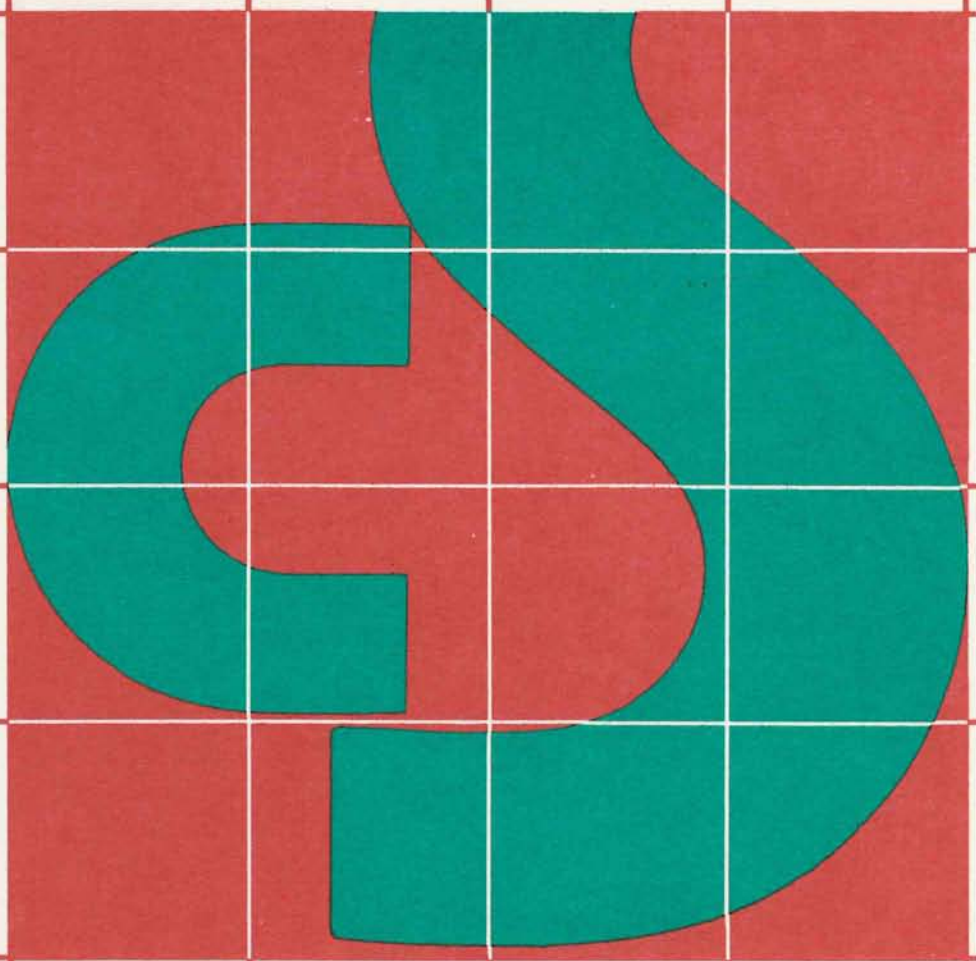
cisco Systems

Gateway System Manual



Gateway System Manual

cisco Systems, Inc.

1350 Willow Road
Menlo Park, California 94025 USA
800-553-NETS 1-415-326-1941

July 1988

The products and the specifications, configurations and other technical information regarding the products contained in this manual are subject to change without notice. All statements, technical information and recommendations contained in this manual are believed to be accurate and reliable but are presented without warranty of any kind, express or implied, and users must take full responsibility for their application of any products specified in this manual. THERE ARE NO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE AS A RESULT OF THIS MANUAL OR THE INFORMATION CONTAINED HEREIN AND ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, ARE EXCLUDED. The exclusive warranty is as stated in the written terms and conditions of **cisco Systems, Inc.** delivered in connection with the sale of products. All other liabilities of **cisco Systems, Inc.** for cost, losses, and damages of any kind, including consequential damages, are hereby expressly excluded.

Notice of Restricted Rights

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the Rights in Technical Data and Software clause in DAR 7-104.9(a).

The *Gateway System Manual* describes the hardware and software of the gateway server. The information in this manual is subject to change without notice.

Gateway System Manual

A Tec-Ed manual.

Preface

This Preface describes the audience, documentation conventions, and ordering information for the *Gateway System Manual*. It also acknowledges contributions to the development of the gateway server.

Audience

This manual addresses three audiences:

- Part-time system administrators with little or no network experience.
- Small system managers who want to understand how to connect the **cisco Systems** gateway server before they actually connect it.
- Experienced network personnel responsible for designing and running their network who want to understand how to use fully the capabilities of the **cisco Systems** gateway server.

The *Gateway System Manual* is organized as follows:

- General information in Chapters 1 through 3
- Installation instructions in Chapters 4 and 5
- Advanced gateway server operation in Chapters 6 through 10
- Detailed protocol and network information in the rest of the manual
  - TCP/IP Internet protocol information in Chapters 11 through 13
  - DDN protocol information in Chapter 14
  - X.25 protocol information in Chapter 15
  - DECnet protocol information in Chapter 16
  - Other non-Internet protocol information in Chapter 17
  - Network monitoring and troubleshooting in Chapter 18
  - Command and other reference information in the appendices

Part-time system administrators may want to read Chapters 1 and 2 before installing their gateway server; small system managers and experienced network personnel can begin with Chapter 3. All audiences need the fundamental information about gateway server operation that Chapters 4 and 5 contain.

Small system managers and experienced network personnel can turn to Chapters 6 through 10 for background as well as more information about gateway server operation. Experienced network personnel will find detailed supporting information about protocols and network management in Chapters 11 through 18 and the appendices.

Documentation Conventions

The command descriptions use these conventions:

- Keywords are in **boldface.**
- Variables for which you supply values are in *italic*.
- Elements in square brackets ([ ]) are optional.
- Alternative keywords are grouped in braces ({ }) and separated by a vertical bar (|).

The sample terminal sessions use these conventions:

- Information the user enters is in **boldface.**
- Non-printing characters are shown in angle brackets (< >).
- Information the system displays appears in ordinary type.
- The system displays default responses in square brackets ([ ]).

Ordering Information

Additional copies of the *Gateway System Manual* are available for $38.00 each from:

cisco Systems, Inc.
1360 Willow Road, Menlo Park, California 94025 USA
800-553-NETS 415-326-1941
customer-service@cisco.com

Acknowledgments

Charles Hedrick at Rutgers University produced the original implementation and documentation for the DECnet implementation on the gateway server.

Table of Contents

Chapter 4: Setting Up the Gateway Server

Chapter 5: Starting Gateway Server Operations

Chapter 11: Understanding Internet Addresses, Broadcasts, and Address Resolution

Chapter 12: Routing With the Gateway Server

Chapter 17: Using Other Protocols

Chapter 18: Network Monitoring and Troubleshooting

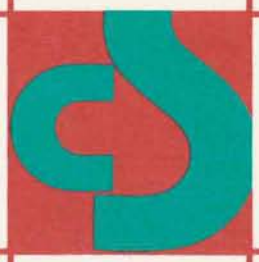
Appendix A: EXEC Command Reference A-1

Appendix B: Configuration Command Reference B-1

List of Figures

List of Tables

cisco Systems



Chapter 1
Why cisco Systems Gateway Servers

cisco Systems designs and produces communications tools to build multi-protocol, multi-media, multi-vendor networks. With **cisco Systems** products, network managers can interconnect terminals, workstations, computer systems, and different networks to create networks of networks.

Internetworking LANs Today

Most organizations today use local area networks (LANs) to connect their computers, workstations, and microcomputers in a single building or over short geographic distances. Users understand how electronic mail, efficient file transfers, shared applications and databases, and shared devices such as printers can help people communicate and reduce operating costs.

Then such organizations try to interconnect the LANs in different divisions or campuses, because they want to achieve the same communication benefits between MIS and manufacturing, or between the San Jose and Boston plants. Unfortunately, the process is not as easy as most advertisements claim.

The San Jose plant uses computers from Company A, and has bought that company's LAN hardware and software. The Boston plant bought its computers and LAN equipment from Company B. A telephone company supplied the data communications link. All the vendors claim their LANs and circuits meet the standard, but when the network managers at both plants try to interconnect the LANs, nothing happens.

Each plant calls its LAN and communications vendors; the service people run diagnostics and assure their customers that nothing is broken. Each vendor says that the network is working exactly as specified. But San Jose computers still cannot talk to those in Boston, and no vendor can explain why not.

Gateway Servers Provide LAN Interconnection

In contrast, **cisco Systems** gateway servers are designed to solve the problem of interconnecting LANs. A single gateway server can interconnect LANs from any vendor that supports one of the many **cisco Systems**-supported protocols.

Part of the frustration in trying to interconnect LANs occurs because many vendors support the "same" networking protocol, such as TCP/IP; but the TCP/IP implementations usually are not completely compatible. **cisco Systems** gateway servers

solve this problem by providing support for all available vendor implementations of TCP/IP. In addition, the gateway servers interconnect LANs using different protocols: X.25, DDN X.25, DECnet, XNS, Chaosnet, and more.

With **cisco Systems** gateway servers, organizations can interconnect their LANs from different vendors using any media they choose: Ethernets, synchronous serial lines, token rings, and media used by public and private data networks. They need not abandon their investment in computers or LANs to achieve internetworking; **cisco Systems** gateway servers interconnect the LANs an organization already uses.

WANs and Complex Internetworks

A wide area network (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 equipment 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. Bridges cannot differentiate the network segments they connect and so are unaware of any network structure.

- Routers can interconnect networks over longer distances and perhaps over different media. Because routers support hierarchical network structures, they offer protection from network errors as well as electrical and data-related problems.

- Gateways are highly capable routers. Gateways perform routing over networks that use different media and, sometimes, different protocols. Often people use the terms "router" and "gateway" interchangeably.

A WAN multiplies both the benefits and the potential problems of LANs. Virtually all organizations large enough to consider a WAN have already made major financial commitments to computing and networking equipment, usually from several vendors.

Unfortunately, most current networking technologies do not permit an organization to leverage its investment in existing equipment while quickly acquiring additional capabilities. These technologies interoperate with only one manufacturer's equipment, using limited media, and support only one network protocol.

cisco Systems Philosophy of Networking

Products from **cisco Systems** solve the problems of complex internetworks by supporting multi-protocol, multi-media, multi-vendor networks. Even organizations that do not yet require a complex 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** gateway server. A **cisco Systems** gateway server is a dynamic gateway that provides a wide variety of network management capabilities in addition to the multi-protocol, multi-media, multi-vendor internetworking already described.

cisco Systems gateway servers automatically perform the network management activities that most vendors require special equipment to achieve. For example, most other network equipment uses static routing tables that cannot handle dynamic network changes. When someone adds a new network host or fixes a broken segment, the network cannot use the changes until the tables in all the network routers are updated.

Instead of static routing tables, **cisco Systems** gateway servers use dynamic internetwork routing, which automatically adjusts to changing topologies. **cisco Systems** gateway servers continually monitor the network to find more reliable links or routes with faster transmission.

Just as in interconnecting LANs, **cisco Systems** gateway servers help build WANs that achieve the interoperability and connectivity large organizations need. Gateway 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** gateway servers is both more cost-effective and more efficient. **cisco Systems** gateway servers help organizations take advantage of all their existing computers and networking equipment. The dynamic routing of the gateway servers ensures that network traffic reaches the intended addresses promptly, regardless of different protocols, equipment from different vendors, and ongoing network configuration changes.

Examples of Networks Using cisco Systems Products

With **cisco Systems** gateway servers, an organization can create networks that combine Ethernets, token ring LANs, and connections to public and private WANs, as shown in Figure 1-1. Equipment on connected LANs can communicate at up to 10,000 kilobits/second, regardless of vendor. One gateway server can connect dozens of computers in an IBM token ring or Ethernet LAN into a larger internetwork.

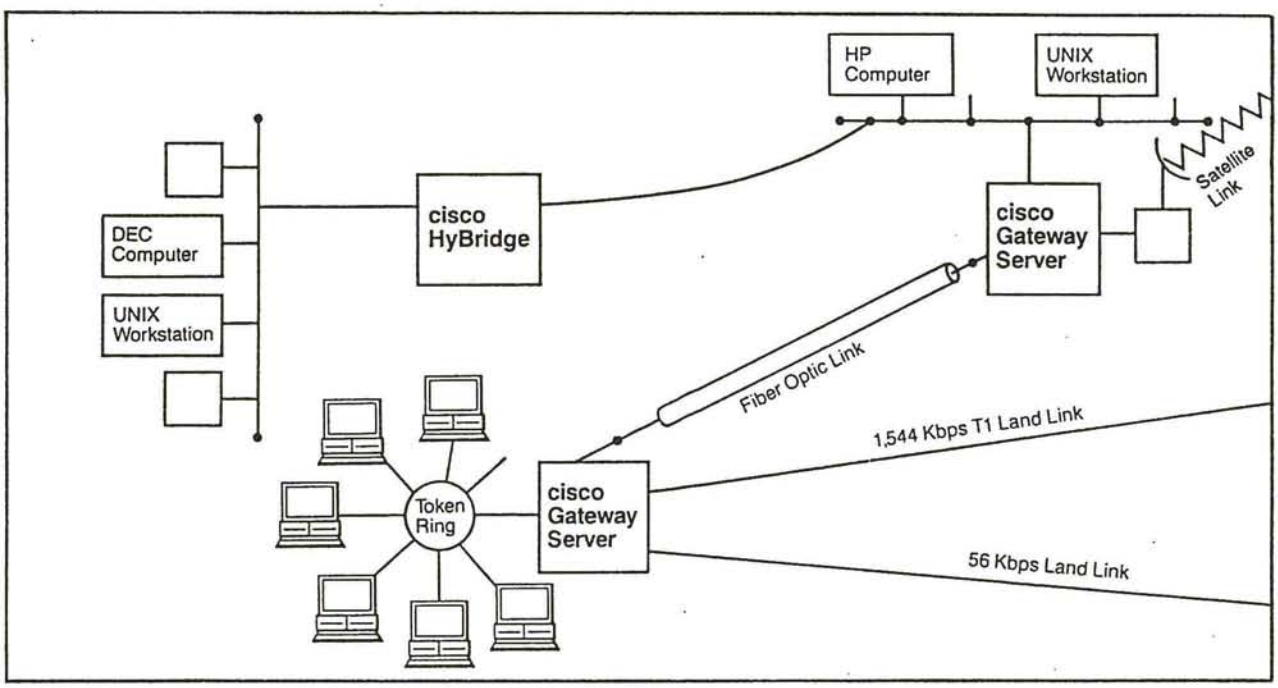


Figure 1-1. A Wide Area Network Built Over Diverse Media

The gateway server can connect LANs into WANs using cost-effective 56 kilobits/second digital service, T1 serial circuits at 1,544 kilobits/second, high-speed T1C serial circuits at 3,100 kilobits/second, and British Telecom Megastream and CEPT DS1 circuits at 2,048 to 4,096 kilobits/second. For future high-performance networking, **cisco Systems** is engineering FDDI, T3, and ISDN support.

Figure 1-2 shows how organizations can create networks of networks with **cisco Systems** gateway servers. A **cisco Systems** gateway server can handle communication among all these public and private networks, regardless of the protocols they use.

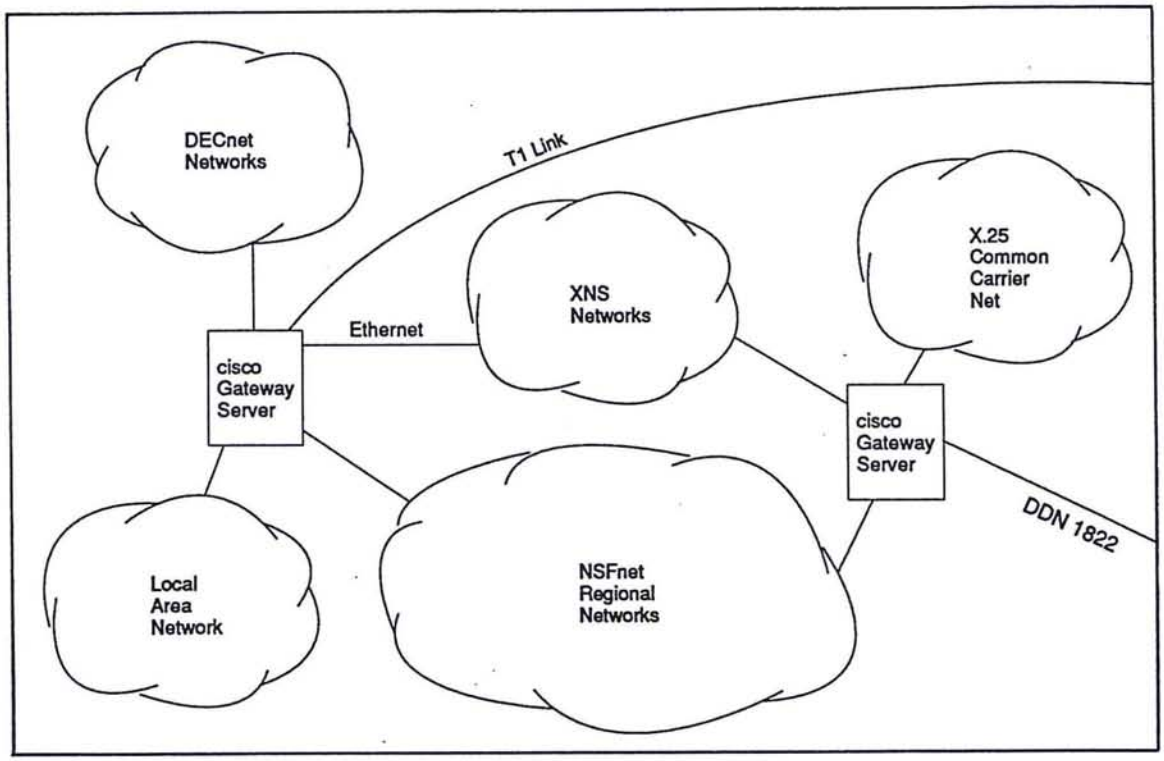


Figure 1-2. A Network of Networks Using Different Protocols

The gateway server can also serve as a protocol translator. For example, a gateway server connected to one network that uses X.25 and to another network that uses TCP/IP automatically performs the protocol conversions needed to transfer data between the networks.

Refer to Chapter 3, "Capabilities of the Gateway Server", for descriptions of the protocols and media **cisco Systems** supports, as well as for more information on **cisco Systems** dynamic routing process.

For Further Reference

The *Gateway System Manual* assumes that readers are familiar with data communication networks. However, many network users and managers may not have experience with Ethernet, TCP/IP, or X.25 concepts or terminology.

Readings About Networks in General

A good general reference on networking technology is *Local Area Networks*, by John E. McNamara. This book is available from Digital Press, Educational Services, Digital Equipment Corporation, 12 Crosby Drive, Bedford, Massachusetts 01730, USA.

Another general reference on networking technology is the *Handbook of Computer Communications Standards*, by William Stallings. This three-volume set discusses many important aspects of modern data communications. Volume 1 describes the ISO (International Standards Organization) Reference Model and the X.25 recommendation. Volume 2 describes physical network standards such as IEEE 802.3. Volume 3 describes the U.S. Department of Defense TCP/IP protocol suite used by **cisco Systems** products. The set was published in 1987 by Macmillan Publishing Company, New York.

For general information on the role of telephone transmission in networking, refer to *Data Communication: A User's Guide*, 2nd edition, by Ken Shernn. This book was published in 1985 by Reston Publishing Company, Inc., in Reston, Virginia.

Information About TCP/IP

The *Gateway System Manual* makes numerous references to the Requests for Comments, or RFCs, that define the Internet Protocols. Complete specifications of the Internet Protocols employed in the **cisco Systems** software may be found in the *DDN Protocol Handbook*, available from the Defense Data Network Information Center, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025, USA.

An excellent introduction to the "nuts and bolts" of Internet Protocols may be found in Douglas Comer's *Operating System Design*, Volume II, "Internetworking with Xinu", published in 1987 by Prentice-Hall. The same author has also written *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, which Prentice-Hall published in 1988.

Another basic overview of Internet Protocols is in *An Introduction to TCP/IP*, written by John Davidson and published in 1988 by Springer-Verlag.

Information About the X.25 Recommendation

The formal definition of X.25 is found in *CCITT Data Communication Networks--Interfaces, Recommendations X.20 - X.32*, Volume VIII, published by CCITT, 1984. The letters CCITT identify the International Telegraph and Telephone Consultative Committee.

An easy-to-understand explanation of X.25 is found in *X.25: The PSN Connection; An Explanation of Recommendation X.25*, published by Hewlett-Packard, October 1985. The Hewlett-Packard part number is 5958-3402.

cisco Systems

Chapter 2
cisco Systems and Its Products

cisco Systems delivers network solutions developed by people who understand network problems. Ongoing hardware and software development ensure support for new protocols and transmission media as they emerge. **cisco Systems** engineers continually apply the latest developments in networking research to create the next generation of internetworking products.

cisco Systems: Pioneers in Network Science

Founded in 1984, **cisco Systems** designs, develops, manufactures, and markets state-of-the-art communications technology. The company specializes in low-cost, high-performance terminal servers and gateway servers, as described in the rest of this chapter.

The founders of **cisco Systems** have been networking pioneers since their work at Stanford University. In 1980, cofounder Len Bosack began to implement the TCP/IP network standard on the Stanford University Network (SUN) under contract to DARPA. This network, one of the earliest to use TCP/IP, was also for many years the largest Ethernet-based LAN, with 70 segments.

Today **cisco Systems** products are in use on four continents. Customers include Northern Telecom, SRI International, Ford Aerospace, Stanford University, Nippon Telephone and Telegraph, Hewlett-Packard Company, the U.S. Department of Defense, BITNET, and many others.

The Product Lines: Network Building Blocks

cisco Systems produces several models of terminal servers and gateway servers along with the HyBridge, a product that combines bridging and routing functions in one device. **cisco Systems** products help achieve interoperability by enabling organizations to build multi-protocol, multi-media, multi-vendor networks.

All **cisco Systems** products also give network managers several options for software configuration. In addition to configuration from the console terminal or a configuration file, the products can configure themselves automatically from non-volatile memory or from a server host on the network.

Terminal Servers for Network Benefits--Now

cisco Systems terminal servers are communication processors that interconnect multi-vendor terminals, modems, and microcomputers with virtually any network, whether a LAN or a WAN. One terminal server provides connections for up to 96 asynchronous devices. Terminal servers from **cisco Systems** also support efficient printing by enabling network users to share high-quality printers and plotters.

For LAN connections, the terminal servers can multiplex data from both RS-232 serial lines and parallel I/O ports onto their high-speed network interfaces. Each terminal line operates at commonly used data rates up to 38.4 kilobits/second, while supporting rotary and modem functions. **cisco Systems** terminal servers can transfer data to the network at rates from 2.4 kilobits/second to 4,000 kilobits/second.

For X.25 network access, the terminal servers (called PADs or Packet Assemblers/Disassemblers in this application) use the X.25 protocol directly and conform to the X.3/X.28/X.29 packet specifications. For DDN network access, the terminal servers (called TACs or Terminal Access Controllers in this application) use the DDN X.25 Standard, 1822-LH/DH, or HDH (1822-J) attachment methods.

cisco Systems also offers SLIP servers, which extend full-function network file transfer services to personal computers and workstations that run under MS-DOS. Using this capability requires only a low-cost asynchronous serial communications port, which is standard on most personal computers today.

HyBridge for Optimum Network Connectivity

cisco Systems HyBridge combines the speed and protocol transparency of a LAN bridge with the network reliability of a **cisco Systems** gateway server (see "Gateway Servers for Building Networks of Networks", next). HyBridge performs simultaneous bridging and routing functions on the same network, processing up to 12,000 packets per second. For the bridging process, HyBridge acts as an adaptive spanning-tree bridge.

