router connects directly to a CSU/DSU (Channel Service Unit/Digital Service Unit) first and the CSU/DSU is connected to a remote frame relay switch.

The CSU/DSU converts V.35 or RS-449 signals to the properly coded T1 transmission signal for successful reception by the frame relay network. Figure 8-3 illustrates the connections between the different components.

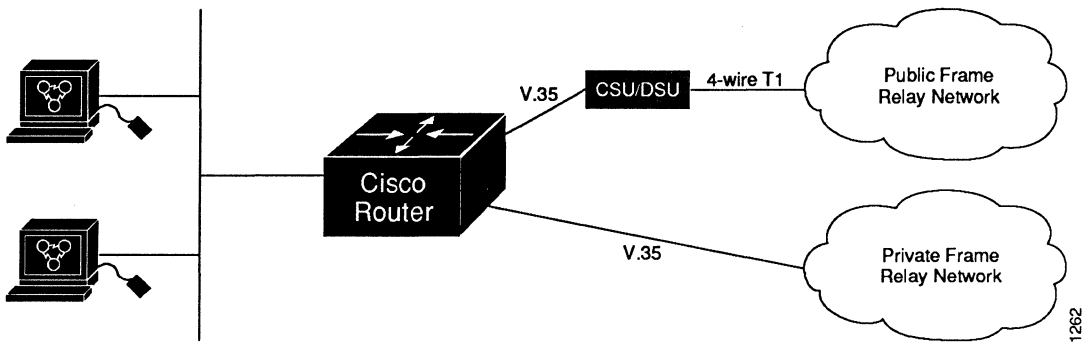


Figure 8-3 Frame Relay Physical Connection

The frame relay interface actually consists of one physical connection between the network server and the switch that provides the service. This single physical connection provides direct connectivity to each device on a network, such as a Stratacom FastPacket wide area network, using only a single connection.

Specifying Frame Relay Encapsulation

Use the **encapsulation frame-relay** interface subcommand to specify frame relay encapsulation on a specific interface.

encapsulation frame-relay

Example:

These commands configure frame relay encapsulation on interface serial 1.

```
interface serial 1
encapsulation frame-relay
```

Setting the Frame Relay Keepalive Timer

The **frame-relay keepalive** interface subcommand enables and disables the LMI mechanism for serial lines using the frame relay encapsulation. The full syntax of this command follows.

frame-relay keepalive *number*

no frame-relay keepalive

The argument *number* defines the keepalive interval. The interval must be set, and must match the interval set on the switch. The default keepalive interval is ten seconds.

Note: The **frame-relay keepalive** and **keepalive** commands perform the same function; both commands enable the keepalive sequence. The keepalive sequence is part ot the LMI protocol, so these commands also control the enabling and disabling of the LMI.

Example:

This command sets the keepalive timer on the Cisco router for a period that is two or three seconds faster (shorter interval) than the interval set on the keepalive timer of the frame relay switch. The difference in keepalive intervals assures proper synchronization between the Cisco router and the frame relay switch.

```
frame-relay keepalive 8
```

Mapping Between an Address and the DLCI

The **frame-relay map** subcommand defines the mapping between an address and the DLCI used to connect to the address. There can be many DLCIs known by a network server that can send data to many different places, but all this data will be multiplexed over the one physical link. The frame relay map tells the network server how to get from a specific protocol and address pair to the correct DLCI. The full syntax of this command follows.

frame-relay map *protocol protocol-address DLCI* **[broadcast]**

no frame-relay map *protocol protocol-address*

The argument *protocol* is one of these keywords: **ip**, **decnet**, **appletalk**, **xns**, **novell**, **vines**, or **clns**. The keyword is followed by the corresponding protocol address and the DLCI number.

This variation of the **frame-relay map** command is used for the ISO CLNS protocol:

frame-relay map clns *DLCI* **broadcast**

This variation of the **frame-relay map** command is used for bridging:

frame-relay map bridge *DLCI* **broadcast**

In both these variations, there is no need to specify a protocol address.

The optional keyword **broadcast** specifies that broadcasts should be forwarded to this address when the multicast is not enabled. The default is not to forward broadcasts. The **broadcast** keyword is required for ISO CLNS and bridging applications.

The **no frame-relay map** subcommand with the appropriate arguments deletes the entry.

Examples of command use follow; see Chapter 20, "Configuring Transparent Bridging" in Part Five for the procedures to configure bridging on frame relay.

Example 1:

This command maps IP address *131.108.123.1* to *DLCI 100*. Broadcasts are not forwarded.

```
frame-relay map IP 131.108.123.1 100
```

Example 2:

This command uses *DLCI 144* for bridging.

```
frame-relay map bridge 144 broadcast
```

Example 3:

This command uses *DLCI 125* for ISO CLNS routing.

```
frame-relay map clns 125 broadcast
```

Requesting Short Status Messages

The **frame-relay short-status** interface subcommand instructs the network server to request the short status message from the switch (see version 2.3 of the joint *Frame Relay Interface* specification). The full syntax of this command follows:

frame-relay short-status

no frame-relay short-status

The default is to request the full status message. Use the **no frame-relay short-status** command to override the default.

Example:

This command returns the interface to the default state of requesting full status messages.

```
no frame-relay short-status
```

Setting a Local DLCI

The **frame-relay local-dlci** interface subcommand sets the source DLCI for use when the LMI is not supported. If LMI is supported and the multicast information element is present, the network server sets its local DLCI based on information provided via the LMI. The full syntax of this command follows:

frame-relay local-dlci *number*

no frame-relay local-dlci

The argument *number* is the local, or source, DLCI number. The **no frame-relay local-dlci** command removes DLCI number.

Note: The **frame-relay local-dlci** command is provided mainly to allow testing of the frame relay encapsulation in a setting where two routers are connected back to back. This command is not required in a live frame relay network.

Example:

This command specifies 100 as the local DLCI.

```
frame-relay local-dlci 100
```

Defining a DLCI for Multicast

The **frame-relay multicast-dlci** interface subcommand define or remove a DLCI to be used for multicasts. The full syntax of this command follows.

frame-relay multicast-dlci *number*

no frame-relay multicast-dlci

This command should only be used when the multicast facility is *not* supported. Network transmissions (packets) sent to a multicast DLCI are delivered to all network servers defined as members of the multicast group. The multicast DLCI is identified by the argument *number.* (Note that this is *not* the multicast group number, which is an entirely different value.) The **no frame-relay multicast-dlci** command removes the multicast group.

Note: The **frame-relay multicast-dlci** command is provided mainly to allow testing of the frame relay encapsulation in a setting where two routers are connected back to back. This command is not required in a live frame relay network.

Example:

This command specifies 1022 as the multicast DLCI.

```
frame-relay multicast-dlci 1022
```

Frame Relay Configuration Examples

The following examples are included to show you how you may configure your router to support frame relay connections.

Two Routers in Static Mode

The following examples illustrate how to configure two routers for static mode.

Example 1—First Router:

```
interface  serial 0
ip address  131.108.64.2 255.255.255.0
encapsulation  frame-relay
frame-relay keepalive 10
frame-relay map ip  131.108.64.1 43
```

Example 2—Second Router:

```
interface  serial 0
ip address  131.108.64.1 255.255.255.0
encapsulation  frame-relay
frame-relay keepalive 10
frame-relay map ip  131.108.64.2 44
```

Routing DECnet Packets

The following example illustrates how to send all DECnet packets destined for address 56.4 out on DLCI 101. In addition, any DECnet broadcasts for interface serial 1 will also be sent on the DLCI.

Example:

```
interface  serial 1
decnet routing 32.6
encapsulation  frame-relay
frame-relay map decnet 56.4 101 broadcast
```

Routing Novell Packets

The following example illustrates how to send packets destined for Novell address *200.0000.0c00.7b21* out on DLCI 102.