HyBridge provides internetwork connectivity: one HyBridge links many kinds of computers and network technologies, eliminating the need for several different protocol-specific or medium-specific routers. The HyBridge bridging and routing functions are user-selectable, thus protecting investments in existing LAN technology--and providing a growth path to large, complex LANs and WANs as no single-function system can.

The HyBridge interfaces can connect to network communication lines using Ethernet and high-speed links. Designed for high-traffic service, the Ethernet links can receive and generate long streams of back-to-back packets. The serial links can operate at up to 4,000 kilobits/second.

Gateway Servers for Building Networks of Networks

cisco Systems gateway servers are high-performance internetwork routers that can build networks of networks over all communication media. The gateway servers can support multiple network and routing protocols simultaneously, enabling communication among equipment and networks from different vendors. A **cisco Systems** gateway server can have up to 20 network interfaces. By "cascading" gateway servers, an organization can form networks as large and complex as necessary--including up to 100,000 subnets and millions of computers.

For more information on this **cisco Systems** product line, see Chapter 3, "Capabilities of the Gateway Server".

Committed Customer Support and Service

Customer support and service from **cisco Systems** helps organizations learn about networks and manage them effectively after their installation.

Learning About Networks

Most organizations considering the purchase of networking equipment need to make many technical and resource-allocation decisions before their network plans are complete. **cisco Systems** offers network planning guidance and advice in choosing the appropriate **cisco Systems** products to meet a variety of internetworking needs. The toll-free number to call for **cisco Systems** network planning and configuration information is:

800-553-NETS

After purchasing **cisco Systems** equipment, many organizations would like help in maintenance and problem-solving. Therefore, **cisco Systems** offers its customers round-the-clock telephone support, before and after installation. Experienced network engineers can assist callers with network planning, installation, monitoring, or troubleshooting related to **cisco Systems** products. The toll-free help-line number is:

800-553-24HR

cisco Systems also conducts seminars and classes to help train network personnel. For example, **cisco Systems** staff teaches a two-day in-depth seminar on TCP/IP for DECUS (Digital Equipment Computer Users' Society), for ACE (Advanced Computing Environments), and by invitation at major corporations. **cisco Systems** offers this seminar at its Menlo Park, California, corporate headquarters as well.

Managing Networks

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 spend time developing and installing other routes.

With **cisco Systems** dynamic routing, gateway servers 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. For more information about **cisco Systems** dynamic routing, see Chapter 3, "Capabilities of the Gateway Server."

The network management software integral to all **cisco Systems** products enables network managers to obtain detailed measurements and diagnostics for examining network performance, refining network configuration, and isolating faults. **cisco Systems** supports the Simple Network Monitoring Protocol (SNMP), the current standard for network management.

Because of the explosive growth of corporate networks, many organizations are unable to find key personnel with the critical technical expertise needed to build and maintain large networks. To fill this gap, **cisco Systems** offers Network Management Service, a custom consulting service for network planning, installation, and ongoing support.

Networking specialists from **cisco Systems** can ensure the success and cost-effectiveness of a network. Whether for a few months or for decades, **cisco Systems** is ready to plan network strategies, define and allocate resources, install and test equipment, implement a preventive maintenance program, monitor network performance, and troubleshoot problems.

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--as are operational costs well below the industry standard. Equipment failures rarely occur, because all **cisco Systems** products meet testing standards usually applied only to military systems.

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.

Modular Components for Flexible Configuration

Part of the power and flexibility of **cisco Systems** terminal and gateway servers derives from their modular physical configuration. 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. Three chassis models are available for the terminal server, and four models are available for the gateway server. All models have the Multibus-I standard backplane.

- The A chassis is a nine-slot, rack-mounted chassis for terminal servers with more than 32 lines, or for gateways in large network configurations requiring high fanouts.
- The M chassis is a mid-range, four-slot, table-top or rack-mountable chassis for medium-size terminal and gateway servers.
- The P chassis is a portable four-slot chassis, enclosed in an RFI/EMI-shielded Halliburton Zero suitcase, that is useful for temporary installations.
- The C chassis is a compact two-slot chassis, designed for remote gateways in an office or desktop environment. This model, which includes a quiet fan, is available in a limited number of gateway configurations only.

Processor Options

For high-speed operation, the terminal and gateway servers use processors based on the MC68000 or MC68020 microprocessor. Both **cisco Systems** processors contain one megabyte of RAM; system ROM holding all operating system, bootstrap, and diagnostic software; and hardware and software support for a control console.

The CSC/1 processor is based on the MC68000 microprocessor. Systems configured with the CSC/1 processor can forward more than 1,200 packets per second, and can achieve data forwarding rates of 2,000 kilobytes/second. This processor is well suited for terminal servers with 32 or fewer lines, and for light-traffic and medium-traffic gateways.

Based on the MC68020 microprocessor, the CSC/2 processor has more than twice the processing capability of the CSC/1 processor. Systems configured with the CSC/2 processor can forward more than 2,200 packets per second, and can achieve data forwarding rates of 4,000 kilobytes/second. This high-performance processor is best suited for terminal servers with 48 or more lines and/or supporting more microcomputers than terminals. It is critical for gateways handling large data volumes and for complex gateway configurations.

cisco Systems also offers optional non-volatile memory that retains configuration information despite power losses or gateway reboots. With the non-volatile memory option, the terminal and gateway servers need not rely on other network servers for configuration and boot service information.

Connector Panel Options

Each chassis model accepts connector panels in numerous formats, enabling easy configuration of terminal and gateway servers to meet current and future needs. Supported connectors include modular connectors (4-pin RJ11C and 6-pin RJ12), 25-pin RS-232C D connectors, 15-pin Ethernet connectors, 50-pin telco connectors, and V.35 connectors. The A chassis also supports the 1822 connector.

Interface Options

cisco Systems offers three interface options: the Multi-port Communications Interface, the Token Ring Interface, and the CSC-A (DDN 1822-LH/DH) interface.

The Multi-port Communications Interface 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 packet types. The serial ports support HDLC, LAPB, X.25, DDN X.25, and HDH (DDN 1822-J) transmission. With the Multi-port Communications Interface, a gateway server using the A chassis (called an AGS) can accommodate up to 10 serial ports or up to eight T1-speed circuits.

By combining multiple network ports on one interface and bypassing the central processor, **cisco Systems** sets new standards for packet-switching performance. The Multi-port Communications Interface achieves a switching rate of 12,000 packets per second, independent of packet size, for transfers between any two of its ports.

The interface processes packets rapidly, without the interframe delays typical of other Ethernet interfaces. By minimizing packet processing time, the Multi-port Communications Interface achieves high circuit utilization. Its Ethernet ports can sustain a switching rate of 12,000 packets per second when communicating with a similar high-performance interface.

The serial interfaces on the Multi-port Communications Interface support signaling rates of 2.4 kilobits/second to 4,000 kilobits/second. Each synchronous channel can perform full-rate transmission, independent of packet size or the load on adjacent channels.

The Token Ring Interface provides service to IEEE 802.5 token rings running at 4,000 kilobits/second. This interface extends network service to microcomputers in an easily expandable fashion.

The CSC-A interface, a two-board product that includes non-volatile configuration and multibus memory, supports DDN 1822-LH/DH attachments. Note that new attachments to the DDN must use DDN X.25 Standard, and thus will require the Multi-port Communications Interface instead of the CSC-A interface.

cisco Systems

Chapter 3
Capabilities of the Gateway Server

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 multi-protocol, multi-media, and multi-vendor networks.

Gateway servers from **cisco Systems** are designed to help LANs and WANs achieve interoperability and connectivity. They support equipment from all vendors over all available media. This chapter describes **cisco Systems**-supported protocols and media, as well as the capabilities of **cisco Systems** gateway servers for routing, network management, and network security.

Support for Multiple Protocols

Large organizations need the flexibility that multi-protocol networks give them to communicate with diverse hardware and software from many vendors. **cisco Systems** gateway servers support many networking protocols, as well as several specific routing protocols for compatibility with other networks. Included are protocols with conferenced standards and protocols from a variety of vendors.

One **cisco Systems** gateway server 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 de facto standard for internetworking, and is supported by more than 100 computer vendors, including IBM and Hewlett-Packard, and by all UNIX-based workstations.
- 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.
- X.25 protocols, which permit cost-effective, as-needed use of major public networks in the United States and Europe. **cisco Systems** products support both the X.25 protocol and the X.3/X.28/X.29 specifications.
- DECnet Phase IV protocol, Digital Equipment's proprietary networking protocol used on DECnet.

- XNS (Xerox Network Service) protocol, used by Xerox and other XNS vendors.
- Chaosnet protocol, used by LISP machine vendors and other organizations in the AI community.

Support for All Media

Gateways from **cisco Systems** support digital circuits at many speeds. Customers use 9.6 and 19.2 kilobits/second synchronous serial service, and 56 kilobits/second service for medium-traffic connections. For fast serial service over routes with heavy data needs, **cisco Systems** gateway servers support T1 circuits at 1,544 kilobits/second, T1C circuits at 3,100 kilobits/second, and British Telecom Megastream and CEPT DS1 circuits at 2,048 to 4,096 kilobits/second.

For convenient access to existing networks, **cisco Systems** gateway servers support a variety of electronic links: dedicated land links, satellite links, token rings, baseband and broadband coaxial cable, and economical dial-up links for backup. **cisco Systems** gateway servers are operating now on fiber optic links, reassuring news to organizations using or planning fiber optic communications. For customer convenience, **cisco Systems** markets a broad line of media adapters.

Dynamic Network Routing

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

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

In contrast, network equipment from most other vendors uses static routing tables to store information about available pathways. Thus when the network configuration changes (due to equipment additions or repairs), the network cannot take advantage of the change until the tables in all the network routers have been updated.

The Interior Gateway Routing Protocol (IGRP), developed by **cisco Systems**, monitors the network to determine the status of each route and select the best route for each data packet. Network traffic, path reliability, and 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** gateway servers can concurrently receive and understand messages from other network segments sent using different routing protocols. For example, IP routing protocols supported by **cisco Systems** include:

- *routed* or RIP (Routing Information Protocol), the interior routing protocol used by the routing process on Berkeley-derived UNIX systems.
- EGP (Exterior Gateway Protocol), the routing protocol used by all gateways attached to the DDN. The **cisco Systems** implementation maintains contact with multiple EGP-speaking gateways, preserving routing information when the DDN core gateways do not respond.
- HELLO, the primary routing protocol used by the National Science Foundation (NSF) regional networks.

On X.25 networks, **cisco Systems** gateways can send IP packets by encapsulating them within X.25 packets. The gateway then treats X.25 like any other medium and optimizes routing.

All **cisco Systems** gateway servers support the routing protocols used in DECnets, XNS networks, and Chaosnet:

- When routing DECnet packets, the **cisco Systems** gateway server acts as a DECnet Level 1 and/or Level 2 router. **cisco Systems** gateway servers support both DECnet and IP routing on the same Ethernet and serial line network segments.
- **cisco Systems** gateway servers can route XNS concurrently with IP and other **cisco Systems**-supported routing protocols.
- All **cisco Systems** gateway servers support full Chaosnet routing and a small set of Chaosnet host functions, including the status and uptime services.

Network Management Software

Most network management systems attempt to resolve routing problems caused by the shortcomings of static table-based routing, primarily an inability to respond quickly to network link changes.

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.

Gateway servers from **cisco Systems** also provide detailed network management statistics, including traffic statistics, counts of messages transmitted (complete messages, incomplete messages, and complete but incorrect messages), and many more. Remote echo diagnostics help network managers isolate faults and refine network measurements.

Network security is an increasingly important aspect of managing complex networks. **cisco Systems** gateway 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 gateways.

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

cisco Systems

Chapter 4
Setting Up the Gateway Server

This chapter describes the site requirements, additional equipment requirements, unpacking procedures, and connection details for the gateway server. This information helps you physically set up the gateway server for operation.

Site Requirements

The AGS and MGS models of the **cisco Systems** gateway server can be used as table-top or rack-mount equipment in any data processing or laboratory environment. The AGS gateway server has superior cooling and can be deployed in inhospitable environments such as phone closets or other rooms with minimal ventilation.

The portable PGS gateway server is designed for testing and maintenance use in field conditions. The CGS gateway server is suitable for an office or desktop environment, and includes a quiet fan.

The gateway server can be factory-configured for either 110-volt or 220-volt operation. All units include a 6-foot electrical power cord.

Additional Equipment Requirements

You may need some of the following data communications equipment to complete your gateway server installation. The needs of your installation depend on the interfaces you plan to use and on other factors, as described below.

To install and configure the gateway server, you need an **RS-232 terminal.** You can detach the terminal after installation and configuration are complete.

To use an IEEE 802.3 or Ethernet interface at your installation, you need an **802.3 Medium Attachment Unit (MAU)** and an **Attachment Unit Interface (AUI)** cable or an **Ethernet transceiver** and **transceiver cable.** You can purchase these devices from **cisco Systems** as additional equipment.

To use a low-speed synchronous serial interface at your installation, you need a **modem** or a **CSU/DSU** (Customer Service Unit/Digital Service Unit) to connect to the network. **cisco Systems** provides RS-232, V.35, or RS-422 as the electrical interface.

To attach a gateway server to a T1 network, you need a **T1 converter.** This device converts the HDLC synchronous serial data stream of the gateway server into a T1 data stream with the correct framing and ones density. (The term *ones density* refers to the fact that some telephone systems require a minimum number of 1 bits per time unit in a data stream.)

Note that several T1 converter-CSU/DSU combination devices are on the market. **cisco Systems** offers a T1 converter-CSU/DSU unit as additional equipment. Note also that a T1 converter provides a V.35 electrical interface to the gateway.

Unpacking the Gateway Server

A gateway server arrives in a box that contains its components and electrical connectors. When you unpack the box, check the receiving list that accompanies all shipments to ensure you received all the pieces you ordered. Inspect all items for shipping damage.

Unfasten the access panel of the gateway server and examine the cards in the system chassis. Remove any packing foam and reseat any cards that may have worked loose during shipment. To avoid static discharge damage, handle the cards by the card edges and avoid touching the components on the card.

If the chassis or any of the cards appears damaged, or if you have any problems installing or configuring your gateway server, call the **cisco Systems** help line at any time. The phone number is 800-553-24HR.

Contents of the AGS Boxes

Each AGS is shipped with the following items:

1. A chassis.
2. A power cord.
3. A rack-mounting kit, including screws and two flanges (you can rack-mount the chassis in a standard 19-inch rack).
4. The *Gateway System Manual.*
5. The central processor card.
6. Up to eight network interface cards.
7. Optionally, a non-volatile memory card.

Contents of the MGS Boxes

Each MGS is shipped with the following items:

1. A chassis.
2. A power cord.
3. A rack-mounting kit, including screws and two flanges (you can rack-mount the chassis in a standard 19-inch rack).
4. The *Gateway System Manual*.
5. The central processor card.
6. Up to three network interface cards.
7. Optionally, a non-volatile memory card.

Contents of the PGS Boxes

Each PGS is shipped with the following items:

1. A chassis enclosed in a Halliburton Zero suitcase enclosure.
2. A power cord, which may be in the compartment near the lock.
3. The *Gateway System Manual*.
4. The central processor card.
5. Up to three network interface cards.
6. Optionally, a non-volatile memory card.

Contents of the CGS Boxes

Each CGS is shipped with the following items:

1. A chassis.
2. A power cord.
3. The *Gateway System Manual*.
4. The central processor card.
5. The network interface card.

Making Connections to the Gateway Server

The **cisco Systems** gateway servers employ a modular system of connector panels. The number and types of connectors installed on your gateway server depend on the number of interface cards installed and the connection options chosen. The back panels of your gateway server can provide 25-pin RS-232C D-type connectors, 15-pin Ethernet transceiver D-type connectors, V.35 connectors, and DDN 1822 connectors. The positions of these connectors vary with the panel types used; the following sections describe the connectors for each of the gateway server models.

Each gateway server includes a system console port wired as a DCE (Data Communications Equipment) device. The default parameters for this port are 9600 baud, eight data bits, no parity generated or checked, and two stop bits. You can specify other terminal speeds using processor jumpers as described in "The Processor Configuration Register" in Chapter 8, "Configuring the Gateway Server Processors".

To prepare for the initial startup and configuration, attach an RS-232 ASCII terminal to the system console port. Flow control is not possible on the console port; however, you can specify padding for output characters with the EXEC command **terminal pad** described in "Setting Terminal Parameters" in Chapter 5, "Starting Gateway Server Operations".

Making Connections to the AGS

Viewed from the rear, the power cable and power switch (which is also a circuit breaker) of the AGS appear on the left side. The system console port, Ethernet ports, and high-speed serial line ports (if any) appear on connector panels above and to the right of the power cable and switch.

The topmost RS-232C connector (above the power cable and switch) is the system console port.

Ethernet ports are located either under or to the right of the console port. The topmost port is labeled *ethernet 0*, the next port (if any) is labeled *ethernet 1*, and so on. All standard 15-pin Ethernet transceiver cables and IEEE 802.3 AUI cables mate with these connectors.

Serial interface ports are located either under the console port or to the right of the console port. The topmost port is labeled *serial 0*, the next port is labeled *serial 1*, and so on. T1-rate ports appear as either V.35 or RS-232 connectors mounted on individual back panel plates.

Making Connections to the MGS

Viewed from the rear, the power cable and power switch (which is also a circuit breaker) of the MGS appear on the right side. The system console port, Ethernet ports, and high-speed serial line ports (if any) appear on connector panels to the left of the power cable and switch.

The topmost RS-232C connector to the left of the left exhaust fan is the system console port.

Ethernet ports are located in the connector panels on the left. The ports are labeled *ethernet 0*, *ethernet 1*, and so on. All standard 15-pin Ethernet transceiver cables and IEEE 802.3 AUI cables mate with these connectors.

Serial interface ports are located adjacent to or in place of the Ethernet ports. The ports are labeled *serial 0*, *serial 1*, and so on. T1-rate ports appear as V.35 connectors mounted on individual back panel plates.

Making Connections to the PGS

The power cable, power switch, console port connector, and network interface connectors are inside the Zero suitcase on the top panel of the PGS chassis. Viewed from the top, the system console port and the Ethernet port are just below the **cisco Systems** logo. The leftmost 25-pin connector is the console port. The leftmost Ethernet interface port (if any) is labeled *ethernet 0*, the next port (if any) is labeled *ethernet 1*, and so on. All standard 15-pin Ethernet transceiver cables and IEEE 802.3 AUI cables mate with these connectors.

Serial interface ports are located to the right of the console port and any Ethernet ports. The leftmost port is labeled *serial 0*, the next port is labeled *serial 1*, and so on.

Making Connections to the CGS

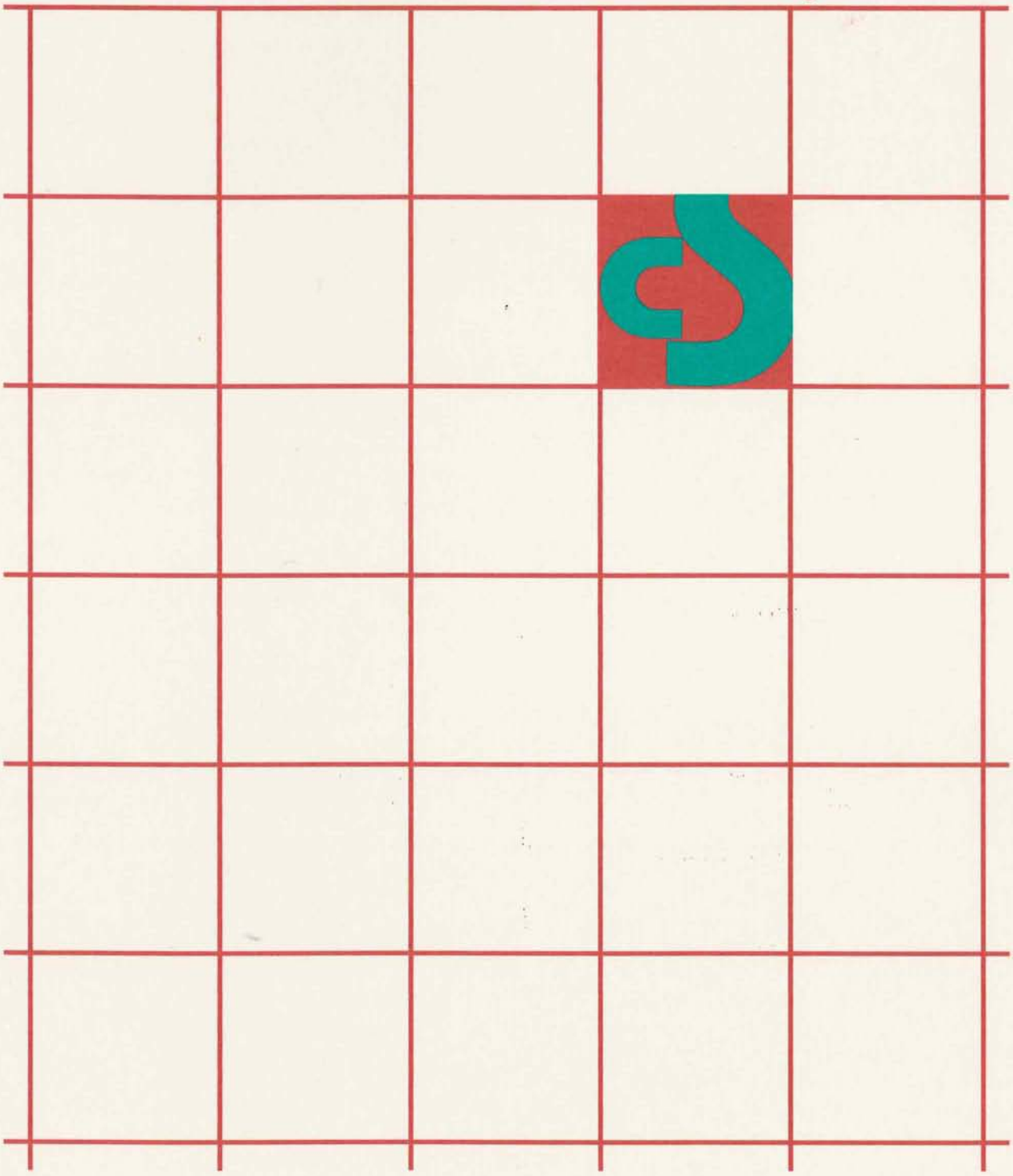
Viewed from the rear, the power cable and power switch (which is also a circuit breaker) of the CGS appear on the right side. The system console port, Ethernet ports, and high-speed serial line ports (if any) appear on connector panels to the left of the power cable and switch.

The topmost RS-232C connector to the left of the exhaust fan is the system console port.

Ethernet ports are located in connector panels on the left. The ports are labeled *ethernet 0* or *ethernet 1*. All standard 15-pin Ethernet transceiver cables and IEEE 802.3 AUI cables mate with these connectors.

Serial interface ports are located adjacent to or in place of the Ethernet ports. The ports are labeled *serial 0* or *serial 1*. The T1-rate port appears as either an RS-232 or a V.35 connector mounted on an individual back panel plate.

cisco Systems



Chapter 5
Starting Gateway Server Operations

This chapter provides the information you need to begin gateway server operations. It includes instructions for starting and configuring the gateway server, and prepares you to use the EXEC command interpreter, Telnet communications, and terminal parameter-setting commands. The chapter contains sample configurations that illustrate typical gateway server configurations. You can use these samples to begin your own gateway server configuration.

Preparing for Startup

Figure 5-1 is a simplified flowchart of the first-time gateway server startup sequence. This flowchart shows your actions on the left and the gateway server actions on the right. The following sections describe each of the steps in the flowchart.