Example:

```
interface  ethernet 0
novell network 2abc
!
interface serial 0
novell network 200
encapsulation frame-relay
frame-relay map novell 200.0000.0c00.7b21 102 broadcast
!
```

Monitoring Frame Relay

Use the EXEC commands in this section to monitor frame relay connections.

Monitoring the Frame Relay Interface

When using the frame relay encapsulation, the EXEC command **show interface** includes information on the multicast DLCI, the DLCI of the interface, and the LMI DLCI used for the local management interface.

The multicast DLCI and the local DLCI can be set using the **frame-relay multicast-dlci** and the **frame-relay local-dlci** subcommands, or provided through the local management interface. The status information is taken from the LMI, when active.

Enter this command at the EXEC prompt:

show interfaces [*type unit*]

Following is sample output.

```
Serial 2 is up, line protocol is up
  Hardware type is MCI Serial
  Internet address is 131.108.122.1, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set (10 sec)
  multicast DLCI 1022,  status defined, active
  source DLCI    20, status defined, active
  LMI DLCI 1023, LMI sent 10, LMI stat recvd 10, LMI upd recvd 2
  Last input 7:21:29, output 0:00:37, output hang never
  Output queue 0/100, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 0 bits/sec, 0 packets/sec
     47 packets input, 2656 bytes, 0 no buffer
     Received 5 broadcasts, 0 runts, 0 giants
     5 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 57 abort
     518 packets output, 391205 bytes
     0 output errors, 0 collisions, 0 interface resets, 0 restarts
     1 carrier transitions
```

In this display, the multicast DLCI has been changed to 1022 with the **frame-relay multicast-dlci** interface subcommand.

In this display, the statistics for the LMI are the number of status inquiry messages sent (`LMI sent`), the number of status messages received (`LMI recvd`), and the number of status updates received (`upd recvd`). See the *Frame Relay Interface* specification for additional explanations of this output. Chapter 6 of this publication also provides explanation about the other fields seen in the **show interfaces** command.

Displaying Frame Relay Map Entries

Use this EXEC command to display the current frame relay map entries and information about these connections:

show frame-relay map

Sample output follows:

```
Serial2: IP 131.108.122.2 dlci 10(0XA,0xA0), dynamic
         status defined, active
```

The display lists the interface, the protocol, the protocol address, and the DLCI being used to reach this address. If the optional broadcast keyword was entered for a static map entry, this will also be shown.

The DLCI is displayed in three forms. For example, if the DLCI is 10, the representations would be `10 (0xA, 0xA0)`. The displays show the decimal value, the hexadecimal value, and the value of the DLCI as it would appear on the wire. In addition, the display indicates whether this is a static or dynamic entry. Status information for the DLCI is display if provided by the LMI.

For the CLNS protocol the map has the following form:

```
Serial0: CLNS  dlci 100(0X64,1840), static, broadcast, BW = 64000
         status defined, active
Serial0: CLNS  dlci 102(0X66,1860), static, broadcast, BW = 64000
         status defined, active
```

Displaying Global Frame Relay Statistics

Use the **show frame-relay traffic** command to display global frame relay statistics. Enter this command at the EXEC prompt:

show frame-relay traffic

Sample output follows:

```
Frame Relay statistics:
ARP requests sent 14, ARP replies sent 0
ARP request recvd 0, ARP replies recvd 10
LMI sent 10, LMI stat recvd 10, LMI upd recvd 2, Multicast sent 48
```

Statistics for all frame relay interfaces in the router are also included in this display.

Debugging Frame Relay

Use the EXEC commands described in this section to troubleshoot and monitor activity on the interface configured for frame relay. For each **debug** command, there is a corresponding **undebug** command that turns messaging off.

debug frame-relay-events

This command enables logging of key events in the transmission or receipt of packets encapsulated using frame relay.

debug frame-relay-lmi

This command enables logging of information on the local management interface packets exchanged between the router and the frame relay service provider.

debug frame-relay-packets

This command displays all packets being sent out on the frame relay network. The display identifies the output interface, the protocol identifier, and the size of the packet being sent.

Frame Relay Interface Subcommand Summary

Following is an alphabetically arranged summary of the frame relay interface subcommands.

encapsulation frame-relay

Specifies frame relay encapsulation on a specific interface.

[no] frame-relay keepalive *number*

Enables and disables the LMI mechanism for serial lines using the frame relay encapsulation.

frame-relay local-dlci *number*

Sets the source DLCI for use when the LMI is not supported. If LMI is supported and the multicast information element is present, the network server sets its local DLCI based on information provided via the LMI. The argument *number* is the local, or source, DLCI number.

frame-relay map *protocol protocol-address DLCI* **[broadcast]**

Defines the mapping between an address and the DLCI used to connect to the address. The frame relay map tells the network server how to get from a specific protocol and address pair to the correct DLCI. The argument *protocol* can be one of these keywords: **ip, decnet, appletalk, xns, novell, vines, clns**. The keyword is followed by the corresponding protocol address and the DLCI number. The optional **broadcast** flag specifies that broadcasts should be forwarded to this address when the multicast is not enabled. The default is not to forward broadcasts.

frame-relay map clns *DLCI* **broadcast**

This variation of the **frame-relay map** command is used for the ISO CLNS protocol.

frame-relay map bridge *DLCI* **broadcast**

This variation of the **frame-relay map** command is used for bridging.

no frame-relay map

Deletes the frame relay map entry.

frame-relay multicast-dlci *number*

Defines a DLCI to be used for multicasts and should only be used when the multicast facility is *not* supported. Network transmissions (packets) sent to a multicast DLCI are delivered to all network servers defined as members of the multicast group. The argument *number* identifies the multicast group.

[no] frame-relay short-status

Instructs the network server to request the short status message from the switch (see version 2.3 of the joint *Frame Relay Interface* specification). The default is to request the full status message.

Configuring Switched Multi-Megabit Data Services (SMDS)

The Switched Multi-Megabit Data Service (SMDS) is a wide area networking service offered by Regional Bell Operating Companies (RBOCs) and other telephone service carriers such as AT&T and MCI/Sprint. The SMDS protocol is based on cell relay technology as defined in the Bellcore Technical advisories, which is in turn based on the IEEE 802.6 Standard (also called the Distributed Queue Dual Bus (DQDB) Metropolitan Area Network Media Access Control protocol). For technical references please see the bibliography in the "References and Recommended Readings" list at the end of this publication.

Cisco provides an interface to an SMDS network using DS1 transmission facilities at the rate of 1.544 Mbps. Connection to the network is made through a device called an SDSU—an SMDS CSU/DSU (Channel Service Unit/Digital Service Unit) developed jointly by Cisco Systems and Kentrox. The SDSU attaches to a Cisco router through an RS-449 connection. On the other side, the SDSU terminates a DS1 line.

Cisco's implementation of SMDS supports the IP, DECnet, AppleTalk, XNS, Novell IPX, Ungermann-Bass Net/One, and OSI internetworking protocols. Routing of IP is fully dynamic; that is, the routing tables are determined and updated dynamically. Routing of the other supported protocols requires that you establish a static routing table of SMDS neighbors in a user group. Once this is set up, all interconnected routers provide dynamic routing.

This section describes Cisco's implementation of SMDS, and how to configure, maintain, and debug SMDS.

Hardware Requirements

You need the following hardware and equipment to configure the Cisco Systems' SMDS implementation:

- An MCI or SCI serial interface controller card, or HSSI interface (chassis-based systems) or the serial port on an IGS router
- An RS-449 applique (chassis-based systems) or RS-449 transition cable (IGS)
- The SDSU device
- The packet-switched software option with the system software

Configuring SMDS

Follow these steps to configure SMDS service on the Cisco router:

Step 1: Determine the protocols you will be running over SMDS. Obtain from the service provider the group addresses that you will need to support those protocols.

Step 2: Obtain the SMDS hardware (individual) address from the service provider for each router that will interface directly into the SMDS network (that is, customer premises equipment).