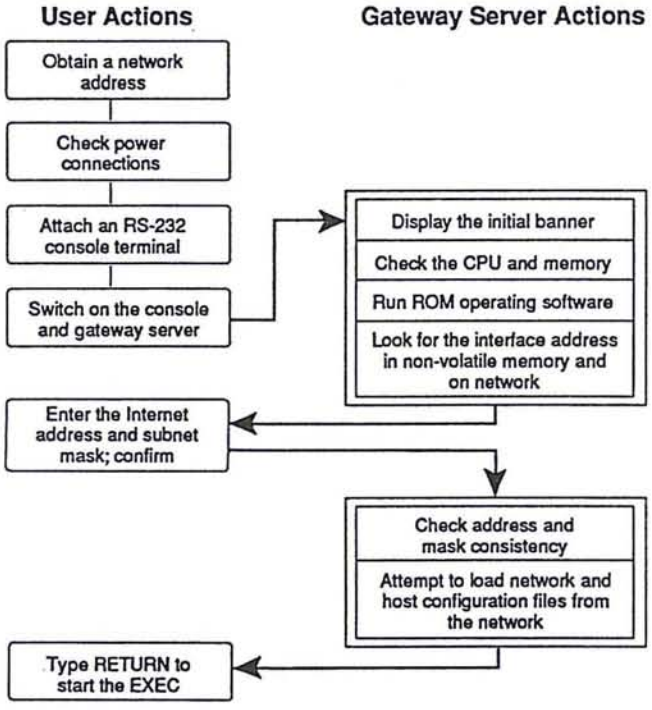


Figure 5-1. Simplified Flowchart of First-Time Gateway Server Startup

Obtaining a Network Address

When you start up the gateway server for the first time, you must supply it with an Internet address for each network port. Internet addresses are 32-bit numbers written in "dotted-decimal" format: X.X.X.X, where "X" is a number between 0 and 255. The gateway server must have an address to communicate with the network. For a complete description of Internet addresses, see "Understanding Address Classes and Formats" in Chapter 11, "Understanding Internet Addresses, Broadcasts, and Address Resolution". If you do not know what address to use for the gateway server, consult your network manager.

Checking Power Connections

The gateway server is set at the factory for either 110-volt or 220-volt power. A label near the power cord indicates the correct voltage for your unit. If the voltage indicated on the label is different from the power outlet voltage, do not plug in the gateway server. A voltage mismatch may cause equipment damage and pose a fire hazard.

Starting Up the Gateway Server

Now you are ready to start up the gateway server and put it into operation.

Switching On the Console and the Gateway Server

Attach an RS-232 ASCII terminal to the system console port and turn on the terminal. The default console port settings are 9,600 baud, eight data bits, no parity generated or checked, and two stop bits. If your console terminal does not use these defaults, see "The Processor Configuration Register" in Chapter 8, "Configuring the Gateway Server Processors," to change the terminal speed.

Plug the power cord from the chassis into a power outlet, and turn on the circuit breaker power switch. You should hear noise from the cooling fan, which should be up to speed within five seconds.

If you do not hear the fan noise, check the outlet for power. If the circuit breaker trips, you have a power problem either with your outlet or with the gateway server. If the problem appears to be with the gateway server, contact **cisco Systems**.

Observing the First-Time Startup and Self-Test

When you first turn on the power, the gateway server displays a one-line banner message on the terminal screen. For example:

```
System Bootstrap, Version 3.1(1), copyright (c) 1987, cisco Systems, Inc.
```

If no message appears, you may have a problem with the terminal or with the processor card itself. To check for a problem with the processor card, open the front panel and look at the processor card. Both the CSC/1 and CSC/2 processors have a halt indicator that lights if the processor encounters a fatal software error.

On the CSC/1 processor card, the halt indicator is the red LED at the extreme left of the card, as viewed through the panel opening. On the CSC/2 processor card, the halt indicator is the middle (red) LED in the group of three LEDs next to the console cable attachment.

If you find the halt indicator lit, or if you suspect some other problem with the processor card, contact **cisco Systems.**

After displaying the first banner line, the gateway server performs a CPU and memory check. If this self-test finds problems, diagnostic messages appear on the console display. Write down any such messages and contact **cisco Systems.** If the self-test finds no problems, the gateway server displays a second banner line announcing the processor type and memory size.

Assuming no problem messages, a typical first-time startup sequence for a gateway server with two Ethernet interfaces looks like this:

```
System Bootstrap, Version 3.1(12), copyright (c) 1987, cisco Systems, Inc.
CSC/2 processor with 1024 Kbytes of memory
################################################################
################################################################
#########
F1: 234387+95321+27540 at 0x1000

System Version 6.1(463), compiled Tue 01-Dec-87 17:00
Copyright (c) 1987, cisco Systems, Inc.

CSC/2 (68020) processor with 1024K bytes of memory.
2 Ethernet/IEEE 802.3 interface.
32K bytes of non-volatile configuration memory.

For interface Ethernet #0
Enter Internet address of the form "A.B.C.D":
```

Entering the Internet Address and Subnet Mask

As the previous example shows, the last line of the startup sequence prompts you to enter the Internet address of the first Ethernet interface. Type in the address and press the RETURN key. The gateway server then prompts:

```
Enter net and subnet mask [255.255.255.0]:
```

The net and subnet mask controls whether or not subnetting is in effect on that network. The Internet address in brackets is the default value that the system uses if you simply press RETURN. The default value provides no subnetting. If you are unsure whether you want subnetting, use the default.

Confirming the Internet Address

After you enter the subnet mask or accept the default, the gateway server prompts you to confirm the information you have entered. Suppose you had entered the Internet address 192.31.7.18 and accepted the default subnet mask of 255.255.255.0. The prompt would read:

```
Set address 192.31.7.18 with net mask 255.255.255.0? [confirm]
```

Press RETURN to set the address. If you type anything else, the gateway server returns to its prompt for an Internet address.

After you confirm the entered information, the gateway server carefully checks the Internet addresses you specified. It rejects any addresses and masks that it identifies as illegal. The gateway server enforces consistency among subnet masks on all interfaces on the same network.

Observing the System Boot

If the interface address is valid, the gateway server displays messages like these:

```
Booting network-confg ... [timed out]
Booting gateway-confg ... [timed out]
Press RETURN to get started!
```

These messages mean that the gateway server first tried to load a configuration command file ("network-confg") common to all **cisco Systems** hosts on the network. It then tried to load a configuration file ("gateway-confg") specific to the host. In this example, both file-loading attempts failed (timed out) because either the files or the network was not available.

As the last line in the example indicates, you start the gateway server operation by pressing RETURN. This action causes the gateway server to display the system prompt "Gateway>", which is the prompt of the EXEC command interpreter.

Using the EXEC Command Interpreter

The "Gateway>" prompt indicates that the gateway server is running the EXEC command interpreter. This command interpreter provides the commands you use to configure and operate the gateway server and its interfaces.

Throughout this manual, EXEC command keywords are in **boldface** and values you supply are in *italics*.

Command Usage

The EXEC accepts commands in uppercase, lowercase, or mixed-case letters. You may abbreviate commands and other keywords to the number of characters that cause the command to be a unique abbreviation.

If you make a typing mistake, you can erase characters with the DELETE or the BACKSPACE key. Press either key to erase the last character you typed. To erase the last word you typed, press CTRL-W; to erase the entire line, press CTRL-U. Press CTRL-R to redisplay a line in its entirety.

The gateway server acts on most commands after you press the RETURN key. For multi-line commands, you press CTRL-Z instead. To abort a command at any time, press CTRL-C.

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 EXEC command level.

Accessing Privileged Commands

If you type the **enable** command, you gain access to privileged commands. These commands change gateway server operating parameters; general network users should not use these commands. To indicate the access to privileged commands, the EXEC prompt changes to the gateway server host name followed by a pound sign (#).

On the console terminal, you can use the **enable** command without a password. However, on virtual terminals connected to the gateway server, you must give a password to use the **enable** command.

The password you use is that for line 0, the console line. When shipped from the factory, the gateway server has no password set for line 0. Set a password for line 0 before you try to use the **enable** command from a virtual terminal. See the **password** command description in "Setting Line Parameters" in Chapter 9, "Configuring the Console and Virtual Terminals".

To end privileged mode operation, type the **disable** command. This command returns the EXEC to its normal, unprivileged command mode.

Getting Help Information

To list available EXEC commands, use the ? (question mark) command. In the normal, unprivileged mode, the ? command produces this list:

```
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>   Synonym for connect command
terminal        Change terminal's parameters, type "terminal ?"
where           Show open connections
<cr>            To resume connection

Gateway>
```

In the privileged mode, the ? command produces this list (the complete gateway server command set):

```
Gateway>enable

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
disconnect <cn> Break the connection specified by name or number
disable         Turn off privileged commands
enable          Turn on privileged commands
exit, quit      Exit from the EXEC
name-connection Give a connection a logical name
ping            Send ICMP Echo message
reload          Halt and reload system
resume          Make the named connection be current
send <line> |*  Send message to a terminal line or lines
show <cmd>      Information commands, type "show ?" for list
systat          Show terminal lines and users
telnet <host>   Synonym for connect command
terminal        Change terminal's parameters, type "terminal ?"
undebug         Disable debugging functions, type "undebug ?" for list
where           Show open connections
write           Write configuration memory, type "write ?" for list
<cr>            To resume connection

Gateway#
```

Exiting the EXEC

To stop the EXEC command interpreter, type **exit** or **quit**. The terminal returns to the idle state.

If you issue no commands to the EXEC interpreter for 10 minutes, the EXEC automatically displays the "vacant message" (if any) and returns the terminal to the idle state. However, if there is a current Telnet connection, the EXEC remains active indefinitely. You may alter the two connection timers; see "Setting Line Parameters" in Chapter 9, "Configuring the Console and Virtual Terminal Lines."

The vacant message is user-defined; see "Setting Line Parameters" in Chapter 9.

Using Telnet Communications

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

Making a Telnet Connection

You can start a Telnet connection by typing a host name or a dotted-decimal Internet address at the EXEC prompt. If the host name you want to use conflicts with a gateway server command name, precede the host name or Internet address with the command **connect** or **telnet**. The gateway server automatically numbers connections for you; several commands use these numbers to identify connections.

If you use a host name, the gateway server must first find the corresponding Internet address. To find this address, the gateway server searches its host-name-to-address cache. If the name is not in the cache, the gateway server uses a dynamic name lookup method (see "Host-Name-to-Address Conversion" in Chapter 13, "Using Other Internet Services"). This method enables the gateway server to query a set of server hosts for the address.

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

To abort a Telnet connection attempt, type the escape sequence (by default, CTRL-^, followed by the letter X).

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 gateway server uses the default Telnet server port, port number 23 (decimal).

Managing Telnet Connections

The gateway server provides an escape sequence with which you can leave a Telnet connection without terminating it and return to the EXEC interpreter. The default escape sequence is a two-character combination: CTRL-^, followed by the letter X. You can change the first part of the escape sequence, the "escape character", with the **terminal** command; see "Setting Terminal Parameters" later in this chapter.

The gateway server supports multiple concurrent Telnet connections. The only limits to the number of concurrent Telnet sessions are the total memory of the system and your ability to keep track of the connections.

In commands, you identify Telnet sessions by connection number or connection name. By default, the name of a connection is the same as that of the host to which it connects. To set another name, use **name-connection** *number*, where *number* is a connection number; **name-connection** prompts you for a new connection name.

To display the name and number of each current Telnet connection, use the **where** command.

To return to the most recently used connection, simply type RETURN or the **resume** command. To resume using a particular Telnet connection, type **resume** *identifier*, where *identifier* is the connection name or number. You can display a list of connection names and numbers using **resume ?**.

To close a particular connection, log off the host computer and let the host close the Telnet connection. You can also use **disconnect** *identifier*, where *identifier* is the connection name or number.

If a connection hangs, preventing any data transfer, use the privileged **clear line** *line-number* command, where *line-number* is a line number. The command closes all connections, terminates associated processes, and resets the data structures associated with the terminal line.

If you have open Telnet connections and use the **quit** or **exit** command to stop the EXEC, the EXEC prompts for confirmation to close the connections.

If you do not have a current Telnet connection and you do not issue a command for 10 minutes, the EXEC automatically displays the "vacant message" and returns the terminal to the idle state.

The Gateway Server as a Telnet Server

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

The user of an incoming Telnet connection can gain access to the privileged EXEC commands through the **enable** command, which requires a password. (See "Using the EXEC Command Interpreter" earlier in this chapter.) With access to the complete EXEC command set, the incoming connection acts as a remote console connection. A remote console connection provides a convenient way to monitor and adjust gateway server operation.

You can control access to the gateway server with access lists; see Chapter 10, "Adding Security Measures".

The gateway server supports the following Telnet options:

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

Setting Terminal Parameters

This section describes how to change terminal parameters for the current EXEC session. To make long-term configuration changes, see Chapter 9, "Configuring the Console and Virtual Terminals".

If the default escape character interferes with communications to the systems on your network, change the escape character with **terminal escape-character** *character*. The argument *character* is either the ASCII decimal equivalent of the character you want as the escape character, or else the character itself. You cannot use BREAK as the escape character on the gateway server console.

To set the number of lines on the terminal screen, use **terminal length** *screen-length*. The gateway server uses the screen length to determine when to pause during multiple-screen output. The argument *screen-length* is the desired number of lines. The default length is 24 lines. A value of 0 disables pausing between screens of output.

When you have multiple concurrent Telnet connections, you may want to know when output is pending on a connection other than the current connection. For example, you may want to know when another connection receives mail or an asynchronous message. To tell the gateway server to notify you of pending output, use the **terminal notify** command. Use the **terminal no notify** command to end such notifications.

To set character padding on your terminal, use the **terminal pad** command. This command takes two arguments, *decimal-number* and *count*. The argument *decimal-number* is the ASCII decimal representation of a character, and the argument *count* is the number of NUL bytes sent after that character. To end padding after the character represented by *decimal-number*, type **terminal no pad** *decimal-number*.

By default, output from the **debug** command is sent to the console terminal. (See Chapter 18, "Network Monitoring and Troubleshooting", for more information.) To copy the **debug** output to a non-console terminal, use the **terminal monitor** command. Use the **terminal no monitor** command to end this copying. When you quit your terminal session, the EXEC automatically performs the **terminal no monitor** action.

Sending Messages

To send a single or multi-line message, use the privileged **send** command. To send the message to a specific connection, type the connection number after **send**. To send the message to all terminals connected to the gateway server, type an asterisk (*) after **send**. The EXEC prompts you for the message, which you terminate with CTRL-Z. To abort the command, type CTRL-C.

Specifying Configurations

In addition to EXEC commands, the gateway server has several configuration commands. You can use these commands to adjust and optimize your system as required.

This section describes configuring the gateway server from the console terminal and from a configuration file. For information on automatic configuration, see Chapter 6, "Advanced Gateway Server Configuration".

As for EXEC commands, this manual shows configuration command keywords in **boldface** and values you supply in *italics*.

Configuring From the Console Terminal

To issue configuration commands from the console terminal, first enter the privileged command mode by using the EXEC command **enable**. Then enter the EXEC command **configure**. The gateway server responds with a prompt asking you to specify the terminal, a file, or non-volatile memory as the source of configuration data. At this point, press RETURN or type "terminal" to start configuration command collection.

During command collection, the gateway server accepts one configuration command per line. As with EXEC commands, you can use uppercase, lowercase, or mixed-case letters, and you can shorten all command keywords to unique abbreviations.

You can use DELETE or BACKSPACE to erase a character, CTRL-W to erase a word, and CTRL-U to erase an entire line. To redisplay all commands entered since the **configure** command, type CTRL-R. To abort the **configure** command, type CTRL-C. Otherwise, command collection continues until you type CTRL-Z.

When you type CTRL-Z, the gateway server executes the configuration commands it has collected. The gateway server does not display confirmation messages as it executes the commands. If the gateway server encounters a problem, it displays an error message on the console terminal.

In most cases, you can negate a configuration command by typing **no** before the command keyword. You can usually omit the arguments of the command when you negate it with **no**. See Appendix B, "Configuration Command Reference", for exceptions to these rules.

The following listing shows an example of what you would see when configuring the gateway server from the console terminal. This example starts configuration command collection, sets a password for the console line, and changes the system host name from "gateway" to "Chaff". (User-supplied information is in boldface.)

```
gateway>enable
gateway#configure
Configuring from terminal, memory, or network [terminal]? terminal
Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U; end with CTRL/Z; abort with CTRL/C:
line 0
password changeme
hostname Chaff
^Z

Chaff#
```

Configuring From a File

To avoid typing configuration commands at a console terminal each time the system starts up, you can use a configuration file. A configuration file is a plain text file containing gateway server configuration commands and comments (as desired). You can prepare a configuration file at any host connected to the network.

To place a comment in a configuration file, begin the line with the keyword **comment** or one of the symbols "!", "#", or ";". Follow the keyword or symbol with a space or RETURN. The gateway server ignores all text between the space and the end of the line.

To end a configuration file, use the **end** keyword. The gateway server displays an error message if it encounters the end of the file before the **end** keyword.

The following listing shows an example of configuring the gateway server from a configuration file on the network.

```
Chaff>enable
Chaff#configure
Configuring from terminal, memory, or network [terminal]? network
Host or network configuration file [host]?<RETURN>
IP address of remote host [255.255.255.255]?<RETURN>
Name of configuration file [chaff-confg]?<RETURN>
Configure using chaff-confg from 255.255.255.255? [confirm]<RETURN>
Booting chaff-confg from 192.31.7.19: !!! [OK]<RETURN>

Chaff#
```

Specifying "network" in the third line initiates the process of loading a file from a host on the network. The gateway server uses TFTP (Trivial File Transfer Protocol) to transfer the file. Most workstations and hosts that offer UNIX and TCP/IP also offer TFTP; contact the vendor of your host computer if you need more information on TFTP support.

The responses to all the remaining prompts in the example accept the default actions. Thus, the configuration file is to be host-specific rather than a generic network file, the gateway server is to accept the configuration file from any host on the network, and the configuration file name to look for is "chaff-confg".

Pressing RETURN at the final prompt confirms these selections and starts the TFTP transfer process.

During the transfer process, the gateway server displays a message and an exclamation point (!) for each data packet received. If the booting attempt is successful, the gateway server displays "[OK]".

If the booting attempt fails, the gateway server displays a period (.) and tries again. After three successive failures, the gateway server gives up. If the gateway server gives up, look for a host-related problem such as a nonexistent or unreadable file, or a misconfigured TFTP server.

Sample Configurations

This section presents sample configuration files for some typical gateway server configurations. These configuration files do not apply to all situations; they are only examples which illustrate the configuration process. To use these examples, you must modify and perhaps add certain commands to correctly specify the configuration of your network.

The configuration commands used in the sample files are described in later chapters. Appendix B, "Configuration Command Reference", summarizes all configuration commands.

Note that the configuration commands in the sample files are also valid when entered at the console terminal. See "Configuring From the Console Terminal" earlier in this chapter.

Types of Files

There are two types of configuration files: the network configuration file and the host configuration file. The network configuration file contains configuration parameters common to all **cisco Systems** gateway servers and terminal servers on a network. The host configuration file contains information specific to a gateway server or terminal server host.

You can place configuration commands in either or both of the configuration file types. Duplicating configuration information between the network and the host configuration files does not present a problem. Furthermore, you can place all configuration commands in the host configuration file, reducing the network configuration file to just the **end** keyword.

However, if conflicting configuration information exists in the network and the host configuration files, the information in the host file takes precedence. This fact allows you to configure all network hosts in general and adjust for specific hosts as necessary.

When configuring a gateway server, you may want to use the EXEC command **write terminal** to review the current configuration. The output of **write terminal** is similar to the examples in this section. Note that the **write terminal** output does not show default settings. Note also that the privileged EXEC command **show configuration** displays only default settings, not current configuration settings.

In the sample files, comment lines begin with "!", followed by a space and some text or simply by RETURN (to leave a blank line and improve readability). The network and host addresses are examples only; substitute your own network and host addresses if you use these configuration files as models.

Network Configuration File

The following listing shows a sample network configuration file.

```
! A sample "network-confg" file.
!
! IGRP is our routing protocol.  Network 192.31.7.0
! is the network we are routing.
!
router igrp 109
network 192.31.7.0
!
! We have a static route to network 26.0.0.0.
!
route 26.0.0.0  192.31.7.67
!
! Whenever network 10.0.0.0 appears in our routing table, we send otherwise
! unroutable packets toward those gateways that know about net 10.0.0.0.
! This approach has the effect of setting up a "dynamic" default gateway.
!
default-network 10.0.0.0
!
```

```
! We define two host-name-to-address bindings.  The cisco hosts may use
! these names to determine their host names (and hence their
! host configuration file names) after they load this network
! configuration file.
!
host fred 192.31.7.20
host wilma 192.31.7.21
!
! We define our default domain name and enable use of the Domain Name
! System. We also define two name servers, 192.31.7.19 and the Internet
! broadcast address.
!
domain-name CISCO.COM
service domain
name-server 192.31.7.19  255.255.255.255
!
end
```

Ethernet-to-Serial-Link Host Configuration File

The following listing shows a sample host configuration file for a gateway server with two Ethernet interfaces and one serial line interface (see Figure 5-2). The serial line interface in this example can be the Multi-port Communications Interface, the CSC-S synchronous serial interface, or the CSC-T synchronous serial interface.

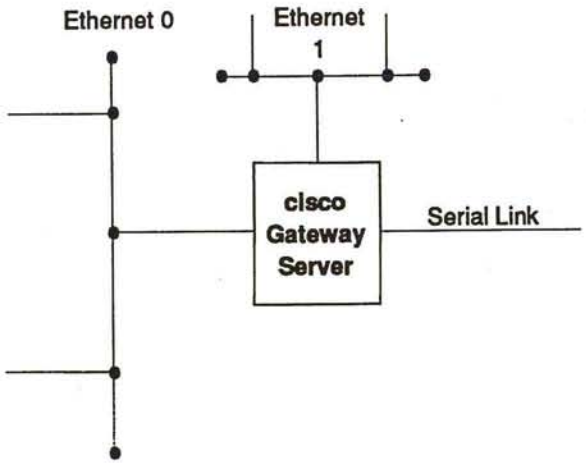


Figure 5-2. Ethernet-to-Serial Link

The serial line interface uses the default HDLC encapsulation method, which includes a special address initialization feature. By setting the "serial 0" address so that the host portion is 1, the host at the other end of the serial line can initialize with a host address of 2. See "Determining Interface Addresses" in Chapter 6, "Advanced Gateway Server Configuration", for a complete description.

Because there is no server host on a serial line, this file specifies a helper address for "serial 0". See "Selecting and Setting Broadcast Information" in Chapter 7, "Configuring the Network Interfaces", for more information.

```
! Ethernet-to-serial-link host configuration
!
interface Ethernet 0
address 192.31.7.26 255.255.255.240
!
interface Ethernet 1
address 192.31.7.34 255.255.255.240
!
interface Serial 0
address 128.89.55.1 255.255.255.0
helper-address 192.31.7.31
!
! We define two networks for which IGRP routing is done.
!
router igrp 109
network 192.31.7.0
network 128.89.0.0
!
hostname dross
!
! We use non-default names for the network and host configuration files
! as well as for the actual boot image.  Non-volatile memory is required
! for these configuration commands to be useful.
2
!
boot network local-confg 255.255.255.255
boot host dross-confg 255.255.255.255
boot system cisco-image 255.255.255.255
!
! We set a password on the console line and then require that the
! password be given before starting an EXEC command interpreter.
!
line 0
password mumblez
login
!
end
```

Ethernet-to-Ethernet Host Configuration File

The following listing shows a sample host configuration file for a gateway server that connects two Ethernet networks (see Figure 5-3). In this example, the two Ethernet networks are subnets of a larger Internet network.