Step 3: You will also need to know the addresses of the other routers with whom you'll be communicating, to set up the static routing tables. Please note that static mapping is needed only for protocols other than IP or CLNS. For IP and CLNS the routing is fully dynamic. (For more details please see these routing chapters in Part Four of this manual.)

Step 4: Configure the desired serial interface with SMDS encapsulation.

Step 5: Set up the static map for the desired protocols using the **smds static-map** command.

Step 6: Set up the multicast map for the desired protocols using the **smds multicast** command.

Using SMDS Addresses

All addresses for SMDS service are assigned by the service provider, and may be assigned to individuals and groups.

A group address (also defined as a multicast address) is entered in the Cisco SMDS configuration software using an E1 prefix; a C1 prefix is used to specify individual addresses. The Cisco software expects the addresses to be entered in a slightly modified E.164 format. E.164 format is 64 bits. The first four bits are type code followed by 10 BCD digits padded to the full 60 bits with ones.

An example of an E.164 address follows:

```
C14155561313FFFF
```

Note: To simplify the addresses, Cisco does not require the full E.164 address. The trailing FFFF's are not needed. They are not displayed and it is not necessary to type them when entering an address.

The addresses may be entered with periods in a manner similar to Ethernet-style notation, or simply as a string of digits.

An example of an individual address entered in Ethernet-style notation would look like this:

```
C141.5555.1212
```

An example of a group address would look like this:

```
E18009999999
```

Enabling SMDS

Enter the **encapsulation smds** interface subcommand to enable SMDS service on the desired interface:

encapsulation smds

The interface to which this command applies must be a serial interface. All subsequent SMDS configuration commands only apply to an interface with encapsulation SMDS.

Example:

Following is an example of the commands you use to configure the SMDS service on interface serial 0:

```
!
interface serial 0
encapsulation smds
```

Note: The maximum packet size allowed in the SMDS specifications (TA-772) is 9188. This is larger than what can be used by routers with most media. We therefore default the MTU to 1500 bytes, to be consistent with Ethernet. If a larger MTU is used, the **mtu** command must be used before the **encapsulation smds** command is used.

Specifying the SMDS Address

Enter the **smds address** interface subcommand to specify the SMDS individual address for a particular interface. The format of the command follows:

smds address *smds-address*

no smds address *smds-address*

Enter an individual address provided by the SMDS service provider for the argument *smds-address*. Enter the address, as described in the section "Using SMDS Addresses." This address is protocol independent.

Enter the **no smds address** command to remove the address from the configuration file.

There is no default for this command.

Example:

Following is an example which shows how a command specifies an individual address.

```
!
interface serial 0
smds address C141.5797.1313
!
```

Note: If bridging is enabled on any interface, the SMDS address is erased, and must be re-entered.

Enabling the Address Resolution Protocol

Enter the **smds enable-arp** interface subcommand to enable the Address Resolution Protocol (ARP). The full syntax of this command follows.

smds enable-arp

no smds enable-arp

The multicast address for ARP must be set before this command is issued.

By default, ARP is not enabled. Once ARP has been enabled, use the **no smds enable-arp** command to return the line to the default state.

Defining a Static Map for an Individual Address

Enter the **smds static-map** interface subcommand to configure a static mapping between an individual SMDS address and a higher level protocol address. The full syntax of the command follows:

smds static-map *protocol-type protocol-address smds-address*

no smds static-map *protocol-type protocol-address smds-address*

Do not enter this command for broadcast or multicast addresses. For those addresses, use the **smds multicast** interface subcommand described in the section "Mapping to a Multicast Address."

Enter the name of the protocol for the *protocol-type* argument, and follow with the address of the protocol to answer the *protocol-address* argument. Provide the SMDS address for the *smds-address* argument to complete the mapping. You must use these keywords to define the protocol type: **ip**, **decnet**, **appletalk**, **xns**, **novell**, or **clns**.

Use the **no smds static-map** command with the appropriate arguments to remove the map.

Example:

Following is an example of the command.

```
!
smds static-map XNS 111.00C0.2711.0123  C141.5688.1212
!
```

The command will map XNS address *111.00C0.2711.0123* to the individual SMDS address C141.5688.1212.

Mapping to a Multicast Address

Enter the **smds multicast** interface subcommand to map an SMDS group address to a broadcast or multicast address used by higher level protocols. The full syntax of the command follows:

smds multicast *protocol-type smds-address*

no smds multicast *protocol-type smds-address*

Enter the name of the protocol for the *protocol-type* argument, and follow with the SMDS address to answer the *smds-address* argument to complete the mapping. You may use these keywords to define the protocol type:

- **ip**—IP
- **arp**—ARP
- **decnet**—DECnet
- **decnet_router**—DECnet multicast address for all routers
- **decnet_node**—DECnet multicast address for all end systems
- **appletalk**—AppleTalk
- **aarp**—AppleTalk ARP address
- **xns**—XNS
- **novell**—Novell IPX
- **clns**—ISO CLNS
- **clns_is**— Multicast address for all CLNS Intermediate Systems
- **clns_es**— Multicast address for all CLNS End Systems

Since SMDS does not incorporate broadcast addressing, a group address for a particular protocol must be defined to serve the broadcast function. There is no default for this command.

Use the **no smds multicast** command with the appropriate arguments to remove a multicast address.

Example:

Following is an example of the command:

```
!
smds multicast IP  E180.0999.9999
!
```

The command maps the IP broadcast address to the SMDS group address *E180.0999.9999.*

Enabling the AT&T SMDS Service

Cisco's implementation of SMDS includes a configuration option that enables the router to interface to an AT&T SMDS router that implements an early, prestandard version of the IP encapsulation on SMDS standard protocol. This version has been superceded by the SMDS standards. However, the pre standard implementation represents an embedded base of available service. Please consult your service provider to find out if this command will be needed.

Use the **smds att-mode** interface subcommand in order to operate with the AT&T SMDS router. The full command syntax follows:

smds att-mode

no smds att-mode

AT&T's implementation does not match Bellcore's SMDS specification. When this switch is enabled, the router will use AT&T's implementation. It should be off if protocols others than IP are to be routed. (In the early AT&T implementation, only IP was supported.)

Use the **no smds att-mode** command to turn this function off. By default, the function is on.

Protocol-Specific Configuration

An SMDS network can be thought of in much in the same way as an X25 cloud. The premises equipment, in this case a Cisco router, represents the edge of the cloud. The service provider enables communication across the cloud. However, proper configuration is needed for communication to occur. This configuration will differ between protocol families.

One major difference between protocol families is dynamic versus static routing among the routers (called remote peers) on the periphery of the cloud. For IP and CLNS, routing across the SMDS cloud is fully dynamic. No action on the user's part is needed for the mapping of higher level protocol addresses to SMDS addresses to occur. For the other supported protocols, a static entry must be made for each of the other neighbors. This entry provides a router with the information that it needs to communicate with all other neighbor routers.

Up until now this discussion has centered on the peer routers. What about all of the end nodes and routers behind the SMDS router? The static entries only need to be made for those routers that are SMDS remote peers. Nothing additional needs to be done in order to communicate with other nodes behind the peer routers.

Some protocol families need separate definitions for associated subprotocols. The next sections illustrates how to implement these kind of configurations. See Table 8-7 for a list of protocol families and what multicast is needed.

Table 8-7 Protocol Families and the Type of Multicasts Needed

Protocol Family	Multicasts needed
IP	IP, ARP
DECNet	DECNET, DECNET_NODE, DECNET_ROUTER
CLNS	CLNS, CLNS_ES, CLNS_IS
Novell	NOVELL
XNS	XNS
Appletalk	APPLETALK, AARP

1268

Configuring IP

Both IP and ARP should be configured with the **smds multicast** command. ARP should be enabled. The results of the ARP activity can be shown with the **show arp** command. If desired, static ARP entries may be made by using this command:

arp *ip-address address*

The argument *ip-address* is the IP address. The argument *address* is the SMDS address.