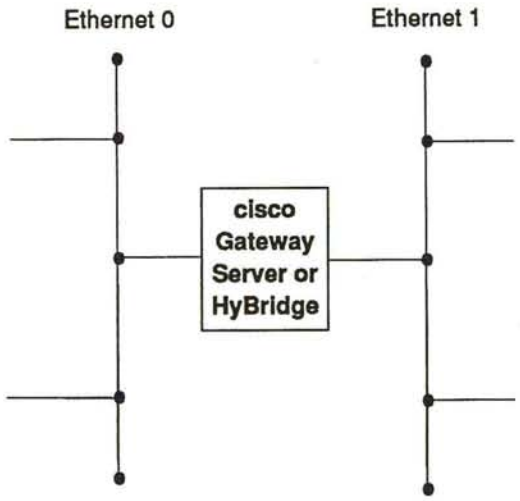


Figure 5-3. Ethernet-to-Ethernet Link

The **address** commands in this example specify that the third octet of addresses is a subnet identifier. See Chapter 11, "Understanding Internet Addresses, Broadcasts, and Address Resolution", for a complete description of addresses and subnetting.

```
! Host configuration file for a two-Ethernet gateway server.
!
! We first define the interface addresses for Ethernet 0 and Ethernet 1.
! The subnet mask must be the same for both interfaces onto
! network 128.89.0.0.
!
interface ethernet 0
address 128.89.5.1 255.255.255.0
interface ethernet 1
address 128.89.9.10 255.255.255.0
!
! We define RIP (aka "routed") as our routing protocol for network 128.89.0.0.
!
router rip
network 128.89.0.0
!
! We define a host-name string.  This string is used for the EXEC prompts and
! in constructing the default host configuration file name.  It is not
! associated with any particular Internet address, as the host command is.
!
hostname Polar-GW
!
! We set a password on line 0, the console terminal line.  This password is
! the same password that the EXEC enable command requires.  We also set a
! vacant terminal message for the console line.
!
line 0
password changeme
vacant-message #
Greenland Icecap Network - Polar Gateway
#
!
end
```

Ethernet-to-X.25 Host Configuration File

The following listing shows a sample host configuration file for a gateway server that connects an Ethernet network and an X.25 network (see Figure 5-4).

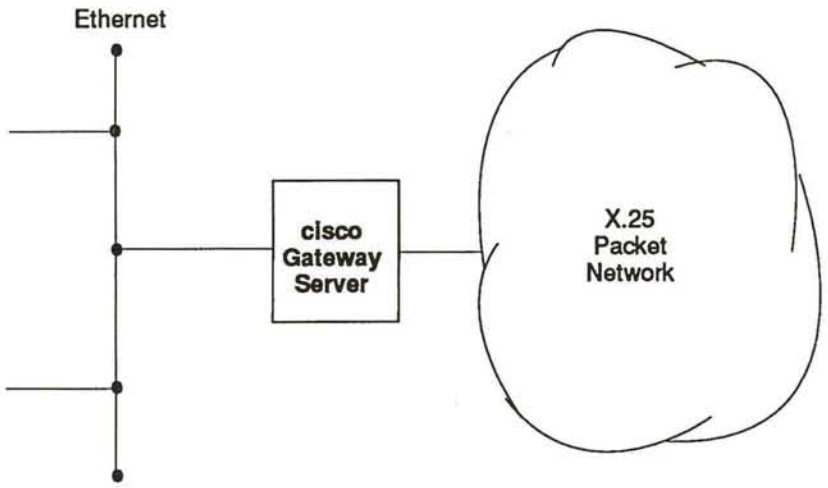


Figure 5-4. Ethernet-to-X.25 Link

```
! Ethernet-to-X.25 host configuration
!
! Note that the appropriate X.25 parameter settings vary greatly among X.25
! networks.
!
interface Ethernet 0
address 192.31.7.21 255.255.255.240
!
! Serial 0 is acting as an X.25 DTE.  We define the X.121 address of the
! interface, set an idle timeout of 10 minutes for idle SVCs, and allow
! up to four SVCs to one destination.  We also define one mapping
! between an Internet and an X.121 address.
!
interface Serial 0
address 1.2.3.4 255.0.0.0
encapsulation X25
x25 address 000000010000
x25 idle 10
x25 nvc 4
x25 map ip 1.0.0.2 000000020000 BROADCAST
!
! We do not automatically look up a configuration file on powerup;
! all our configuration information is stored in non-volatile memory.
!
no service config
!
! In addition to setting a password on the console line, we clear the EXEC
! timeout so that the EXEC never exits.
!
line 0
password test
no exec-timeout
!
! For our virtual terminals (lines 1 through 5), we specify an access list
! to control which hosts are allowed to connect.
!
line 1 5
access-class 5 in
!
! We now define the access list just specified to allow connections from any
! host on network 192.31.7.0 and from host 128.99.0.5.  No one else may
! connect to our virtual terminals.
!
access-list 5 permit 192.31.7.0  0.0.0.255
access-list 5 permit 128.99.0.5
!
end
```

Ethernet-to-DDN X.25 Host Configuration File

The following listing shows a sample host configuration file for a gateway server that connects an Ethernet network and a DDN X.25 network (see Figure 5-5).

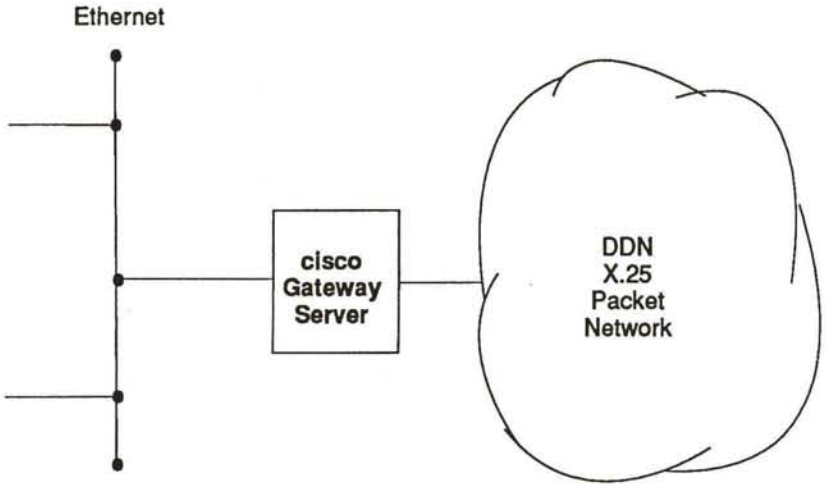


Figure 5-5. Ethernet-to-DDN X.25 Link

```
! Ethernet-to-DDN X.25 host configuration
!
! This configuration specifies EGP routing and defines a list of EGP neighbors.
! The file sets some X.25 parameters, but leaves most at their DDN default
! values.
!
interface ethernet 0
8address 128.89.65.18 255.255.255.0
!
interface serial 0
address 10.8.0.132 255.0.0.0
encapsulation ddnx25
!
! We prepare some EGP definitions.  We define our autonomous system number.
! We also define a list of EGP primary (core) gateways and keep at most two
! of them active at any one time.
!
autonomous-system 109
primary-neighbors 2
egp-neighbor 10.3.0.27  109 primary      ! gateway.isi.edu
egp-neighbor 10.2.0.37  109 primary      ! purdue-cs-gw.arpa
egp-neighbor 10.7.0.63  109 primary      ! bbnnet2-arpanet-gw.arpa
!
! We start an EGP router for both the DDN network and our local network.
!
router egp
network 10.0.0.0
network 128.89.0.0
!
! We use IGRP for our internal routing protocol.
!
router igrp 109
network 128.89.0.0
!
! We set a password on line 0, the console terminal line.
!
line 0
password yowzer
!
! We set a password on the virtual terminals and require that it
! be given before starting an EXEC process.  The line 0 password
! will still be needed to enter the privileged console mode.
!
line 1 5
password yessir
login
!
end
```

Ethernet-to-DDN 1822 Host Configuration File

The following listing shows a sample host configuration file for a gateway server that connects an Ethernet network and a DDN 1822 network (see Figure 5-6).

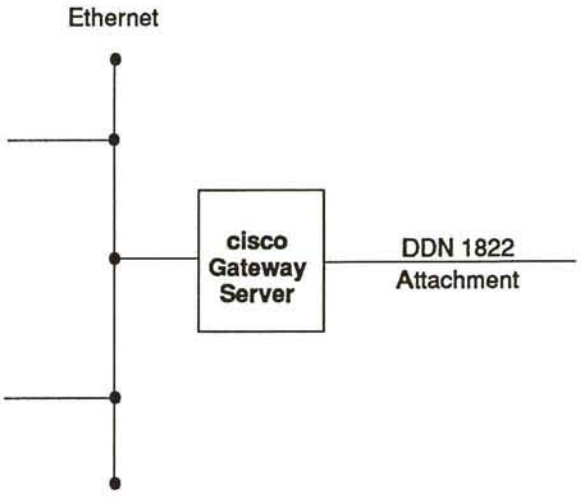
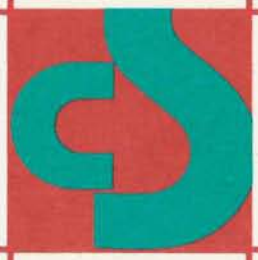


Figure 5-6. Ethernet-to-DDN 1822 Link

```
! Ethernet-to-DDN 1822 host configuration
!
interface Ethernet 0
address 192.31.7.1  255.255.255.0
!
! We define the DDN interface address.  Note that we also specify an
! access list to control traffic through this interface.
!
interface DDN 1822 0
address 10.8.0.131  255.0.0.0
access-group 1
!
! We always look up our configuration files from a server host.
! We also disable the use of Probe on our network.
!
service config
no service probe
!
! We define a useful hostname-to-address mapping.
!
host ARPAWAY 10.8.0.131  192.31.7.1

!
! We prepare some EGP definitions.  We define our autonomous system
! number.  We allow up to two active EGP peers at one time.  We define a
! list of EGP peers (core gateways) from which to obtain EGP information.
!
autonomous-system 109
primary-neighbors 2
egp-neighbor 10.3.0.27 109 primary
egp-neighbor 10.2.0.37 109 primary
egp-neighbor 10.7.0.63 109 primary
!
! We start an EGP router for networks 10.0.0.0 and 192.31.7.0.
!
router egp
network 10.0.0.0
network 192.31.7.0
!
! We define an access list that applies to the DDN interface.  By allowing
! only a few hosts to connect to the DDN, we limit our possible
! security exposure.
!
access-list 1 permit 10.8.0.131
access-list 1 permit 192.31.7.1
access-list 1 permit 192.31.7.2
access-list 1 permit 192.31.7.8
!
! We set a password on line 0, the console terminal line.
!
line 0
password tediumm
!
end
```

cisco Systems



Chapter 6
Advanced Gateway Server Configuration

The gateway server performs a sequence of operations when you switch on the power or reboot. After performing CPU and memory checks, the gateway server attempts to determine the Internet addresses of installed interfaces, load operating software from the network (if instructed to do so), and load configuration information from non-volatile memory and/or from the network.

This chapter describes the last three startup operations in detail. These operations provide fully automatic startup and configuration for the gateway server. The chapter also describes how to save and examine configuration information in non-volatile memory, and how to instruct the gateway server to act as a network host.

Determining Interface Addresses

The gateway server cannot complete its startup sequence until it identifies the Internet address of each of its installed interfaces. The gateway server first tries to find the required addresses in the non-volatile memory. If the attempt succeeds, the gateway server continues with the next step in the startup sequence loading operating software or configuration information. If the attempt fails, the gateway server tries to get address information from the network.

To get address information from the network, the gateway server sends a broadcast request to all network hosts. If a host responds and supplies the required address information, the gateway server proceeds to load operating software or configuration information. If no host responds in a specified time, the gateway server displays an address request prompt on the console terminal and continues periodic network queries. The gateway server waits indefinitely for input from the console terminal or for a response from the network.

The gateway server can use different protocols to acquire address information from the network. The appropriate protocol depends on the interface type and whether a previous attempt failed.

The following sections describe address resolution for Ethernet and serial interfaces using Internet, DDN, and X.25 protocols. For address resolution under DECnet, see Chapter 16, "Using DECnet Protocol". For address resolution under Chaosnet and other protocols, see Chapter 17, "Using Other Protocols".

Determining Ethernet Interface Addresses

For an Ethernet interface, the gateway server uses Reverse Address Resolution Protocol (RARP) and, if RARP fails, Boot Protocol (BootP). Both protocols send a broadcast message to all available hosts. The message requests the 32-bit Internet address corresponding to the 48-bit hardware address of the Ethernet interface.

Many UNIX systems support RARP and BootP, which are defined in RFC 903 and RFC 951, respectively. The main difference between RARP and BootP is scope: RARP is limited to a single Ethernet cable; BootP can cross gateways to other networks. For more information, see Chapter 11, "Understanding Internet Addresses, Broadcasts, and Address Resolution".

Determining Serial Interface Addresses

The gateway server uses a simple address-determining convention for serial network interfaces using HDLC encapsulation, the default encapsulation method. Under this convention, the gateway server sends a message to the host on the other end of the serial link. The message asks the host to send back an Internet address for the serial interface in the gateway server. In response, the host returns its own address with one modification: if the host portion is 1, the host returns an address with a host portion of 2, and vice versa. (See Chapter 11 for information about Internet addresses.)

For example, if the host address is 192.6.34.1, the host would return 192.6.34.2 as the address of the serial interface in the gateway server. Note that the address-determining convention preserves network, subnet, and host identification in the first three octets of the address.

For serial interfaces using DDN 1822 and X.25 protocols (and for those using HDLC and an address not ending in 1 or 2), no message-based method exists for determining the interface address. To enable the gateway server to determine addresses for these interfaces, you must place address specification information in non-volatile memory or in a host-specific configuration file. For more information, see "Selecting and Setting Interface Addresses" in Chapter 7, "Configuring the Network Interfaces", and "Loading Configuration Information" later in this chapter.

Determining Subnet Masks

The gateway server can determine the subnet mask of an interface using the Internet Control Message Protocol (ICMP) Mask Request message, described in RFC 950. In this procedure, the gateway server sends ICMP Mask Request messages to the local network, one for each interface with an unknown subnet mask. Other gateways on the local network, including other **cisco Systems** gateway servers, return subnet mask information in an ICMP Mask Reply message.

Loading Operating Software Over the Network

When you first apply power, the gateway server loads and executes operating software from its ROM. To upgrade the operating software, you replace the ROM chips. However, if you have several **cisco Systems** gateway servers, there is an easier way to distribute upgraded operating software: you can load it over the network. This procedure is called *netbooting*.

You can specify netbooting through the processor configuration register or by placing a command in the non-volatile memory, if installed. The gateway server checks these sources for netbooting instructions when you turn on the power.

Understanding Netbooting

If the gateway server finds netbooting instructions, it determines at least one interface address and then loads an operating software file over the network using the Trivial File Transfer Protocol (TFTP). If the file load succeeds, the gateway server reboots to use the new operating software and then continues the startup process.

By default, the gateway server uses an Internet address of all ones to broadcast TFTP read requests. This broadcast address conforms to the current Internet standards. However, many hosts use an old style of broadcast address consisting of all zeros. You can change gateway server operation to accommodate hosts using the old style of broadcast address; see "Broadcast Addresses" in Chapter 11, "Understanding Internet Addresses, Broadcasts, and Address Resolution", for details.

Also by default, the gateway server continues sending TFTP read request messages until it receives a response. The gateway server remains unusable as long as the network or the host with the specified file is unavailable. To limit the number of netbooting attempts, set bit 13 in the processor configuration register to 1. The gateway server then gives up after five netbooting attempts and returns to the ROM operating software.

The gateway server can use any network interface to load operating software. However, for serial interfaces using DDN 1822 and X.25 protocols, the gateway server cannot automatically determine a host address. In these cases, you must use non-volatile memory to specify the address of the network host with the desired file.

The gateway server may load operating software and configuration files at the same time. This occurrence does not indicate a problem.

To display the Internet address of the network host that provided the current operating software, use the EXEC command **show hardware**.

Specifying Netbooting With the Processor Configuration Register

The processor configuration register includes four bits to specify booting instructions. The four bits are the lowest-order bits in the register. The bit pattern 0-0-0-0 prevents the gateway server from automatically booting, and the bit pattern 0-0-0-1 causes the gateway server to boot from ROM; any other bit pattern causes the gateway server to convert the bit pattern to an equivalent octal number in preparation for netbooting. For details on setting configuration register bits, see Chapter 8, "Configuring the Gateway Server Processors".

The gateway server uses the octal number from the configuration register to form a file name for use in TFTP read request messages. To form the file name, the gateway server starts with "cisco" and appends the octal number, a hyphen, and the processor type (in lowercase letters). The processor type is as shown by the EXEC command **show hardware**.

For example, if you set the lowest four bits of the configuration register to 1-0-0-0, they form the octal number "10". Then, if "CSC/2" is the processor type, the resulting file name is "cisco10-csc2".

Specifying Netbooting in the Non-Volatile Memory

To use the non-volatile memory option to specify netbooting, place a **boot system** command in the non-volatile memory. This command takes two arguments: *filename* and *address*. The *filename* argument is the file name of the operating software to load. 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 bits to any pattern other than 0-0-0-0 or 0-0-0-1.

Loading Configuration Information

After obtaining interface addresses and loading operating software (if so instructed), the gateway server loads configuration information. In this operation, the gateway server starts with the default values for all configuration parameters. It then tries to load configuration information from two sources: non-volatile memory and network hosts.

The configuration information loaded in these attempts overrides the default values and any previously loaded configuration information. This approach enables you to configure all gateway servers with general information and then refine the configuration to suit an individual gateway server.

Automatic Configuration Using Non-Volatile Memory

As part of its startup sequence, the gateway server always checks for configuration information in non-volatile memory. If the non-volatile memory is installed and holds valid configuration commands, the gateway server executes the commands.
If the gateway server finds no non-volatile memory or detects a problem with the non-volatile memory or the configuration information it contains, the gateway server proceeds to load configuration information from a network host. Problems can include a bad checksum for the information in the non-volatile memory and the absence of critical information in the non-volatile memory.

Even if the non-volatile memory provides configuration information, the gateway server attempts to load configuration information from a network host. You can instruct the gateway server to skip this attempt by placing the **no service config** command in non-volatile memory; see "Saving, Examining, and Moving Configuration Information" later in this chapter. This command limits the source of automatic configuration information to the non-volatile memory.

Automatic Configuration Using Network Hosts

After loading configuration information from non-volatile memory (if any), the gateway server attempts to load two configuration files from the network. The network configuration file contains commands that apply to all **cisco Systems** gateway servers and terminal servers on a network. The host configuration file contains commands that apply to one gateway server in particular. However, if the non-volatile memory included the **no service config** command, the gateway server does not load either configuration file.

The gateway server uses TFTP to load configuration files. By default, the gateway server uses an Internet address of all ones to broadcast TFTP read requests. This broadcast address conforms to the current Internet standards. However, many hosts use an old style of broadcast address consisting of all zeros. You can change gateway server operation to accommodate hosts using the old style of broadcast address; see "Broadcast Addresses" in Chapter 11, "Understanding Internet Addresses, Broadcasts, and Address Resolution", for details.

If the gateway server fails to load a configuration file during startup, it tries again every 10 minutes until a host provides the requested file. With each (failed) attempt, the gateway server displays a message on the console terminal. For example, if the gateway 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 **no service config** command on the console terminal.

To display the current configuration file names and the Internet addresses of the hosts that supplied the files, use the EXEC command **show hardware**.

Loading the Network Configuration File

The first file the gateway server tries to load is the network configuration file. This file contains commands that apply to all **cisco Systems** gateway servers on a network. Typically, the network configuration file contains commands to specify static routing entries, dynamic routing techniques, a default domain name, a list of name servers, and a set of host-name-to-address bindings.

By default, the gateway server uses the name "network-confg" for the network configuration file. You can specify a new network configuration file name by putting the **boot network** command into non-volatile memory; see "Saving, Examining, and Moving Configuration Information" later in this chapter. The **boot network** command can take two arguments: *filename* and, optionally, *address*. The argument *filename* is the new name for the network configuration file. If you omit the optional argument *address*, the gateway server uses the default broadcast address (as specified by the configuration register or the non-volatile memory). If you use the address, you can specify a specific network host or a subnet broadcast address.

Establishing the Host Name

After loading the network configuration file, the gateway server establishes its host name. The gateway server first looks for name-to-address information provided by a **host** command in the network configuration file. If this attempt fails, the gateway server broadcasts a Domain Name System (DNS) address-to-name query. If the gateway server receives no response, it uses "Gateway" as its host name.

By default, the gateway server uses its name to form a host configuration file name. To form this file name, the gateway server converts its host name to all lowercase, removes all domain information, and appends "-confg". For example, if the host name is "Chaff.cisco.com", the resulting host configuration file name is "chaff-confg".

You can specify a different host configuration file name by placing the **boot host** command in the network configuration file or in non-volatile memory. This command can take two arguments: *filename* and, optionally, *address*. The argument *filename* is the host configuration file name. The optional argument *address* identifies a specific network host or a subnet broadcast address. The gateway server uses the address you specify when requesting the host configuration file. If you omit the *address*, the gateway server uses the default broadcast address (usually all ones, 255.255.255.255).

Loading the Host Configuration File

Having determined a file name, the gateway server attempts to load the host configuration file. This configuration file contains commands specific to a gateway server, such as **interface** and **line** configuration commands.

Saving, Examining, and Moving Configuration Information

Gateway servers equipped with the non-volatile memory option can store configuration information in non-volatile memory (which has battery backup). Non-volatile memory can reduce the reliance of a gateway server on network hosts for configuration information. Reducing this reliance increases the dependability of the gateway server; the gateway server can start operations even when key network hosts are unavailable.

To copy current configuration information to non-volatile memory, use the EXEC command **write memory**. This command stores all non-default configuration information as configuration commands in text (ASCII) format. To protect against data corruption, **write memory** records a checksum for the information stored in non-volatile memory.

To display the configuration information stored in non-volatile memory, use the EXEC command **show configuration**.

To display current gateway server configuration information, use the EXEC command **write terminal**. (This command does not require non-volatile memory.) You can use this command and the **show configuration** command to find differences between the current configuration and that stored in non-volatile memory.

To clear the contents of non-volatile memory, use the EXEC command **write erase**. This command writes null (all zero) bytes to all non-volatile memory locations.

To send a copy of the current configuration information to a server host on the network using TFTP, use the EXEC command **write network**. This command prompts you for a host name or address and a destination file name. Usually, the destination file must already exist and have world-write permission. If the gateway server encounters an error during the transmission of configuration information, it displays an appropriate error message on the console terminal.

Using the Gateway Server as a Network Host

You can configure the gateway server to act as a (limited) TFTP-server host. As a TFTP-server host, the gateway 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 a file name that you specify.

This gateway server feature greatly simplifies upgrading one or more gateway servers to a new software release. Furthermore, if you also use non-volatile memory in all gateway servers, this capability eliminates reliance on network server hosts.

To specify TFTP-server operation for a gateway server, use the **tftp-server system** configuration command. This command takes two arguments: *filename* and *access-list*. The argument *filename* is the name you give the gateway server ROM "file", and the argument *access-list* is an access-list number. The gateway server sends the ROM software copy to the host issuing a TFTP read request with this file name. The access list specifies which hosts may receive the ROM software; to learn how to specify an access list, see "Defining Access Lists" in Chapter 10, "Adding Security Measures".

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

cisco Systems

Chapter 7
Configuring the Network Interfaces

This chapter describes how to configure interface software and hardware. The configuration parameters determine interface addresses, encapsulation type, and several other interface operations. This chapter also describes the hardware of the Ethernet, the synchronous serial network, and the DDN 1822 interfaces.

The gateway server displays a list of installed interfaces on the console terminal during startup. You can also display this information with the EXEC command **show hardware**. For more detailed interface information, use the EXEC command **show interface**.

Before you use the interface configuration commands described in this chapter at the console terminal, you must enter the privileged **configure** command. This command puts the gateway server in configuration command mode. You do not need to enter the **configure** command if you are placing the interface configuration commands in a configuration file. For information on how to specify configuration information from the console terminal, see Chapter 5, "Starting Gateway Server Operations"; for information on automatic configuration, see Chapter 6, "Advanced Gateway Server Configuration".

Starting Interface Configuration

In configuration command mode, you begin configuring a hardware interface with the **interface** command. This command identifies a specific interface for configuration and starts interface configuration command collection mode. The **interface** command takes two arguments: *type* and *unit*. The argument *type* can be **ethernet, serial,** or **ddn-1822**. The argument *unit* is an integer that specifies the *n*th interface of type *type*, starting with 0 for the first interface of that type.

For example, the following command starts interface configuration command collection for the first Ethernet interface in the gateway server:

```
interface ethernet 0
```

In interface configuration command collection mode, you can enter the interface subcommands described in the next sections of this chapter. Interface subcommands all apply to the interface specified in the **interface** command.

The interface configuration command collection mode ends when you enter a command that is not an interface subcommand. Comments do not affect the command collection.

Selecting and Setting Interface Addresses

This section describes how to set the address and subnet mask for an interface. For background information on Internet addresses, see Chapter 11, "Understanding Internet Addresses, Broadcasts, and Address Resolution".

To set the interface address and subnet mask, use the **address** interface subcommand. This subcommand takes two arguments: *address* and *subnet-mask*. The argument *address* is a Class A, Class B, or Class C Internet address for the interface. (The gateway server fully supports the subnet specifications in RFC 950.) The argument *subnet-mask* is a mask of address bits that specify the network portion of the address.

The following example shows the two commands necessary to set an address for the first Ethernet interface:

```
interface ethernet 0
address 192.31.7.17  255.255.255.240
```

Specifying the server interface addresses incorrectly often causes problems. The following rules can help you correctly specify interface addresses.

- Every interface must have a unique Internet address.
- Normally, connecting two interfaces to the same network or subnet is an error. However, for parallel serial links, two or more serial lines can connect to the same subnet. The gateway server displays a warning message if it finds two interfaces on the same network or subnet.
- Each serial link between a pair of gateways must constitute a single subnet or network. However, redundant serial links between a pair of gateways can belong to the same subnet or network. In all cases, each end of a serial link must have a unique address on that subnet or network.
- You cannot normally use the subnet represented by all zeros; the all-zero subnet is reserved. However, if you cannot avoid using the all-zero subnet, use the **service subnet-zero** configuration command to instruct the gateway server to accept this subnet.
- You cannot use the subnet represented by all ones; the all-ones subnet is also reserved. There are no exceptions to this restriction.
- The network and subnet portion of the interface address must match the network and subnet portion of all other host addresses on that network or subnet. That is, the subnet mask must be the same throughout a network.
- The gateway server will not accept interface addresses and subnet masks that are illegal in any way. If the gateway server rejects an interface address or subnet mask, check the value carefully.

- All hosts on your network must use the same kind of broadcast address: all ones or all zeros, and with or without a network number. See Chapter 11 for more information on broadcast addresses.

Selecting and Setting Broadcast Information

The gateway server provides Internet broadcast support on all media for which broadcast messages are defined. These media include Ethernet networks and point-to-point synchronous serial links. For a complete description of broadcasts and gateway server support of broadcasts, see Chapter 11, "Understanding Internet Addresses, Broadcasts, and Address Resolution".

Changing the Broadcast Address

If your gateway server has the non-volatile memory option, you can set the Internet broadcast address with the **broadcast-address** interface subcommand. This subcommand takes one argument, *address*, which specifies an Internet address for the gateway server to use in all broadcasts. The default broadcast address is all ones.

You can also specify the Internet broadcast address by setting jumpers in the processor configuration register. For a description of this method, see Chapter 8, "Configuring the Gateway Server Processors".

Forwarding UDP Broadcasts

The **cisco Systems** gateway server occasionally uses UDP broadcasts to determine address, configuration, and name information. ("UDP" stands for User Datagram Protocol, defined in RFC 768.) If the gateway server is connected to an Ethernet or serial network that does not include a server host, UDP broadcast requests fail.

To correct this situation, you can configure a gateway server interface to forward certain classes of UDP broadcasts to a specified address. To specify UDP broadcast forwarding, use the **helper-address** interface subcommand. This subcommand takes one argument, *address*, which can be either a directed broadcast address or a host address.

To avoid UDP forwarding problems, the gateway server examines the 48-bit hardware destination address in UDP broadcasts. If the hardware address is the all-ones broadcast address, the gateway server limits UDP forwarding to broadcasts using the TFTP, BootP, Domain, IEN-116 Name, and Time protocols. All other UDP broadcasts must use the hardware address of the gateway server. If the hardware destination address is not the all-ones broadcast address or the gateway server hardware address, the gateway server does not forward the UDP broadcast.

If the hardware address and protocol type meet the requirements, the gateway server delivers the broadcast to the specified helper address. This approach supports directed broadcasts.

The following summary lists the requirements for forwarding UDP broadcasts through a gateway server.

- Gateway server startup must be complete; the gateway server must know all interface addresses.
- The UDP broadcast must be a physical (MAC-level) broadcast directed to all hosts on the local Ethernet.
- The UDP broadcast must be directed to a port of one of the following types: TFTP server, BootP client, BootP server, Domain server, IEN-116 name server, or Time server.
- The gateway server interface receiving the UDP broadcast must have a specified helper address.
- A route to the network, subnet, or host specified in the helper address must exist.
- The route to the helper address must not use the interface that received the UDP broadcast.

To display information on UDP forwarding events, use the privileged EXEC command **debug packet-events**. For a complete description, see Chapter 18, "Network Monitoring and Troubleshooting".

Specifying an Encapsulation Method

Networks carry data packaged in various forms of encapsulation. An encapsulation method is a specified way of including certain information in each data packet. Each network medium has one or more encapsulation methods.

Encapsulation information can include hardware addresses for the source and destination hosts, the protocol type used for the enclosed data, and the length of the data in bytes. Network hosts use the encapsulation information to process the data packets.

The gateway server supports a variety of encapsulation methods. To specify an encapsulation method for a network interface, use the **encapsulation** interface subcommand. This subcommand has one argument, *method*, which specifies the encapsulation type. The valid encapsulation types depend on the type of interface.

The gateway server uses the encapsulation specified for an interface when the gateway server originates a data packet, and when it has no information to use a specific encapsulation method for a host. The gateway server uses ARP, Probe, SNAP ARP, and static ARP entries to determine which encapsulation method is appropriate for sending packets to a particular host.

Specifying Ethernet and IEEE 802.3 Encapsulation Methods

For Ethernet and IEEE 802.3 interfaces, the **encapsulation** interface subcommand may take one of three keywords: **arpa, iso1**, or **iso2.**

The **arpa** keyword establishes Ethernet 2.0 encapsulation as the default. This encapsulation consists of six bytes of hardware destination address, six bytes of hardware source address, and two bytes of protocol type. This method is the default encapsulation method for Ethernet interfaces.

The **iso1** keyword establishes encapsulation that complies with IEEE 802.2 and IEEE 802.3 as the default. This encapsulation consists of six bytes of hardware destination address, six bytes of hardware source address, two bytes of length, one byte of destination Service Access Point (SAP), one byte of source SAP, and one byte of control information.

The **iso2** keyword establishes encapsulation that is a combination of the **arpa** and **iso1** encapsulation methods as the default. The **iso2** encapsulation starts with the **iso1** encapsulation and includes five additional bytes. Of these five bytes, the last two indicate the protocol type as in **arpa** encapsulation. The **iso2** encapsulation method is also known as SNAP (Subnet Access Point) encapsulation.

These encapsulation methods determine only the encapsulation information included in data packets; they do not change electrical characteristics of an interface.

Specifying Serial Line Encapsulation Methods

For serial lines, the **encapsulation** interface subcommand takes the default keyword **hdlc.** The **hdlc** keyword specifies X.25 Level 2 bit-synchronous framing with four bytes of encapsulation. The encapsulation consists of one address byte, one control byte, and two bytes of protocol type code. The **hdlc** protocol type codes are the same as the Ethernet protocol type codes. This encapsulation method requires only framing and checksumming; it does not incur overhead from acknowledgment or retransmission strategies.

For DDN gateways with HDLC Distant Host (HDH) software, the encapsulation keyword is **hdh.** See Chapter 14, "Using DDN Protocols", for information on this encapsulation method.

For X.25 Level 2 and X.25 Level 3 protocols, the encapsulation keywords are **lapb, multi-lapb, x25, ddnx25, lapb-dce, multi-lapb-dce, x25-dce,** and **ddnx25-dce.** See Chapter 15, "Using X.25 Protocols", for information on these encapsulation methods.

Specifying the DDN 1822 Encapsulation Method

For the DDN 1822 interface, the default encapsulation keyword is **ddn-1822.** The DDN 1822 interface has only one encapsulation method.

Setting Other Characteristics

This section describes interface subcommands that apply to all interface types.

Setting the Maximum Transmission Unit

The maximum transmission unit (MTU) of an interface specifies the size of the largest Internet packet that an interface can send whole. To send a packet larger than the MTU, the gateway server must fragment the packet into two or more smaller packets.

To set the MTU of an interface, use the **mtu** interface subcommand. This subcommand takes one argument, *bytes*, which specifies the MTU size.

The interface hardware places an upper limit on MTU size. For Ethernet interfaces and serial interfaces, the maximum (and default) MTU is 1,500 bytes; for DDN 1822 interfaces, the maximum (and default) MTU is 1,006 bytes.

Every encapsulation method has its own maximum packet-size parameter. This parameter can be much smaller than the largest packet supported by the interface hardware. If the encapsulation method can fragment and reassemble packets (as the X.25 methods can), leave the MTU at its default value. This approach reduces the number of Internet fragments that the end host must process and speeds communication.

Note: In almost all cases, the gateway server software handles fragmentation issues automatically; it is unlikely that you will need to change the MTU parameter.

To display the current MTU for an interface, use the EXEC command **show interface**.

Shutting Down an Interface

If you need to disable an interface, use the **shutdown** interface subcommand. This subcommand prevents the transmission of packets on the specified interface. The subcommand also marks the interface as unavailable, which is communicated to other gateway servers through all dynamic routing protocols.

To restart a shutdown interface, use the **no shutdown** interface subcommand.

To check whether an interface is shut down, use the EXEC command **show interface**.

Looping Back an Interface

You can specify software and/or hardware loopback operation for serial line and DDN 1822 interfaces. During loopback operation, the interface receives back every packet it sends. Loopback operation has the additional effect of "disconnecting" gateway server functionality from the network. The **ping** command can be useful during loopback operation; see "Probing Nodes With the **ping** Command" in Chapter 18, "Network Monitoring and Troubleshooting," for information about this command.

You can use loopback operation to distinguish gateway server problems from line, modem, or CSU/DSU problems. If correct data transmission is not possible when an interface is in loopback mode, the interface is the source of the problem. The modem or CSU/DSU may have similar loopback functions you can use to isolate the problem, if the interface loopback test passes.

To start loopback operation, use the **loopback** interface subcommand.

To restore a loopback interface to normal operation, use the **no loopback** interface subcommand.

To show interfaces currently in loopback operation, use the EXEC command **show interface.**

Controlling Interface Hold Queues

Each network interface in a gateway 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.

After the interface hold queue reaches its limit, the gateway server sends an ICMP Source Quench message to the source host of any additional packet and discards that packet. The gateway server limits the rate at which it sends ICMP Source Quench messages to one per second per interface.

To change the hold queue limit of an interface, use the **hold-queue** interface subcommand. This subcommand takes one argument, *number*, which specifies the new hold queue limit. The hold queue limit can be as large as memory allows.

For X.25 interfaces, the default hold queue limit is 10 packets per Logical Channel Identifier. For LAPB interfaces, the default hold queue limit is 12 packets. For all other network interface types, the default hold queue limit is 100 packets. A hold queue limit of 100 packets prevents a malfunctioning interface from consuming an excessive amount of memory.

For slow links, use a small 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 hold queue limit. A fast link may be busy for a short time (and thus require the hold queue), but can empty the 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 interface.**

Controlling ICMP Host Redirect Messages

A host occasionally sends a packet to a gateway even though another gateway provides a better route. In this situation, the default gateway server response is to send an ICMP Host Redirect message to the host and forward the packet. (The gateway server does

not assume the host will change routing in response to the ICMP Host Redirect message.) The gateway server limits the rate at which it sends ICMP Host Redirect messages to one per second per interface.

To prevent the gateway server from sending ICMP Host Redirect messages on a particular interface, use the **no redirects** interface subcommand.

To re-enable the sending of ICMP Host Redirect messages, use the **redirects** interface subcommand.

To display the interfaces that have the ICMP Redirect function disabled, use the EXEC command **show interface**.

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. TCP 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. This subcommand takes one argument, *kilobits*, which specifies the intended bandwidth in kilobits/second. Default bandwidth values are set during startup and can be displayed with the EXEC command **show interface**.

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.

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. This subcommand takes one argument, *tens-of-microseconds*, which specifies the delay for an interface or network segment in tens-of-microseconds.

For an Ethernet segment, the default delay is 1,000 microseconds. For a serial line, the default delay is 20,000 microseconds.

The **delay** subcommand sets an informational parameter only; you cannot adjust the actual delay of an interface with this subcommand.

Resetting an Interface

In normal operation, you rarely need to reset an interface. However, if you need to re-initialize an interface during software debugging or for some other reason, use the EXEC command **clear interface**. This command takes two arguments, *type* and *unit*. The argument *type* can be **ethernet, serial,** or **ddn-1822**. The argument *unit* is an integer that specifies the *n*th interface of type *type*, starting with 0 for the first interface of that type. Resetting an interface re-initializes the interface hardware; it does not affect the configuration information for that interface.

Configuring for Bridging Operation

The bridging support provided by **cisco Systems** in its HyBridge and gateway server product lines is in accordance with draft versions of the IEEE Standard 802.1 (D) on MAC Bridges. **cisco Systems** is committed to comply with the 802.1 (D) standard as soon as it is adopted by the IEEE.

cisco Systems bridging support includes both learning and spanning-tree functionality. The HyBridge or gateway server acting as a bridge learns of host locations and addresses by "wire tapping" the Ethernet traffic. This bridging process interacts with other **cisco Systems** bridges to execute a spanning-tree algorithm that works to eliminate the use of multiple paths (which hinder the learning functionality of the bridge). Bridging does not limit the ability of the gateway server to route over redundant paths.

cisco Systems also supports remote bridging over synchronous serial lines that use the HDLC encapsulation method. Bridging over serial lines that use X.25 and LAPB encapsulation is not currently supported.

Bridging functions on a gateway server are currently restricted to the Ethernet and serial interfaces of a single Multi-port Communications Interface controller. Bridging is not possible using the Type 1 or Type 2 Ethernet interfaces or the CSC-S series or CSC-T serial interfaces.

Note that you can execute multiple, separate spanning-tree algorithms on separate Multi-port Communications Interfaces. However, the result is logically separate, unbridged networks.

When you configure a Multi-port Communications Interface to serve as a MAC-level bridge, the interface transfers traffic between networks that cannot otherwise be routed. Before bridging operation can begin, you must specify the parameters for the spanning-tree algorithm. These parameters must be the same for all **cisco Systems** HyBridges and gateway servers acting as bridges on the same network.

To specify bridging parameters, use the **bridge** configuration command. This command takes up to three arguments: *number*, a keyword, and a value related to the keyword. The argument *number* is a number between 1 and 10 that identifies the spanning tree to configure. The keyword and value arguments are optional. If you do not specify a keyword, the gateway server creates a spanning-tree data structure using the default parameters.

To specify the time a host address entry is allowed to exist without the gateway server hearing again from the host, use the keyword **max-age**. This keyword takes a value in seconds as its argument.

To specify how long the gateway server must wait to begin the learning (bridging) process after the spanning-tree algorithm has marked an interface as eligible to forward packets, use the keyword **forward-delay**. This keyword takes a value in seconds as its argument.

To specify the interval between Bridge Protocol Data Unit (BPDU) transmissions (which are used in executing the spanning-tree algorithms), use the keyword **hello-time**. This keyword takes a value in seconds as its argument.

To specify the priority of the bridge, use the keyword **priority**. This keyword takes a number from 0 through 65,000 as its argument. The gateway server uses the priority parameter to calculate the root node of the spanning tree.

Additional bridge keywords may be included in later releases of the gateway server software.

To shut down the bridging actions of all interfaces for a given spanning tree, use the **no bridge** configuration command. This command takes one argument, *number*, which specifies the number of the spanning tree to shut down.

You must identify each interface over which bridging will take place. To do so, use the **bridge-group** subcommand of the **interface** configuration command. This subcommand takes one argument, *number*, which specifies the number of a spanning tree. If *number* specifies a non-existent spanning tree, the gateway server ignores the subcommand.

To display the spanning-tree parameters and bridging tables, use the EXEC command **show bridge**. To log execution of the spanning-tree algorithm, use the privileged EXEC command **debug bridge**.

Multi-port Communications Interface Hardware

The **cisco Systems** Multi-port Communications Interface provides up to two Ethernet ports and up to two synchronous serial ports. The Ethernet ports support Ethernet Versions 1 and 2 and IEEE 802.3. The serial ports can be ordered from the factory with support for HDLC, LAPB, X.25, DDN X.25, and DDN 1822-J (HDH) transmission.

For packet transfers between two of its interfaces, the Multi-port Communications Interface provides a switching rate of 12,000 packets per second (independent of packet size). You can take advantage of this fast transfer rate by assigning closely related network segments to ports on the same Multi-port Communications Interface.

The serial ports on the Multi-port Communications Interface are externally clocked at data rates between 2.4 and 4,000 kilobits/second. The transmission rate is independent of packet size and activity on other channels. Each serial port can use a V.35, RS-232C, or RS-422 electrical interface.

Using Multi-port Communications Interfaces, the AGS can accommodate up to 10 serial ports or 8 T1 circuits, the MGS can accommodate up to 6 serial ports or 2 T1 circuits, the PGS can accommodate up to 4 serial ports or 2 T1 circuits, and the CGS can accommodate up to 2 serial ports or 1 T1 circuit.

Figure 7-1 shows the Multi-port Communications Interface as viewed from the component side with the bus connectors at the bottom. Note the 10 jumper areas, W1 through W10, and the one dip switch, S1.

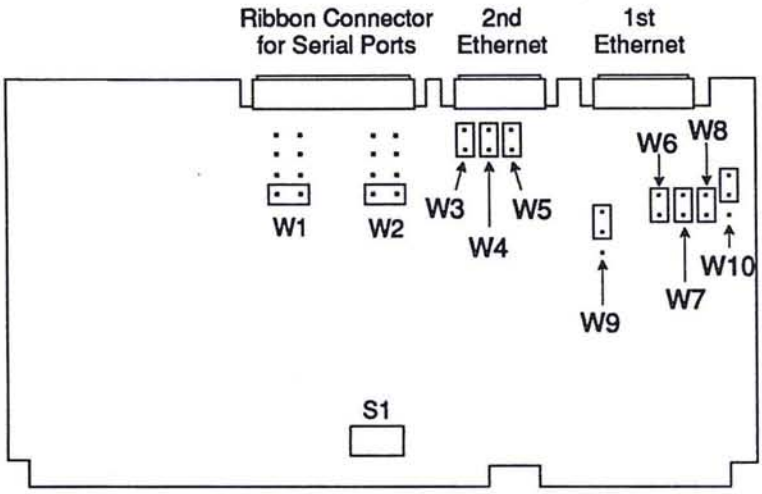


Figure 7-1. Component-side View of the Multi-port Communications Interface

Jumper area W1 consists of four two-pin jumper pairs. This jumper area controls the source of clock signals for the first serial interface. Placing a jumper on the topmost pair of pins specifies inverted, internal clocking; placing the jumper on the next pair down specifies non-inverted internal clocking; placing the jumper on the third pair specifies inverted external clocking; and placing the jumper on the bottom pair specifies non-inverted external clocking (the factory default).

Jumper area W2 functions in the same way as W1, but applies to the second serial interface.

To use internal clocking, you must also specify the clock rate with the **clockrate** subcommand of the **interface** configuration command. This command takes one argument, *rate*, which is the clock rate desired. The allowed internal clock rates are 1.2, 2.4, 4.8, 9.6, 19.2, 38.4, 56, and 64 kilobits/second.

Choose inverted or non-inverted clock signals in accordance with the modem or CSU/DSU employed.

Jumper pairs W3 through W8 select grounding options as shown in Table 7-1. Inserting a jumper grounds a signal; removing a jumper allows the signal to float. The Multi-port Communications Interface provides these options to accommodate the differences between the Ethernet and IEEE 802.3 electrical specifications; Ethernet permits certain signals to float, whereas IEEE 802.3 requires the signals be grounded. Table 7-1 shows the jumpers, the corresponding signals, and the affected interfaces. The factory default is to ground all signal pairs, which is compatible with both Ethernet and IEEE 802.3 requirements.

Table 7-1. Jumper Settings for Grounding Options

Jumper Pair	Signal Description	Interface
W3	Receive Pair Shield	First Ethernet
W4	Receive Pair Shield	Second Ethernet
W5	Transmit Pair Shield	First Ethernet
W6	Transmit Pair Shield	Second Ethernet
W7	Power Pair Shield	First Ethernet
W8	Power Pair Shield	Second Ethernet

Jumpers W9 and W10 are three-pin jumpers that select between Ethernet and IEEE 802.3 electrical formats. Viewing the jumpers as shown in Figure 7-1, placing a jumper on the upper pair of pins selects IEEE 802.3, and placing a jumper on the lower pair of pins selects Ethernet. Jumper W9 controls the first Ethernet and jumper W10 controls the second Ethernet. The factory default is to select IEEE 802.3 compliance. This setting also works with nearly all Ethernet connections.

Switch S1 determines the unit number of the Multi-port Communications Interface. This switch is a four-position type that selects unit numbers as shown in Table 7-2.

Table 7-2. S1 Settings for Multi-port Communications Interface Unit Numbering

Unit	4	5	6	7
0	off	off	off	off
1	off	off	off	on
2	off	off	on	off
3	off	off	on	on
4	off	on	off	off
5	off	on	off	on

Other IEEE 802.3 and Ethernet Interface Hardware

This section describes the Type 1 and Type 2 IEEE 802.3 and Ethernet network interface hardware. These interfaces can use IEEE 802.3 MAUs (Media Access Units) and Ethernet Version 1 and 2 transceivers. The Type 1 interface is blue and has a light blue non-locking connector socket. This interface is obsolete and described in this manual for customers who already have this interface. The Type 2 interface is green and has a locking connector socket. This interface is used mostly in single-Ethernet configurations.

The nine-slot AGS supports a maximum of five Type 1 interfaces, and a maximum of seven Type 2 interfaces.

The Type 1 Interface

Figure 7-2 shows the Type 1 Ethernet interface as viewed from the component side with the bus connectors at the bottom. Note the three jumper areas J1, J2, and J3, and the two dip switches SW1 and SW2.

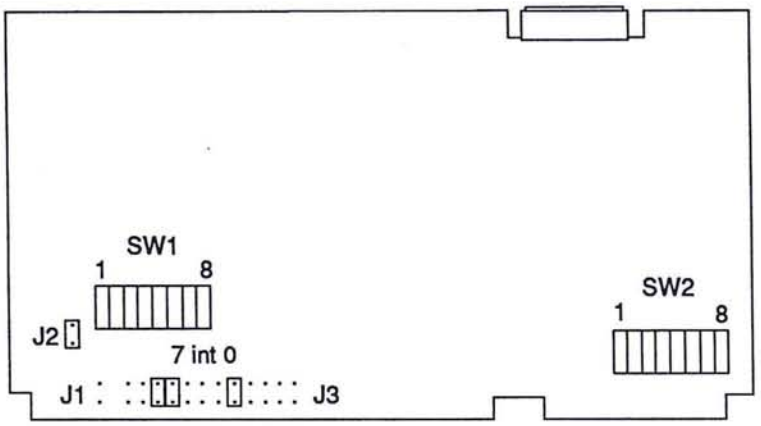


Figure 7-2. Component-side View of the Type 1 Ethernet Interface

J1 and J2 are single-bit jumper areas. J1 must have no jumper installed; J2 should have a jumper installed. Jumper area J3 consists of 12 jumper positions, three of which have jumpers installed, as Figure 7-2 shows. These jumpers control address space utilization and interrupt level assignment; do not change their settings.