Configuring AppleTalk

Currently, dynamic address assignment does not work, and therefore an AppleTalk address must be assigned to the interface, and each remote router peer must be listed with an **smds static-map** command. The AppleTalk ARP (AARP) multicast address must still be configured with the **smds multicast** command. ARP should be enabled.

Configuring XNS and Novell

The XNS and/or NOVELL the multicast address must be configured.

For Novell, RIP Routing packets, SAP packets, NetBios Name Lookups, directed broadcasts, and traffic to the helper addresses (if that helper address is a broadcast address) will be sent to the SMDS novell multicast address.

For XNS, only RIP, directed broadcasts, and helper traffic will be sent to the XNS multicast address.

For XNS and/or Novell configurations, a static map entry must be made for each remote peer.

Configuring CLNS

Multicasts must be configured for CLNS_ES, and CLNS_IS. No static maps are necessary. ESH's, ISH's, and Router Hellos are sent to the multicast address, and neighbor entries are created automatically.

SMDS Configuration Examples

This section provides some examples of configurations to use as models for your configuration files.

Typical Multiprotocol Configuration

Following is a typical interface configured for IP, DECNET, CLNS, Novell, XNS, and AppleTalk.

Example:

```
interface Serial 4
ip address 1.1.1.2 255.0.0.0
decnet cost 4
appletalk address 92.1
appletalk zone smds
clns router igrp FOO
novell net 1a
xns net 17
encapsulation SMDS
! smds configuration follows
smds address c120.1580.4721
no smds att-mode
smds static-map APPLETALK 92.2 c120.1580.4592
smds static-map APPLETALK 92.3 c120.1580.4593
smds static-map APPLETALK 92.4 c120.1580.4594
smds static-map NOVELL 1a.0c00.0102.23ca c120.1580.4792
smds static-map XNS  17.0c00.0102.23ca c120.1580.4792
smds static-map NOVELL 1a.0c00.0102.23dd c120.1580.4728
smds static-map XNS 17.0c00.0102.23aa c120.1580.4727
smds multicast NOVELL e180.0999.9999
smds multicast XNS e180.0999.9999
smds multicast ARP e180.0999.9999
smds multicast IP e180.0999.9999
smds multicast APPLETALK e180.0999.9999
smds multicast AARP e180.0999.9999
smds multicast CLNS_IS e180.0999.9990
smds multicast CLNS_ES e180.0999.9990
smds multicast DECNET_ROUTER e180.0999.9992
smds multicast DECNET_NODE e180.0999.9992
smds enable-arp
```

Configuration with a Remote Peer on the Same Network

An example of a remote peer on the *same* SMDS network follows. Note that this router does not have DECnet routing enabled.

Example:

```
interface Serial 0
ip address 1.1.1.1 255.0.0.0
appletalk address 92.2
appletalk zone smds
clns router igrp FOO
novell net 1a
xns net 17
encapsulation SMDS
! smds configuration follows
smds address c120.1580.4792
no smds att-mode
smds static-map APPLETALK 92.1 c120.1580.4721
smds static-map APPLETALK 92.3 c120.1580.4593
smds static-map APPLETALK 92.4 c120.1580.4594
smds static-map NOVELL 1a.0c00.0102.23cb c120.1580.4721
smds static-map XNS 17.0c00.0102.23cb c120.1580.4721
smds static-map NOVELL 1a.0c00.0102.23dd c120.1580.4728
smds static-map XNS 17.0c00.0102.23aa c120.1580.4727
smds multicast NOVELL e180.0999.9999
smds multicast XNS e180.0999.9999
smds multicast ARP e180.0999.9999
smds multicast IP e180.0999.9999
smds multicast APPLETALK e180.0999.9999
smds multicast AARP e180.0999.9999
smds multicast CLNS_IS e180.0999.9990
smds multicast CLNS_ES e180.0999.9990
smds enable-arp
```

Monitoring SMDS Service

Use the following EXEC commands to monitor the SMDS service.