To set the unit number of a Type 1 interface, use SW1. Set switches SW1-4 through SW1-7 to the intended unit number as shown in Table 7-3. Leave switches SW1-1 through SW1-3 and SW1-8 turned off.

Table 7-3. SW1 Settings for Type 1 Interface Unit Numbering

Unit	4	5	6	7
0	off	off	off	on
1	off	off	on	off
2	off	off	on	on
3	off	on	off	off
4	off	on	off	on

The gateway server ignores address switches SW2-1 through SW2-8; leave these switches turned off.

Gateway servers equipped with Type 1 interfaces require Ethernet Version 1 transceivers, or Ethernet Version 2 transceivers with the heartbeat disabled. To use IEEE 802.3 Media Access Units (MAUs), you must connect pin 4 of the Access Unit Interconnect (AUI) cable to ground (pin 1). The principal difference between Ethernet transceivers and IEEE 802.3 MAUs is that Ethernet allows pin 4 to float, whereas IEEE 802.3 requires that pin 4 be grounded.

The Type 2 Interface

Figure 7-3 shows the Type 2 Ethernet interface as viewed from the component side with the bus connectors at the bottom. Note the seven jumper areas, each of which consists of one to eight three-post groups. In these jumper areas, a jumper can connect the middle post to either of the outer posts.

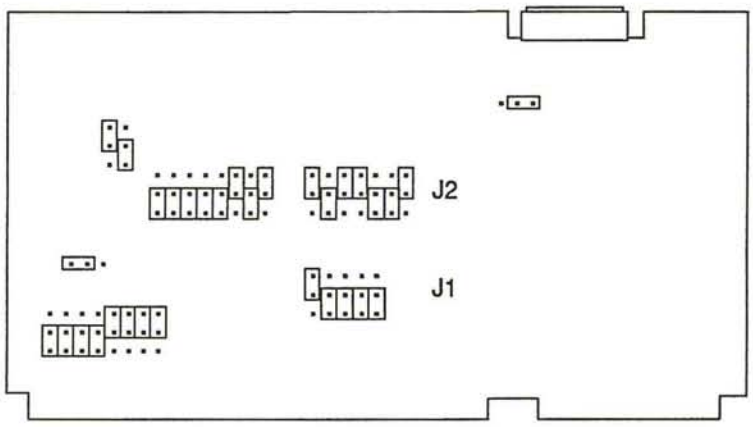


Figure 7-3. Component-side View of the Type 2 Ethernet Interface

The jumper areas labeled J1 and J2 determine the unit number and Multibus address of a Type 2 interface. To set the interface unit number and Multibus address, set jumper areas J1 and J2 as illustrated in Figure 7-4.

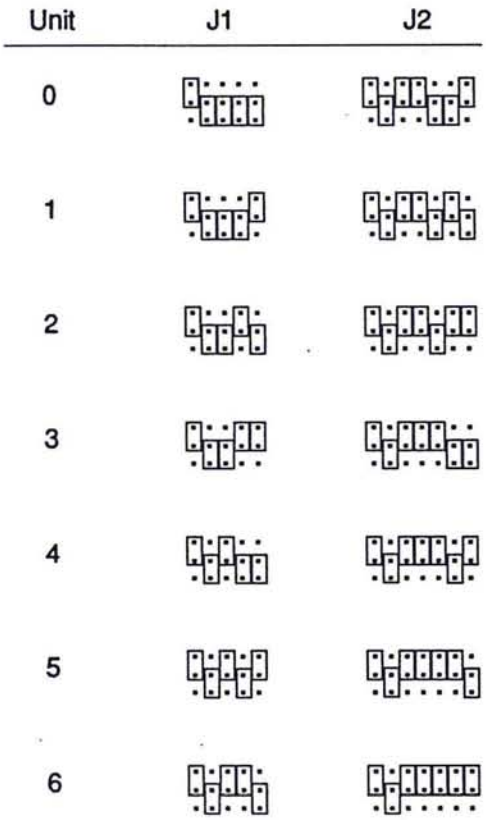


Figure 7-4. Type 2 Interface Unit Jumpering

By factory-set default, the Type 2 interface connects to an IEEE 802.3 MAU. To specify that the Type 2 interface connects to an Ethernet transceiver, move the jumper of jumper area W1 to the left side.

The remaining jumper areas on the Type 2 interface should not be changed.

Serial Network Interface Hardware

In addition to the Multi-port Communications Interface, the gateway server supports two basic synchronous serial network interfaces, the CSC-S and the CSC-T. The Multi-port Communications Interface supersedes these interfaces. The CSC-S serial interface operates at common data rates between 1.2 kilobits/second and 64 kilobits/second. The CSC-T interface operates from 1.2 kilobits/second to over 3,100 kilobits/second, which includes T1 and T1C data rates.

The CSC-S interface has two derivative variations: the CSC-D and the CSC-X. The CSC-D interface supports DDN X.25 protocol and DDN 1822-J (HDH) protocol, and the CSC-X interface supports X.25 protocols. All interfaces support raw HDLC framing and LAPB data encapsulation.

This section describes hardware-specific details of the CSC-S and CSC-T serial interface. The hardware description for the CSC-S interface applies to the CSC-D and CSC-X interfaces as well.

The CSC-S Serial Interface

The AGS and MGS can accommodate up to three CSC-S serial interface cards. The CGS can accommodate one CSC-S card. Each CSC-S card provides two serial interfaces, yielding a maximum of six slower-speed serial interfaces. Figure 7-5 shows the CSC-S interface as viewed from the component side with the bus connectors at the bottom.

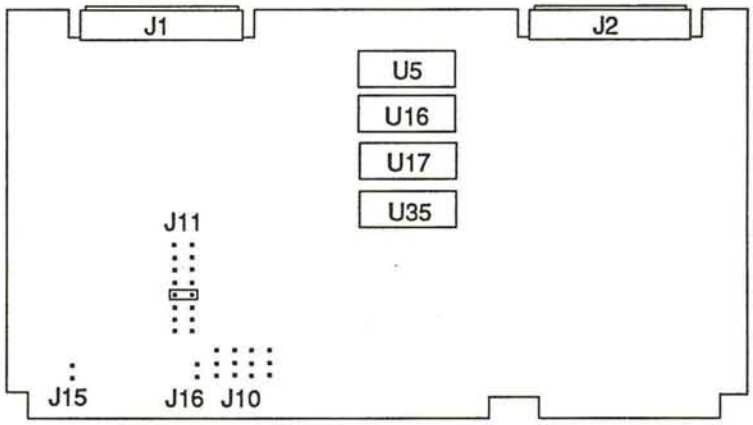


Figure 7-5. Component-side View of the CSC-S Interface

The CSC-S interface uses the RS-232C electrical interface standard. A 50-pin micro-ribbon cable carries the RS-232C signals to male (DTE) D-type connectors at the rear of the chassis. The 50-pin micro-ribbon cable attaches to the connector labeled J1 in Figure 7-5 (which appears on the right side when viewing the installed card from the front of the gateway server). V.35 and RS-422 adapter kits are available from cisco Systems.

The CSC-S interface displays information using nine LEDs. Looking at the interface card edge-on with the component side up, you can identify the LEDs by counting from left to right. LEDs 1 through 4 indicate information for the first port (port A), LEDs 5 through 8 indicate information for the second port (port B), and LED 9 indicates when the interface is running.

In each of the groups of four (from left to right), the first LED indicates when the data carrier detect signal is high, the second LED indicates interface data transmission, the third LED indicates interface data reception, and the fourth LED is unused. Table 7-4 shows the use of LEDs on the CSC-S interface card.

Table 7-4. CSC-S LED Indicators

1	2	3	4	5	6	7	8	9
Interface run	-	Port B rx data	Port B tx data	Port B carrier	-	Port A rx data	Port A tx data	Port A carrier

The CSC-S interface performs a series of self-checks when you turn on the power. If one of these self-checks fails, the interface stops normal operation and blinks its LEDs on and off in unison. The blinking continues until you can resolve the problem.

To set the unit number of a CSC-S interface, use the J11 jumper area shown in Figure 7-5. Figure 7-6 shows how to set jumpers for each unit number. The remaining jumpers on the CSC-S interface are not user-configurable and should be left at their factory settings.

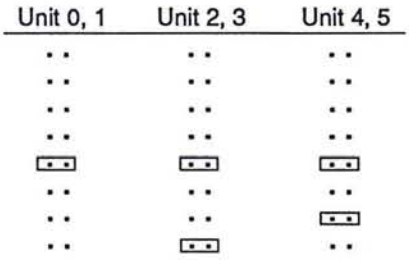


Figure 7-6. CSC-S Unit Number Jumpering

The modem or CSU/DSU provides clocking for the synchronous serial communication. However, the multi-protocol UART used in the CSC-S interface sends an interrupt to the interface processor at every CTS transition. This action causes severe performance degradation. The UART manufacturer provides no software means to disable this behavior. *To avoid performance degradation, configure the modem or CSU/DSU to hold RTS and CTS high at all times.*

The CSC-S interface card includes a connector header (labeled J2 in Figure 7-5) used to enable or disable port B. Viewed from the component side of the card with the bus connectors at the bottom, connector J2 is on the right. To determine pin numbers in the connector header, count from the inside of the card to the outside of the card starting with zero. To disable port B, place a horizontal jumper across the bottom pins at positions 11 and 12 (see Figure 7-7). To enable port B, place a vertical jumper across the upper and lower pins at position 12 (see Figure 7-8).

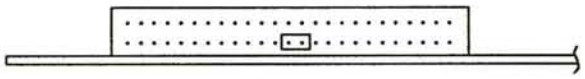


Figure 7-7. Edge-on View of the CSC-S Serial Interface Header With Port B Disabled

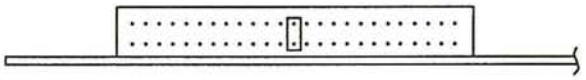


Figure 7-8. Edge-on View of the CSC-S Serial Interface Header With Both Ports Enabled

If one or both of the ports on a CSC-S interface card run at 20 kilobits/second or less, you can improve throughput with the **interleave** interface subcommand. To disable this throughput enhancement, use the **no interleave** interface subcommand.

The CSC-T Serial Interface

The AGS can accommodate up to two CSC-T serial interface cards. Each CSC-T card provides two serial interfaces, yielding a maximum of four higher-speed serial interfaces. Figure 7-9 shows the CSC-T interface as viewed from the component side with the bus connectors at the bottom.

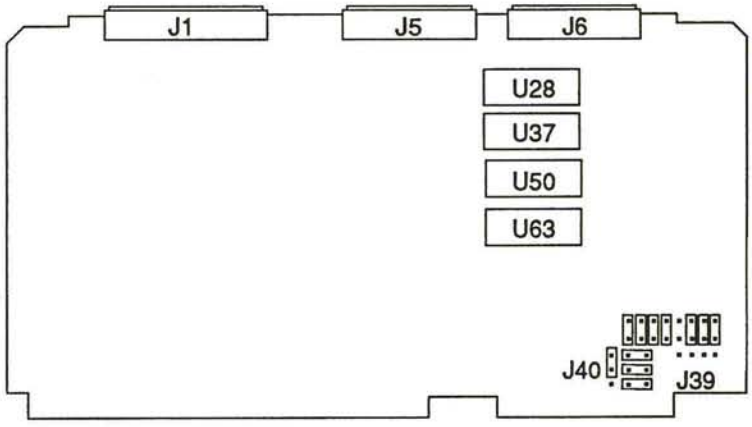


Figure 7-9. Component-side View of the CSC-T Interface

The CSC-T interface sends TTL-level data signals from its center connector area, labeled J= in Figure 7-9, over a 40-pin micro-ribbon cable to a V.35 converter. The V.35 converter is mounted internally at the rear of the gateway server. From the converter, a standard V.35 cable carries the data signals to the CSU/DSU.
The CSC-T interface displays information using nine LEDs. Viewing the installed interface card from the front of the gateway server, you can identify the LEDs from left to right as follows. The first LED (which is green) illuminates when the CSC-T card is running. The second, third, and fourth LEDs illuminate when the -12 volt, +12 volt, and +5 volt power supplies for the V.35 converter unit are working correctly. The fifth LED illuminates when the +5 volt power supply for the CSC-T card is working correctly.

The sixth LED illuminates when the interface transmits or receives data for port B. The seventh LED illuminates when the carrier signal for port B is high. The eight and ninth LEDs perform the same functions for port A. Table 7-5 summarizes these LED indicators.

Table 7-5. CSC-T LED Indicators

1	2	3	4	5	6	7	8	9
Interface run	-12V V.35 F3	+12V V.35 F2	+5V V.35 F1	+5V CSC-T	Port B data	Port B carrier	Port A data	Port A carrier

To set the unit number of a CSC-T interface, use the J39 jumper area shown in Figure 7-9. Figure 7-10 shows how to set jumpers for each unit number.

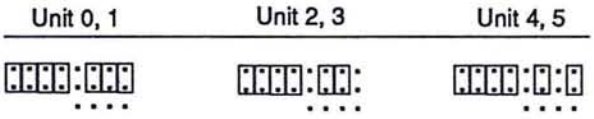


Figure 7-10. CSC-T Unit Number Jumpering

The CSC-T interface produces data in a synchronous HDLC framing format; you must use a T1 converter to produce T1-format signals. Note that T1 converters that can invert the resulting bitstream satisfy the ones density requirement of the T1 standard and use the T1 bandwidth efficiently. T1 converters that cannot invert the bitstream require more of the T1 bandwidth to meet the ones density requirement.

The CSC-T interface uses external clocking. Clocking signals pass through the V.35 converter and can range from 1.2 kilobits/second to 3.1 megabits/second. Any modem that meets the V.35 interface specification can provide clocking for the CSC-T interface.

The CSC-T card has fuses to protect the circuitry from connector accidents. Fuse F1 is a 1-amp fuse and fuses F2 and F3 are 1/2-amp fuses. As Table 7-5 suggests, if fuse F1, F2, or F3 blows, the corresponding power supply LED goes out to indicate the problem.

The CSC-T firmware resides in EPROM sockets U37 and U63 on the interface card. To disable port B, you must insert a "disable" EPROM, available from cisco Systems, in either socket U28 or socket U50. Removing the "disable" EPROM re-enables port B.

DDN 1822-LH/DH (CSC-A) Interface Hardware

The AGS can accommodate up to four CSC-A interface cards. The CSC-A interface is not available on the MGS, the PGS, or the CGS. Each CSC-A card provides one DDN 1822 interface. Figure 7-11 shows the CSC-A interface as viewed from the component side with the bus connectors at the bottom.

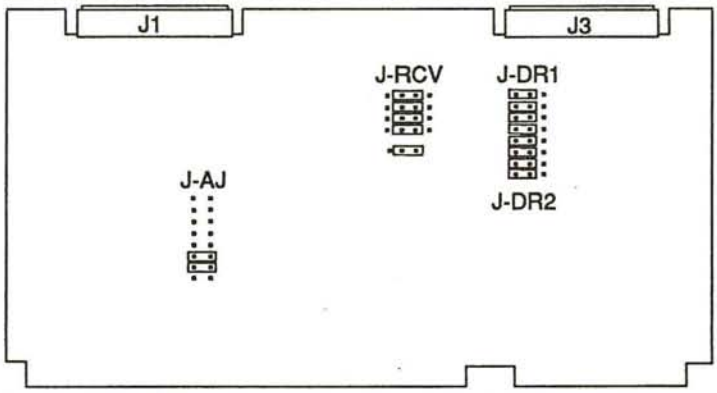


Figure 7-11. Component-side View of the CSC-A Interface

The CSC-A interface produces signals in the format specified in BBN Report 1822. These signals are indicated by LEDs mounted on an insert in the gateway server front panel, as illustrated in Figure 7-12. The LEDs are briefly described in Table 7-6.

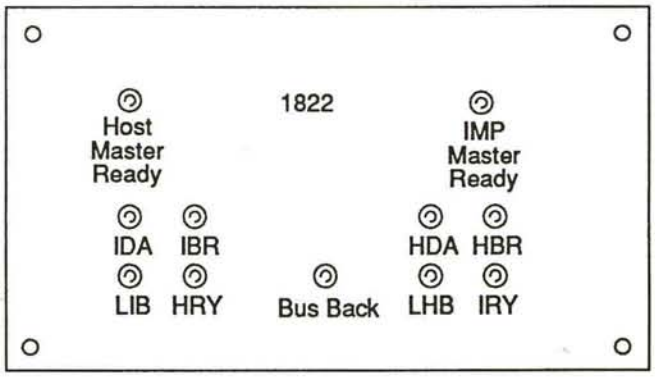


Figure 7-12. 1822 Display Panel

Table 7-6. CSC-A Display Panel Indicators

Indicator	Meaning When Indicator On
Host Master Ready	Host has initialized the 1822 interface.
IMP Master Ready	IMP (Internet Message Processor) is active.
IDA (IMP Data)	IMP asserts the "DATA-IMP-TO-HOST" signal.
IBR (IMP Bit Ready)	IMP asserts the "THERE'S-YOUR-IMP-BIT" signal.
LIB (Last IMP Bit)	IMP asserts the "LAST-IMP-BIT" signal.
HRY (Host Ready)	Host asserts the "HOST-READY-FOR-NEXT-BIT" signal.
HDA (Host Data)	Host asserts the "DATA-HOST-TO-IMP" signal.
HBR (Host Bit Ready)	Host asserts the "THERE'S-YOUR-HOST-BIT" signal.
LHB (Last Host Bit)	Host asserts the "LAST-HOST-BIT" signal.
IRY (IMP Ready)	IMP asserts the "IMP-READY-FOR-NEXT-BIT" signal.
BUS BACK	CSC-A is in local loopback mode.

The placement of CSC-A interfaces in the chassis backplane is critical for correct operation. The first CSC-A interface must be installed in the backplane slot directly below the processor slot. The second CSC-A interface must be installed in the very next slot, and so on. A CSC-A interface in the wrong slot claims to be "initializing" when you enter the EXEC command **show interface**.

To set the unit number of a CSC-A interface, use the J-AJ jumper area shown in Figure 7-11. Figure 7-13 shows how to set jumpers for each unit number.

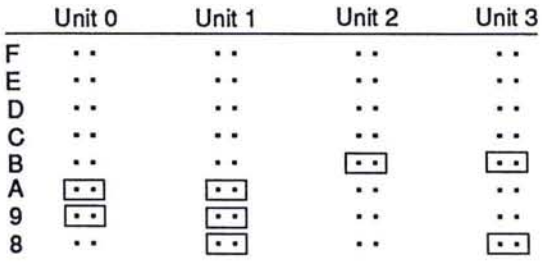


Figure 7-13. CSC-A Unit Number Jumpering

A common problem during the installation of a gateway server with a CSC-A interface is a mismatch of the electrical interface between the CSC-A and the DDN Packet Switched Network (PSN) port. The CSC-A may be configured for the Distant Host (DH) standard and the PSN port for the Local Host (LH) standard, or vice versa.

You can configure the CSC-A interface to use either the Distant Host or the Local Host standard. Jumper areas J-RCV, J-DR1, and J-DR2 determine which standard the CSC-A uses. (See Figure 7-11 for the locations of these jumper areas.) Jumper area J-RCV consists of four-post groups, and jumper areas J-DR1 and J-DR2 consist of three-post groups.

By factory-set default, the CSC-A interface uses the Distant Host standard. Figure 7-14 shows the jumper settings that specify Distant Host operation.

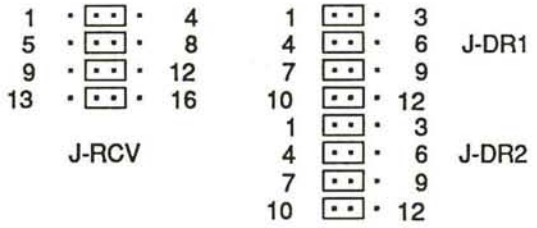


Figure 7-14. Jumper Settings for Distant Host Operation

To specify Local Host operation, use the jumper settings shown in Figure 7-15.

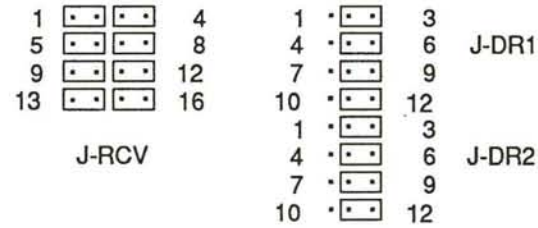
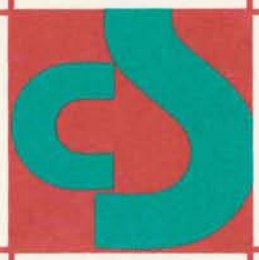


Figure 7-15. Jumper Settings for Local Host Operation

cisco Systems



Chapter 8
Configuring the Gateway Server Processors

cisco Systems offers two processors for use in its gateway servers. The CSC/1 processor is an MC68000 processor running at 10 MHz with 1 megabyte of RAM and up to 256 kilobytes of ROM. The CSC/2 processor is an MC68020 processor running at 12 MHz with 1 megabyte of RAM and up to 2 megabytes of ROM. The CSC/2 has more than twice the processing capacity of the CSC/1.

The CSC/1 Processor

Figure 8-1 shows the component side of the CSC/1 processor as viewed with the bus connectors at the top.

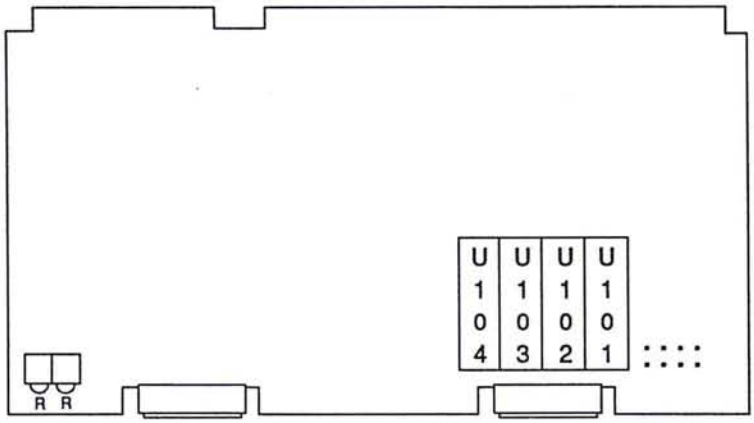


Figure 8-1. Component-side View of the CSC/1 Processor

The CSC/1 processor has two red LEDs at its bottom left corner, as Figure 8-1 shows. The LED on the left lights during initialization; if this LED remains lit more than 15 seconds, the processor self-test has failed. The LED on the right is the processor halt light, which lights when the processor is in a halt state.

To the right of the LEDs is a 50-pin header. The rightmost 16 pairs of pins of this header make up the configuration register, described later in this chapter. The rightmost pair of pins is the least significant bit of the configuration register.

The 50-pin header to the right of the configuration register is the console cable attachment. A cable connects this header to an RS-232C connector on the back panel (of the AGS, MGS, and CGS) or the top panel (of the PGS).

The four sockets near the console cable attachment hold EPROMs. **cisco Systems** uses EPROMs to distribute gateway server software updates. From right to left, the sockets are labeled U101 through U104. Sockets U102 and U104 can accept only 512 kilobit (type 27512) EPROMs. Sockets U101 and U103 can accept 64 kilobit (type 2764) to 512 kilobit (type 27512) EPROMs, depending on the settings of the jumper area next to U101. Figure 8-2 shows the jumper settings for the various EPROM types.

EPROM Type	Jumper Settings
2764	
27128	
27256	
27512*	

* Factory default.

Figure 8-2. CSC/1 EPROM Jumper Settings

The CSC/2 Processor

Figure 8-3 shows the component side of the CSC/2 processor as viewed with the bus connectors at the top.

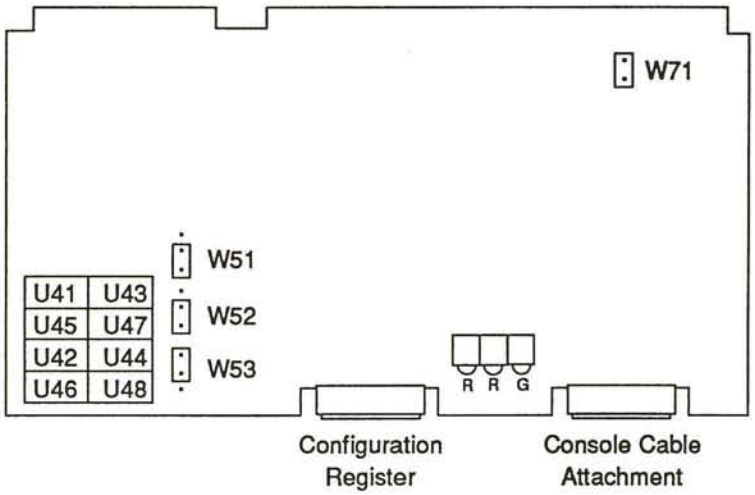


Figure 8-3. Component-side View of the CSC/2 Processor

The CSC/2 processor has eight EPROM sockets near its bottom left corner, as Figure 8-3 shows. These sockets are labeled U41 through U48. Currently, the CSC/2 uses only four of the eight sockets (specifically, U42, U44, U46, and U48).

The CSC/2 EPROM sockets can accept 64 kilobit (type 2764) to 8 megabit EPROMs, depending on the settings of jumper areas W51, W52, and W53. Each of these jumper areas consists of three pins. Figure 8-4 shows the jumper settings for the various EPROM types. To configure the CSC/2 processor for EPROMs with more than 1 megabit of storage, you must return the processor to cisco Systems.

Jumper Areas	EPROM Types				
	2764	27128	27256	27512*	27010
W51					
W52					
W53					

* Factory default.

Figure 8-4. CSC/2 EPROM Jumper Settings

To the right of the jumpers is a 50-pin connector header (see Figure 8-3). The rightmost 16 pairs of pins of this header make up the configuration register, described later in this chapter. The rightmost pair of pins is the least significant bit of the configuration register.

To the right of the configuration register, the CSC/2 processor has two red LEDs and a green LED. The red LED on the left is a software-programmable status light; it lights during initialization, flashes to indicate an error, and remains off under normal operation. The middle LED is the processor halt light; it lights when the processor is in a halt state. The green LED is another software-programmable status light; it lights when the system is running.

The 50-pin header to the right of the LEDs is the console cable attachment. A cable connects this header to an RS-232C connector on the back panel (of the AGS, MGS, and CGS) or the top panel (of the PGS).

Jumper area W71 is located on the right side and near the back of the CSC/2 processor (see Figure 8-3). This jumper is a simple two-pin type. Installing a jumper in W71 (the factory default) causes the CSC/2 to act as a Multibus master, regardless of its position in the Multibus backplane. Do not remove this jumper unless instructed to do so by an installation note accompanying a gateway server upgrade.

The Processor Configuration Register

Both the CSC/1 and CSC/2 processors have a 16-bit hardware configuration register located on the rightmost 16 pairs of jumper pins on the left 50-pin connector header. Bit 0 is the rightmost pair of pins. To set a bit (to 1), insert a vertical jumper. To clear a bit (to 0), remove the vertical jumper.

To change configuration register settings, turn off the gateway server, set or clear the bits, and restart the gateway server. Configuration register changes take effect when the gateway server restarts.

Figure 8-5 shows a schematic of the configuration register with the factory settings. These settings apply to both the CSC/1 and CSC/2 processors.

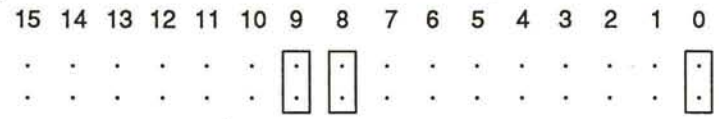


Figure 8-5. The Configuration Register With Factory Settings

The lowest four bits of the processor configuration register (bits 3, 2, 1, and 0) form the "boot field". The boot field specifies a number in binary. If you set the boot field value to 0, you must boot the operating system manually with a **b** command to the system bootstrap program. If you set the boot field value to 1 (the factory default), the gateway server boots using the default ROM software. If you set the boot field to any other bit pattern, the gateway server uses the resulting number to form a boot file name for netbooting.

The gateway server creates a boot file name as part of the automatic configuration processes described in Chapter 6, "Advanced Gateway Server Configuration". To form the boot file name, the gateway server starts with "cisco" and concatenates the octal equivalent of the boot field number, a dash, and the processor type name. Table 8-1 lists the default boot file names or actions for the CSC/1 processor. The list is the same for the CSC/2 processor, with "csc2" substituted for "csc1".

Table 8-1. Default Boot File Names for the CSC/1 Processor

Action/File Name	Bit 3	Bit 2	Bit 1	Bit 0
bootstrap mode	0	0	0	0
ROM software	0	0	0	1
cisco2-csc1	0	0	1	0
cisco3-csc1	0	0	1	1
cisco4-csc1	0	1	0	0
cisco5-csc1	0	1	0	1
cisco6-csc1	0	1	1	0
cisco7-csc1	0	1	1	1
cisco10-csc1	1	0	0	0
cisco11-csc1	1	0	0	1
cisco12-csc1	1	0	1	0
cisco13-csc1	1	0	1	1
cisco14-csc1	1	1	0	0
cisco15-csc1	1	1	0	1
cisco16-csc1	1	1	1	0
cisco17-csc1	1	1	1	1

The four bits after the boot field (bits 4, 5, 6, and 7) in the configuration register are unused and must be left cleared (0).

Bit 8 in the configuration register controls the console BREAK key. Setting bit 8 (the factory default) causes the processor to ignore the console BREAK key. Clearing bit 8 causes the processor to interpret BREAK as a command to force the system into the bootstrap monitor, effectively halting normal operation.

Bit 9 in the configuration register controls the automatic restart actions of the gateway server. Setting bit 9 (the factory default) causes the system to restart automatically when it encounters a software error condition. Clearing bit 9 causes the gateway server to cease normal operation and return to the bootstrap command level when it encounters a software error condition.

Bit 10 in the configuration register controls the host portion of the Internet broadcast address. Setting bit 10 causes the processor to use all zeros; clearing bit 10 (the factory default) causes the processor to use all ones. Bit 10 interacts with bit 14, which controls the network and subnet portions of the broadcast address (see the description later in this section). Table 8-2 shows the combined effect of bits 10 and 14.

Table 8-2. Configuration Register Settings for Broadcast Address Destination

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

Bits 11 and 12 in the configuration register determine the baud rate of the console terminal. Table 8-3 shows the bit settings for the four available baud rates. The factory default is 9,600 baud.

Table 8-3. Console Terminal Baud Rate Settings

Baud Rate	Bit 12	Bit 11
9600	0	0
4800	0	1
1200	1	0
300	1	1

Bit 13 in the configuration register determines the gateway server response to a boot load failure. Setting bit 13 causes the gateway server to load operating software from ROM after five unsuccessful attempts to load a boot file from the network (see Chapter 6, "Advanced Gateway Server Configuration"). Clearing bit 13 causes the gateway server to continue attempting to load a boot file from the network indefinitely. By factory default, bit 13 is cleared to 0.

Bit 14 in the configuration register controls the network and subnet portions of the Internet broadcast address. Setting bit 14 causes the gateway server to include the network and subnet portions of its address in the broadcast address. Clearing bit 14 causes the gateway server to set the entire broadcast address to all ones or all zeros, depending on the setting of bit 10. By factory default, bit 14 is cleared to 0. See Table 8-2 for the combined effect of bits 10 and 14.

Bit 15 in the configuration register controls factory diagnostic mode in the gateway server. Setting bit 15 causes the gateway server to produce detailed CPU self-check messages, automatically prompt for interface addresses (not look for addresses on the network), not read configuration files or non-volatile memory, and automatically set diagnostic tracing modes using the **debug** commands. Clearing bit 15 (the factory default) causes the gateway server to operate normally.

cisco Systems

Chapter 9
Configuring the Console and Virtual Terminal Lines

The gateway server has one console line and five virtual terminal ports. For compatibility with the **cisco Systems** terminal server, the gateway server includes the ability to configure the console and virtual terminals. This chapter describes how to configure terminal lines and related gateway server characteristics.

Before you use the line configuration commands described in this chapter at the console terminal, you must enter the **configure** command. This command puts the gateway server in configuration command mode. You do not need to enter the **configure** command if you are placing the line configuration commands in a configuration file. For information on how to specify configuration information from the console terminal, see Chapter 5, "Starting Gateway Server Operations"; for information on automatic configuration, see Chapter 6, "Advanced Gateway Server Configuration".

Specifying a Terminal Line

To begin configuring a terminal line, you use the **line** command. This command identifies a single line or a range of lines for configuration, and enters line configuration command collection mode. To identify a line, the **line** command can take two arguments: *first-line* and, optionally, *last-line*. The argument *first-line* is the octal number of the terminal line (or the first line in a contiguous group) you want to configure. The optional argument *last-line* is the octal number of the last line in a contiguous group you want to configure.

In line configuration command collection mode, you can enter the line subcommands. Line subcommands all apply to the terminal line(s) specified in the **line** command. The line configuration command collection mode ends when you enter a command that is not a line subcommand.

The **line** command enables you to configure a large group of lines at the same time. After you set the defaults for the group, you can use additional line subcommands to set special characteristics for individual terminal lines.

Note: Gateway server line numbers are always octal.

Setting Line Parameters

The configuration parameters you set with the line subcommands determine line security, timeouts, and other operations. Some of the subcommands described in this section use the decimal representation of an ASCII character as an argument. See Appendix C, "ASCII Character Set", for ASCII-to-decimal conversion information.

Providing Line Security

To restrict connections on a line to certain Internet addresses, use the **access-class** line subcommand. This subcommand takes two arguments: *list* and the keyword **in** or **out**. The argument *list* identifies an access list of Internet addresses. The keyword **in** applies to incoming connections, such as virtual terminals; the keyword **out** applies to outgoing Telnet connections.

You can also set a password for the line. The **password** line subcommand takes one argument, *password*, which specifies the password to enter. For all lines, setting a password controls access to the EXEC command interpreter; the **login** command prompts for the password. Because an incoming connection may attach to any of the five virtual terminal lines, it is a good idea to set the same password for all these lines. For the console line (line 0), setting a password also controls access to the privileged command mode; the **enable** command prompts for the password on all non-console lines.

See Chapter 10, "Adding Security Measures", for a description of the security features available on **cisco Systems** gateway servers.

Defining Timeouts

You can define the amount of time the gateway server maintains a connection or an EXEC process without user input.

To set the interval that the gateway server waits for user input before closing the connection to a remote computer and returning the terminal to an idle state, use the **session-timeout** line subcommand. This subcommand takes one argument, *minutes*, which specifies the interval in minutes. The default interval value is 0, indicating that the gateway server maintains the connection indefinitely.

To set the interval that the EXEC waits for user input before resuming the current connection or, if no connections exist, returning the terminal to the idle state, use the **exec-timeout** line subcommand. This subcommand takes one argument, *minutes*, which specifies the interval in minutes. The default interval is 10 minutes; an interval of 0 specifies no timeouts.

Describing the Terminal

You can enter information about the terminal for use in network monitoring and certain Telnet negotiations. To describe the location of the terminal, its status, or both, use the **location** line subcommand. This subcommand takes one argument, *text*, which is the desired description. The description appears in the output of the **systat** command.

To record the type of the terminal connected to the line, use the **terminal-type** line subcommand. This subcommand takes one argument, *text*, which the Telnet terminal-type negotiation uses to inform the remote host of the terminal type. This command is most useful for the console line, but may be used for virtual terminals as well.

Padding Characters

To set padding for a specified output character, use the **pad** line subcommand. This subcommand takes two arguments: *decimal-number* and *count*. The argument *decimal-number* is the decimal representation of the ASCII character. The argument *count* is the number of NUL bytes sent after that character is output. For example, to pad carriage returns (decimal 13) with 25 NUL bytes, use the subcommand:

```
pad 13 25
```

Virtual terminal handlers discard padding characters; the **pad** subcommand is useful only for the console line.

Defining an Escape Character

To define a new escape character, use the **escape-character** line subcommand. This subcommand takes one argument, *decimal-number*, which is the decimal representation of the escape character or the character itself. The default escape character is CTRL-^. The BREAK key may not be used as an escape character on the gateway server console; it is useful on virtual terminals only if the host physically serving a terminal supports Telnet Break Service.

Using Other Line Configuration Commands

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

To set the number of lines on the terminal screen, use the **length** line subcommand. This subcommand takes one argument, *screen-length*, which is the number of lines on the screen. The gateway server uses this value to determine what constitutes a screenful of output and thus when to pause. The default length is 24 lines. A value of 0 (zero) disables pausing between screens of output.

You can define a message to be printed on the screen of a vacant terminal before the EXEC returns the terminal to the idle state. To define this message, use the **vacant-message** line subcommand. Follow **vacant-message** with one or more blank spaces and a delimiting character you choose. Then type one or more lines of text, terminating the message with the second occurrence of the delimiting character.

Using Related Configuration Commands

This section describes configuration commands that set gateway server characteristics related to console and terminal lines.

Specifying a Host Name

The host name for the gateway server appears in the name used in prompts and default configuration file names. The default host name is "Gateway". To change the host name, use the **hostname** configuration command. The **hostname** command takes one argument, *name*, which is the new host name. The **hostname** command preserves the case of letters in the argument *name*.

Although not required, it is a good idea to set the host name and the gateway server name supplied to other hosts to the same value. You set the latter name with the **host** configuration command described in "Loading Configuration Information" in Chapter 6, "Advanced Gateway Server Configuration".

Setting a Banner Message

You can set a "message of the day" banner, which the EXEC command interpreter displays whenever a user starts an EXEC process. To set this message, use the **banner** configuration command. Follow **banner** with one or more blank spaces and a delimiting character you choose. Then type one or more lines of text, terminating the message with the second occurrence of the delimiting character. For example:

```
banner #
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.

cisco Systems

Chapter 10
Adding Security Measures

The gateway server supports system security with two forms of access controls: passwords and access lists. Passwords control access on a per-line basis.

Access lists, which are based on Internet addresses and their masks, control the connections that may be made to and from hosts. If you need information on Internet addresses before creating access lists, see Chapter 11, "Understanding Internet Addresses, Broadcasts, and Address Resolution".

You may assign different access controls to different terminal lines and network interfaces. Note that each network host should have additional security procedures.

Controlling Line Access Using Passwords

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

To set a password on a terminal line, first identify the line with the **line** configuration command. Use the **password** line subcommand to specify a password, and the **login** line subcommand to enable password checking. The following example sets the password "letmein" on terminal line 10:

```
line 10
password letmein
login
```

If your gateway server has the non-volatile 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 situation, force the gateway server into factory diagnostic mode by turning off the gateway server, inserting a jumper in bit 15 of the processor configuration register, and turning on the gateway server. See "The Processor Configuration Register" in Chapter 8, "Configuring the Gateway Server Processors", for more information.

When the gateway server restarts in factory diagnostic mode, it does not read the non-volatile memory, thus avoiding the command to set a password for the console terminal. To resume normal operation, set a new terminal line password, turn off the gateway server, remove the jumper from bit 15 of the configuration register, and turn on the gateway server again.

Defining Access Lists

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

To specify an access condition, use the **access-list** configuration command. This command takes four arguments: *list*, the keyword **permit** or **deny**, *address*, and *mask*.

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

The gateway server compares the address being tested to the address you specify in the argument *address*, ignoring any bits specified in *mask*. If you use the condition keyword **permit**, a match causes the address to be accepted. If you use the condition keyword **deny**, a match causes the address to be rejected.

An access list can contain an indefinite number of actual addresses and up to 20 wildcard addresses. If you plan to assign the access list to an interface, include the addresses of all the gateway server interfaces in the list. A *wildcard address* has a non-zero address mask and thus potentially matches more than one actual address. The gateway server examines first the actual addresses, then the wildcard addresses. The order of the wildcard addresses is important, because the gateway server stops examining access-list entries after it finds a match.

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

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

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

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

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

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

To display the contents of all access lists, use the EXEC command **show access-lists**.

Controlling Line Access Using Access Lists

To restrict incoming and outgoing connections between a particular terminal line and the addresses in an access list, use the **access-class** subcommand of the **line** configuration command. This command takes two arguments: *list* and the keyword **in** or **out**. The argument *list* is an integer from 1 through 99 that specifies an access list. Use the keyword **in** to control which hosts may make Telnet connections into the gateway server virtual terminals. The following example defines an access list that permits only hosts on network 192.89.55.0 to connect to the virtual terminals on the gateway server.

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

Use the keyword **out** to define the access checks made on outgoing connections. (A user who types a host name at the gateway server prompt to initiate a Telnet connection is making an outgoing connection.) The following example defines an access list that denies connections to networks other than network 36.0.0.0 on terminal lines 1 through 5.

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

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

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

Controlling Interface Access Using Access Lists

To control access to an interface, use the **access-group** subcommand of the **interface** configuration command. This subcommand takes one argument, *list*, which is an integer from 1 through 199 that specifies an access list.

After receiving and routing a packet to a controlled interface, the gateway server checks the source address of the packet against the access list. If the access list permits the address, the gateway server transmits the packet. If the access list rejects the address, the gateway server discards the packet and returns an ICMP Destination Unreachable message.

Controlling Interface Access Using Extended Access Lists

You can filter interface traffic based on source address, destination address, and protocol information using extended access lists. You can use extended access lists only with the **access-group** interface subcommand, which takes an extended-access-list number as an argument.

To define an extended access list, use the extended version of the **access-list** command. This command takes seven required arguments: *list*, the keyword **permit** or **deny**, *protocol*, *source*, *source-mask*, *destination*, and *destination-mask*. The command may also take two optional arguments, *operator* and *operand*.

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

The argument *protocol* can be **ip**, **tcp**, **udp**, **icmp**, or **chaos**. Use the keyword **ip** to match any Internet protocol, including TCP, UDP, and ICMP. Note that the **cisco Systems** implementation of the Chaosnet protocol assumes that Chaosnet addresses are mapped to Internet address schemes: the 8-bit Chaosnet subnet corresponds to the low eight bits of the Internet subnet, and the 8-bit Chaosnet host number corresponds to the low eight bits of an Internet address.

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

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

For **tcp** and **udp** protocol keywords, *operator* can be **lt** (less than), **gt** (greater than), **eq** (equal), or **neq** (not equal), and *operand* is the decimal destination port for the specified protocol. For the **chaos** protocol keyword, *operator* can be **eq** (equal) or **neq** (not equal), and *operand* is the contact string of a Request For Connection packet.

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

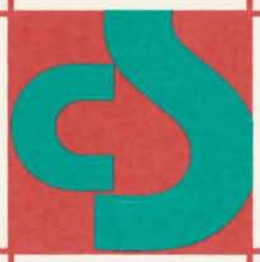
The "trick" in this example is that the initial request for an SMTP connection is made on TCP destination port 25. After this initial request, all further packets are addressed to a TCP destination port greater than 1023; the SMTP server determines the actual port used. Also remember that the access list used is that of the interface on which the packet would ordinarily be transmitted.

This example is illustrated by the following listing, in which the Ethernet network is a Class B network 128.88.0.0 and the mail host is 128.88.1.2.

```
! Permit outgoing TCP connections to any destination
access-list 101 permit tcp  128.88.0.0 0.0.255.255 0.0.0.0 255.255.255.255
! Permit any incoming TCP connections with destination port .gt. 1023
access-list 101 permit tcp  0.0.0.0 255.255.255.255 128.88.0.0 0.0.255.255  gt 1023
! Permit incoming TCP connections to the SMTP port of host 128.88.1.2
access-list 101 permit tcp  0.0.0.0 255.255.255.255 128.88.1.2 0.0.0.0  eq 25
! Deny outgoing ICMP messages
access-list 101 deny icmp  128.88.0.0 0.0.255.255 0.0.0.0 255.255.255.255
! Permit incoming ICMP messages for error feedback
access-list 101 permit icmp  0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
! End of list, disallow everything else
```

When using extended lists, performance degradation is linearly related to the number of terms in the access list. In the example above, the performance loss is minimal, because the gateway server spends most of its time waiting for the DDN link to send packets. Performance loss would be more noticeable in an Ethernet-to-Ethernet gateway.

cisco Systems



Chapter 11 Understanding Internet Addresses, Broadcasts, and Address Resolution

This chapter presents the details of Internet addresses and address resolution.

Understanding Address Classes and Formats

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

Internet Address Classes

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

The Class A Internet address format allocates the highest eight bits to the network field and sets the highest-order bit to 0 (zero). The remaining 24 bits form the host field. Only 128 Class A networks can exist, but each Class A network can have almost 17 million hosts. Figure 11-1 illustrates the Class A address format.

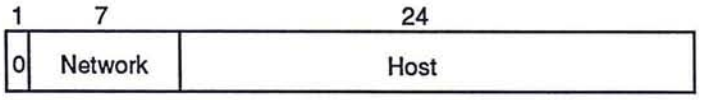


Figure 11-1. The Class A Internet Address Format

The Class B Internet address format allocates the highest 16 bits to the network field and sets the two highest-order bits to 1,0. The remaining 16 bits form the host field. Over 16,000 Class B networks can exist, and each Class B network can have up to 65,000 hosts. Figure 11-2 illustrates the Class B address format.

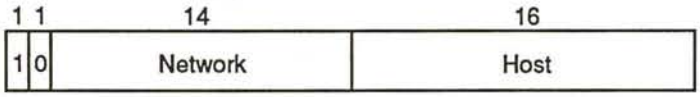


Figure 11-2. The Class B Internet Address Format

The Class C Internet address format allocates the highest 24 bits to the network field and sets the three highest-order bits to 1,1,0. The remaining eight bits form the host field. Over 2 million Class C networks can exist, and each Class C network can have up to 255 hosts. Figure 11-3 illustrates the Class C address format.

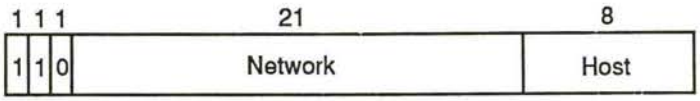


Figure 11-3. The Class C Internet Address Format

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

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

Internet Address Notation

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

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