Displaying SMDS Individual Addresses

Use the **show smds addresses** command to display the individual addresses and the interface that they are associated with. Enter this command at the EXEC prompt:

show smds addresses

A sample display of the command output follows:

```
Artemis#show smds addresses
SMDS address - Serial0   c141.5555.1212
```

Displaying Mapped SMDS Addresses

Use the **show smds map** command to display all SMDS addresses that are mapped to higher level protocol addresses. Enter this command at the EXEC prompt:

show smds map

The display for this command includes all addresses entered with both the **smds static-map** and **smds multicast** command.

A sample display of the command output follows:

```
Artemis#show smds map
Serial0:  ARP maps to e180.0999.9999 multicast
Serial0:  IP maps to e180.0999.9999 multicast
Serial0:  XNS 1006.AA00.0400.0C55 maps to c141.5688.1212 static
```

Note: Trailing Fs are implied in displays showing the SMDS addresses.

Displaying SMDS Counters

Use the **show smds traffic** command to display all the SMDS counters. Enter this command at the EXEC prompt:

show smds traffic

A sample display of the command output follows:

```
Artemis#show smds traffic
0 Bad BA size errors
0 Bad Header extension errors
0 Invalid address errors
```

In the display:

- The `Bad BA size errors` field lists the number of corrupted packets received based on the expected Level 3 PDU buffer allocation size.
- The `Bad Header extension errors` field lists the number of invalid packets received in which the header extension length did not match what was expected in the Level 3 PDU.
- The `Invalid address errors` field lists the number of packets passed that were incorrectly sent to router. Both individual and multicast address errors are included in this count.

Debugging SMDS

Use the following EXEC command to debug the SMDS service. For each **debug** command, there is a corresponding **undebug** command that turns the messages off.

debug arp

Use this command to see if ARPs are being sent or received. This command prints one line for each ARP sent or received.

debug serial-interface

Use this EXEC command to enable logging of SMDS events. This command prints a one-line message for each packet that is sent. The packet size, packet type, and SMDS source and destination addresses is printed. If packets are received with an incorrect destination address, it will be noted and counted.

SMDS Interface Subcommand Summary

This section provides an alphabetically arranged summary of the SMDS interface subcommands. These commands must be preceded by an **interface** command.

encapsulation smds

Enables or disables SMDS on a particular interface. It should precede any SMDS command. This command has no default.

[no] smds address *smds-address*

Sets or removes the SMDS individual address for a particular interface. The argument *smds-address* is the individual address provided by the SMDS service provider, and is protocol independent. This command has no default.

[no] smds att-mode

Enables or disables AT&T's early prestandard SMDS implementation. Default is **smds att-mode**. This command is on by default.

[no] smds enable-arp

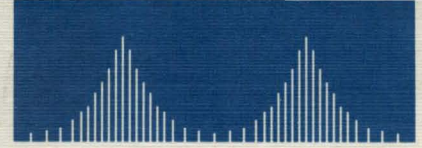
Enables or disables ARP processing on a particular interface. The multicast address for ARP must be set before this command is issued. Default is **no smds enable-arp**.

[no] smds multicast protocol-type smds-group-address

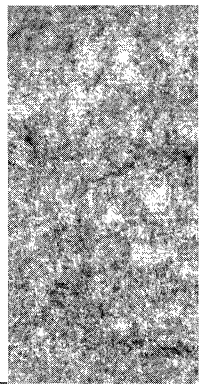
Maps an SMDS group address to a broadcast or multicast address used by higher-level protocols. This command has no default.

[no] smds static-map protocol-type protocol-address smds-address

Sets up a static mapping between an SMDS address and a higher-level protocol address. This should not be used for broadcast addresses, for broadcast address, use the **smds multicast** command. This command has no default.



Index



Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Corporate Headquarters:
Cisco Systems, Inc.

1525 O'Brien Drive
Menlo Park, CA 94025
Tel: 1-415-326-1941
Fax: 1-415-326-1989
1-800-553-NETS