```
192.31.7.19
10.7.0.11
255.255.255.255
0.0.0.0
```

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

Allowable Internet Addresses

Some Internet addresses are reserved for special uses and cannot be used for host, subnet, or network addresses. Table 11-1 lists ranges of Internet addresses and shows which addresses are reserved and which are available for use.

Table 11-1. Reserved and Available Internet Addresses

Class	Address or Range	Status
A	0.0.0.0	Reserved
	1.0.0.0 through 126.0.0.0	Available
	127.0.0.0	Reserved
B	128.0.0.0	Reserved
	128.1.0.0 through 191.254.0.0	Available
	191.255.0.0	Reserved
C	192.0.0.0	Reserved
	192.0.1.0 through 223.255.254	Available
	223.255.255.0	Reserved
D, E	224.0.0.0 through 255.255.255.254	Reserved
	255.255.255.255	Broadcast

Internet Address Conventions

If the bits in the host portion of an address are all 0, that address refers to the network specified in the network portion of the address. For example, the Class C address 192.31.7.0 refers to a particular network.

Conversely, if the bits in the network portion of an address are all 0, that address refers to the host specified in the host portion of the address. For example, the Class C address 0.0.0.234 refers to a particular host.

If the bits in the host portion of an address are all 1, that address refers to all hosts on the network specified in the network portion of the address. For example, the Class B address 128.1.255.255 refers to all hosts on the 128.1.0.0 network.

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

Addresses and Routing

Addresses are fundamental to the routing and delivery of data packets. Consider a host sending an Internet data packet. The host sends the packet directly to the destination host if that host is on the same network. If the destination host is on another network, the host sends the packet to a gateway server.

To determine whether the destination host is on the same network, the sending host compares the network portions of the destination address and its own address. If these network numbers are the same, the destination host is on the same network. If the network numbers are different, the destination host is on another network.

A gateway server has two or more network interfaces onto different networks. The primary function of the gateway server is to route packets between these networks, delivering them to their final destination or to another gateway. (A gateway-to-gateway transmission is called a *hop*.)

To begin the routing process, the gateway server examines the network number of the destination address. Using this number as a key, the gateway server locates applicable routing information in its routing table. The gateway server uses this routing information to send the packet to its final or an intermediate destination.

You can manually configure the **cisco Systems** gateway server routing table, or you can specify that a high-level routing protocol dynamically build the routing table. In both cases, the routing table is based on the network portion of addresses. Consequently, the addresses of hosts on a single physical network must have the same network number to permit automatic routing. If a network does not meet this requirement, gateway servers will be unable to communicate with all of the hosts on that network.

Subnetting

Subnetting is a scheme for imposing a simple hierarchy on hosts on a single physical network. The usual practice is to use a few of the leftmost bits in the host portion of the network addresses for a subnet field. For example, Figure 11-4 shows a Class B address with five bits of the host portion used as the subnet field. The official description of subnetting is RFC 950, "Internet Standard Subnetting Procedure".

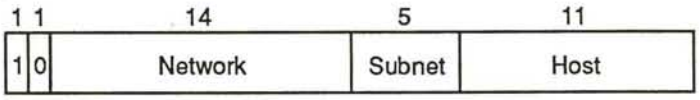


Figure 11-4. A Class B Address With a Five-Bit Subnet Field

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

Subnetting and Routing

Gateway servers and hosts can use the subnet field for routing. The rules for routing on subnets are identical to the rules for routing on networks. However, correct routing requires that all subnets of a network be physically contiguous. In other words, the network must be set up such that it does not require traffic between any two subnets to cross another network. Also, RFC 950 implicitly requires that all subnets of a network have the same number of bits in the subnet field.

Subnet Masks

A *subnet mask* identifies the subnet field of network addresses. This mask is a 32-bit Internet address written in dotted-decimal notation with all ones in the network and subnet portions of the address. For the example shown in Figure 11-4, the subnet mask is 255.255.248.0.

Table 11-2 shows the subnet masks you can use to divide an octet into subnet and host fields. The subnet field can consist of any number of the host field bits; you do not need to use multiples of eight. However, you should use three or more bits for the subnet field--a subnet field of two bits yields only four subnets, two of which are reserved (the 1,1 and 0,0 values).

Table 11-2. Subnet Masks

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

Hosts can determine subnet masks using the ICMP Mask Request message. Gateway servers respond to this request with an ICMP Mask Reply message. See Chapter 13, "Using Other Internet Services", for descriptions of ICMP messages.

Broadcast Addresses

A broadcast is a data packet destined for all hosts on a particular network. Network hosts recognize broadcasts by special addresses. This section describes the meaning and use of Internet broadcast addresses. For detailed discussions of broadcast issues in general, see RFC 919, "Broadcasting Internet Datagrams", and RFC 922, "Broadcasting Internet Datagrams in the Presence of Subnets". The gateway server support for Internet broadcasts complies with RFC 919 and RFC 922.

The current standard for an Internet broadcast address requires that the host portion of the address consist of all ones. If the network portion of the broadcast address is also all ones, the broadcast applies to the local network only. If the network portion of the broadcast address is not all ones, the broadcast applies to the network or subnet specified. (Broadcast addresses that include the network or subnet fields are called *directed broadcasts*.) In the latter case, the sender need not be on the same network or subnet as the intended receiving hosts.

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

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

Gateways provide some protection from broadcast storms by limiting their extent to the local cable. Level 2 bridges, by the nature of their design, forward broadcasts to all network segments, thus propagating all broadcast storms.

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

By default, the gateway server uses all ones for both the network and host portions of the Internet broadcast address (255.255.255.255). You can change the Internet broadcast address by setting jumpers in the process configuration register or by using the **broadcast-address** subcommand of the **interface** configuration command. The latter method requires the non-volatile memory option. See Chapter 8, "Configuring the Gateway Server Processors", and Chapter 7, "Configuring the Network Interfaces", respectively, for details.

Address Resolution

Internet addressing is independent of the media used. Therefore, Internet addresses are not the same as hardware (medium-dependent) addresses.

To communicate with a host that uses the IEEE 802.3 protocol, the gateway server must first determine the 48-bit Ethernet or IEEE 802.3 hardware address of that host. The process of determining the hardware address from an Internet address is called *address resolution*.

During the startup process, the gateway server must determine the Internet address corresponding to the hardware address for each interface. The process of determining the Internet address from a hardware address is called *reverse address resolution*.

The gateway server uses three forms of address resolution: Address Resolution Protocol (ARP), proxy ARP, and Probe (which is similar to ARP). The gateway server uses one form of reverse address resolution: Reverse Address Resolution Protocol (RARP). The ARP, proxy ARP, and RARP protocols, which are used on Ethernets, are defined in RFCs 826, 1027, and 903, respectively. Probe is a protocol developed by the Hewlett-Packard Company and used on IEEE 802.3 networks.

Address Resolution Using ARP

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

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

You can instruct the gateway server to use IEEE 802.2-compatible (SNAP format) ARP requests with the **service iso2** configuration command. The gateway server uses SNAP-format requests in conjunction with standard ARP requests. To end the use of SNAP-format requests, use the **no service iso2** command.

Address Resolution Using Proxy ARP

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

Address Resolution Using Probe

By default, the gateway server uses Probe (in addition to ARP) whenever it attempts to resolve an IEEE 802.3 or Ethernet hardware address. The subset of Probe that performs address resolution is called *Virtual Address Request and Reply*; the gateway server implements only this subset. Using Probe, the gateway server can communicate transparently with Hewlett-Packard IEEE 802.3 hosts that use IEEE 802.2 data encapsulation.

To disable the use of Probe, use the **no service probe** configuration command. To restore the default operation, use the **service probe** command.

Reverse Address Resolution Using RARP

The gateway server uses RARP during startup processing to attempt to determine the Internet addresses of its network interfaces from their hardware addresses. RARP works in the same way as ARP, except that the RARP Request packet requests an Internet address instead of a hardware address. RARP requires a RARP server on the network segment served by a gateway interface.

Working With the ARP Cache

Whenever the gateway server determines the hardware address corresponding to an Internet address (or vice versa), it stores the address pair in the ARP cache to speed later address resolution processing. The process of using the address resolution protocols to determine and cache address information is called *dynamic address resolution*. The gateway server automatically removes an ARP cache address pair (called an *entry*) if it has not received an ARP Reply packet from that host within four hours.

Most hosts support dynamic address resolution. For those that do not, you can make permanent ARP cache entries using the **arp** configuration command. This command takes three arguments: *internet-address*, *ethernet-address*, and *type*. The argument *internet-address* is the Internet address in dotted-decimal format corresponding to the hardware address specified by the argument *ethernet-address*, specified as a "dotted triple" of hexadecimal digits. The argument *type* is one of the encapsulation method keywords **arpa**, **iso1**, or **iso2**. For example:

```
arp 192.31.7.19 0800.0C00.1F5C arpa
```

To remove an entry from the ARP cache, use the **no arp** configuration command. This command takes one argument, *internet-address*, which is the Internet address of the entry you want to remove.

To clear all ARP cache information except the interface entries, use the privileged EXEC command **clear arp**.

To display the contents of the ARP cache, use the EXEC command **show arp**.

cisco Systems

Chapter 12
Routing With the Gateway Server

Routing is the process of determining where to send data packets destined for addresses outside the local network. Gateway servers gather and maintain routing information to enable the transmission and receipt of such data packets. Conceptually, routing information takes the form of entries in a *routing table*, with one entry for each identified route. The gateway server can create and maintain the routing table dynamically to accommodate network changes whenever they occur.

This chapter describes gateway server support for routing in Internet Protocol (IP) networks. See Chapter 16, "Using DECnet Protocol", and Chapter 17, "Using Other Protocols", for descriptions of routing in other networks.

Gateway Server Routing Capabilities

The **cisco Systems** gateway server provides great flexibility in determining and using routing information. This flexibility results from the four dynamic routing protocols that the gateway server supports, and from its many protocol-independent routing features.

Supported Routing Protocols

The gateway server supports four protocols for dynamically determining routing information: Routing Information Protocol (RIP), Exterior Gateway Protocol (EGP), HELLO, and Interior Gateway Routing Protocol (IGRP).

RIP is the routing protocol used by the *routed* process on Berkeley-derived UNIX systems. Most networks use RIP; it works well for small, isolated, and topologically simple networks.

EGP is the protocol for exchanging routing information with the DDN (Defense Data Network) core gateway system and similar large groups of networks.

HELLO is a routing protocol used in older networks. Developed for the Fuzzball gateways of the Distributed Computer Network project, HELLO is used in the National Science Foundation (NSF) regional networks.

IGRP, developed by **cisco Systems**, is a versatile routing protocol especially valuable in large networks with indefinitely complex topology and comprised of segments having different bandwidth and delay characteristics.

Routing Features

The gateway server offers many protocol-independent routing features. Subnetting, supported by RIP, HELLO, and IGRP, enables you to divide a network into logical subparts. Load-balancing enables you to split network traffic over parallel paths in a round-robin fashion, which provides greater overall throughput and extra reliability through redundancy.

Because the gateway server can avoid routing loops, you can implement general network topologies. Its network notification of disabled interfaces eliminates network "black holes". Static routing table entries can provide routing information when dynamically obtained entries are not available. Finally, the capability to use multiple concurrent routing protocol provides extra flexibility.

A Note About Using Multiple Routing Protocols

Although the gateway server can support multiple routing protocols simultaneously, the use of multiple routing protocols is usually undesirable. The routing protocols available today were not designed to interoperate with one another. Each protocol collects different types of information and reacts to topology changes in its own way. Because of this variation, it is very difficult to use multiple routing protocols and still have stable and consistent routing.

cisco Systems strongly recommends that only one routing protocol be used as the interior routing protocol for an autonomous system. However, when communicating with another network administered by a different political entity, you may need to use a second routing protocol. In this case, you can start a second routing process on the gateway server and selectively pass routing information between the interior and exterior routing protocols.

You can minimize multiple routing protocol problems by reducing the routing information obtained from another routing protocol to availability information and ignoring other information such as hop counts or delays. To obtain this effect, use the **default-metric** router subcommand when redistributing information between two dynamic routing protocols (see "Redistributing Routing Information" later in this chapter).

Setting Up Routing

To begin configuring an IP routing process, use the **router** configuration command followed by router subcommands. The **router** command takes one required argument, *protocol*, and one optional argument, *autonomous-system*. The argument *protocol* is a protocol-type keyword: **rip**, **egp**, **hello**, or **igrp**. Only IGRP uses the argument *autonomous-system*, which is the number of an autonomous system.

To shut down a routing process, use **no router** followed by a protocol-type keyword.

Defining an Autonomous System

An *autonomous system* is an administratively defined collection of networks that exchange routing information with each other. An autonomous system may comprise one or many networks, and each network may or may not have an internal structure (subnetting). The *autonomous system number*, assigned by the DDN Network Information Center, is a 16-bit decimal number that uniquely identifies the autonomous system. All gateway servers that belong to an autonomous system must be configured with the same autonomous system number.

Sending Routing Information About a Network

To send routing information about a directly connected network to other routers, use the **network** router subcommand. This subcommand takes one required argument, *address*, which is a network address, and an optional keyword, **passive.**

Without the keyword **passive**, the gateway server sends all routing updates generated by the routing protocol specified in the **router** command to the network specified in the **network** subcommand (and to all other connected networks). If the specified routing protocol uses broadcasts, the gateway server sends the broadcast to that network. With the keyword **passive**, the gateway server does not send routing updates to the specified network.

For example, the following commands specify that the gateways on network 128.88.0.0 do not receive routing updates. However, the gateways on network 192.31.7.0 do receive routing updates, including routes to 128.88.0.0.

```
router rip
network 128.88.0.0 passive
network 192.31.7.0
```

To stop sending routing information about a network, use the **no network** router subcommand followed by the network address.

It is important to define routing protocol use in terms of networks and subnets rather than in terms of interfaces, because more than one interface may connect to a single network.

The following example configuration defines RIP as the routing protocol to be used on networks 128.88.0.0 and 192.31.7.0.

```
router rip
network 128.88.0.0
network 192.31.7.0
```

In this example, a third network attached to the gateway server would not receive RIP routing updates and would remain unknown to networks 128.88.0.0 and 192.31.7.0 for routing purposes.

Directly Connected Routes

Directly connected routes are routes to the networks specified by the interface addresses of the gateway server. The gateway server automatically enters a directly connected route in the routing table if the interface is usable. A "usable" interface is one through which the gateway server can send and receive packets. If the gateway server determines that an interface is not usable, it removes the directly connected routing entry from the routing table. Removing the entry allows the gateway server to use dynamic routing protocols to determine backup routes to the network (if any).

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

To remove a directly connected route from the routing table, use the **shutdown** subcommand of the **interface** configuration command.

Defining Static Routes

A static route is a routing table entry that you make manually (as opposed to an entry made by a dynamic routing protocol). The entry remains in effect until you remove it. Note that static routes are more trouble to maintain and less reliable than dynamically determined routes. Therefore, use static routes only when the available dynamic routing processes do not adequately provide necessary routing information.

To define a static route, use the **route** configuration command. This command takes two required arguments, *network* and *gateway*, and an optional keyword, **default**. The argument *network* is the Internet address of the target network or subnet, and the argument *gateway* is the Internet address of a gateway that can reach that network. The keyword **default** marks the static route as a path toward a gateway with a more complete routing table; see "Using Default Routes" later in this chapter.

Removing a Static Route

To remove a static route from the routing table, use the **no route** configuration command. This command takes two required arguments: *network* and *gateway*. These arguments are the network or subnet address and the gateway server address of the entry to be removed. You can also remove a static route with the privileged EXEC command **clear route**, followed by *network*, the network or subnet address.

Establishing Multiple Networks or Subnets on One Interface

You can set up multiple networks or subnets on one interface with an alternative form of the **route** configuration command. For this purpose, the **route** command takes three required arguments: *network*, *interface*, and *unit*. The argument *network* is the Internet address of the target network or subnet; *interface* specifies the interface type with one of the keywords **ethernet, serial,** or **ddn-1822**; and *unit* specifies the device number of the individual interface.

For example, to set up the route to network 128.58.6.0 through the first Ethernet interface (identified as "ethernet 0"), enter the following command:

```
route 128.58.6.0 ethernet 0
```

This example command sets up a routing table entry that, in effect, superimposes another network on the "true" network.

To remove a multiple network or subnet specification, use the **no route** configuration command with the arguments *network*, *interface*, and *unit* as described above.

Dynamic Routing Using RIP

RIP uses broadcast UDP data packets to exchange routing information. Each gateway sends routing information updates every 30 seconds; this process is termed *advertising*. If a gateway does not receive an update from another gateway for 90 seconds or more, it marks the routes served by the non-updating gateway as being unusable. If there is still no update after 450 seconds, the gateway removes all routing table entries for the non-updating gateway.

The measure, or *metric*, that RIP uses to rate the value of different routes is the hop count. The *hop count* is the number of gateway-to-gateway links in a route. A directly connected network has a metric of 1; an unreachable network has a metric of 16. This small range of metrics makes RIP unsuitable as a routing protocol for wide area networks.

If the gateway server has a default network path (see "Using Default Routes" later in this chapter), RIP advertises a route that links the gateway server to the pseudo-network 0.0.0.0. The network 0.0.0.0 does not exist; RIP treats 0.0.0.0 as a network to implement the default routing feature.

To enable logging of RIP routing transactions on the console terminal, use the privileged EXEC command **debug rip**.

Dynamic Routing Using EGP

The Exterior Gateway Protocol (EGP), specified in RFC 904, is used for communicating with certain gateways in the Defense Data Network (DDN) that the U.S. Department of Defense designates as *core gateways*. An *exterior gateway* uses EGP to advertise its knowledge of routes to networks within its autonomous system. It sends these advertisements to the core gateways, which then re-advertise their collected routing information to the exterior gateway.

Note that before you can set up EGP routing, you must specify an autonomous system number using the **autonomous-system** configuration command. This command takes one required argument, *number*, which is the autonomous system number. To remove the autonomous system number, use the **no autonomous-system** configuration command.

Specifying Neighbor and Primary Gateways

A gateway using EGP cannot dynamically determine its neighbor gateways. (A *neighbor gateway* is any gateway with which the gateway server can communicate using EGP.) You must provide a list of neighbor gateways using the **egp-neighbor** configuration command. This command takes two required arguments, *gateway* and *autonomous-system*, and an optional keyword, **primary**.

The argument *gateway* is the Internet address of the neighbor gateway, and the argument *autonomous-system* is your autonomous system number. The keyword **primary** identifies *gateway* as a core gateway, called a *primary gateway*. You can define any number of EGP neighbor gateways using multiple **egp-neighbor** commands.

To remove the definition of a neighbor gateway, use the **no egp-neighbor** configuration command with the same arguments as **egp-neighbor**.

Exterior gateways can lose contact with the core gateway system. To help maintain communication, the **cisco Systems** gateway server can use EGP to communicate with any number of gateways, either core or other exterior gateways.

You can specify the maximum number of primary gateways with which a gateway server can simultaneously exchange EGP information with the **primary-neighbors** configuration command. This command takes one required argument, *number*, which is the maximum you want to specify. The default for *number* is 1, and *number* has no upper limit. To return the primary neighbors parameter to its default value, enter **"primary-neighbors 1"**.

The gateway server cycles through the list of primary gateways as EGP contact is established and lost with the primary gateways. In addition to exchanging routing information with the primary gateways, the gateway server always attempts to exchange EGP routing information with the non-primary gateways on its list of neighbors. This approach enables the gateway server to use neighbor gateways for useful routing information when the core gateways are unreliable.

Another defense against problems with core gateways is to disable the "aging" of EGP routes when the gateway server cannot contact the primary gateways. (Aging is an EGP procedure that automatically removes routing information after a certain length of time without an update.) This approach preserves old routing information when it cannot be replaced. For EGP, disabled route aging is the default. To enable route aging, use the **no hold-routes** router subcommand. To disable route aging again, use the **hold-routes** router subcommand.

An Example EGP Configuration

The following example shows how to set up an EGP routing process. The **autonomous-system** command sets the autonomous system number to 109; the **router** command sets the routing protocol to EGP; and the two **network** subcommands specify 10.0.0.0 (the DDN Arpanet network) and 192.31.7.0 (a local Ethernet) as directly connected networks. The **primary-neighbors** command specifies that the gateway server can communicate with only two of three primary gateways specified in the following **egp-neighbor** commands.

```
autonomous-system 109
router egp
network 10.0.0.0
network 192.31.7.0

primary-neighbors 2
egp-neighbor 10.3.0.27 109 primary
egp-neighbor 10.2.0.37 109 primary
egp-neighbor 10.7.0.63 109 primary
```

Monitoring EGP Operations

To display EGP transactions on the console terminal, use the privileged EXEC command **debug egp** or **debug egp-events**.

To display the status of the current EGP neighbors, use the EXEC command **show egp**. The following example shows the command output:

```
Maximum primary routers is 2, currently 2 active

  EGP Neighbor FAS/LAS State   SndSeq RcvSeq Hello Poll j/k Flags
  10.3.0.27      1/109 IDLE       625  61323    60  180   0 Primary
* 10.2.0.37      1/109 UP 12:29   250  14992    60  180   3 Primary, Act
* 10.7.0.63      1/109 UP  1d19   876  10188    60  180   4 Primary, Act
* 10.5.0.111    19/109 UP  1d19   868   1950    60  180   4 Temp, Act
* 10.4.0.96     55/109 UP 12:28   250    998    60  180   4 Temp, Pass
```

For an explanation of the EGP parameters displayed by **show egp**, see RFC 904.

Core Gateways That Use EGP

As of mid-1988, three core gateways on the Arpanet use EGP: 10.3.0.27 (ISI), 10.2.0.37 (Purdue), and 10.7.0.63 (BBN). Three core gateways on the Milnet also use EGP: 26.1.0.40 (BBN-Milnet), 26.1.0.65 (Aeronet), and 26.3.0.75 (Yuma). Contact DDN authorities or send Internet mail to gateway@BBN.COM for an updated list of core gateways that use EGP.

Dynamic Routing Using HELLO

The HELLO protocol, described in RFC 891, was developed for the Fuzzball gateways of the Distributed Computer Network project. HELLO can serve as an internal routing protocol or be used to connect to the NSF backbone network, which uses HELLO as its primary routing protocol.

The **cisco Systems** implementation of HELLO does not implement the extensive time-keeping and delay measurements. Specifically, the gateway server sets the "invalid" bit in the HELLO date field and clears the time and timestamp fields.

The metric used in HELLO is a delay value measured in milliseconds. This metric can range from 0 to 30,000 milliseconds, making HELLO a good candidate for routing larger networks. A network with a 30,000-millisecond delay is considered unreachable. The **cisco Systems** implementation uses a delay of 100 milliseconds for all routes, regardless of their actual delay characteristics.

To display HELLO routing transactions on the console terminal, use the privileged EXEC command **debug hello**.

Dynamic Routing Using IGRP

cisco Systems designed the Interior Gateway Routing Protocol (IGRP) for routing in an autonomous system having arbitrarily complex topology and consisting of media with diverse bandwidth and delay characteristics. The IGRP protocol advertises all connected and IGRP-derived networks for a particular autonomous system. A single gateway server can service up to four autonomous systems and keep the routing information for each system separate.

Interior, System, and Exterior Routes

IGRP advertises three types of routes: interior, system, and exterior. Interior routes are routes between subnets in the network attached to a gateway server interface. If the network attached to a gateway server is not subnetted, IGRP does not advertise interior routes.

System routes are routes to networks within the autonomous system. The gateway server derives system routes from directly connected network interfaces and system route information provided by other IGRP-speaking gateways. System routes do not include subnetting information.

The **network** router subcommand sets up routing for both interior and system networks; see "Setting Up Routing" earlier in this chapter for a description of the **network** subcommand.

Exterior routes are routes to networks outside the autonomous system; these networks usually have gateways that exchange routing information with the Internet core gateway system. You can specify the exterior routes that the gateway server is to advertise with the **default-network** command (see "Using Default Routes" later in this chapter).

The Gateway of Last Resort

The gateway server chooses a "gateway of last resort" from the list of exterior gateways that IGRP provides. The gateway server uses the gateway of last resort if it does not have a better route for a packet. If the autonomous system has more than one connection to an external network, different gateway servers may choose different exterior gateways as the gateway of last resort.

Metric Information

IGRP uses several types of metric information. For each path through an autonomous system, IGRP records the segment with the lowest bandwidth, the accumulated delay, the smallest MTU (Maximum Transmission Unit), and the reliability and load minima.

The IGRP metric is a 32-bit quantity that is a sum of the segment delays and the lowest segment bandwidth (scaled and inverted) for a given route. For a network of homogeneous media, this metric reduces to a hop count. For a network of mixed media (Ethernets and serial lines running from 9,600 baud to 56 kilobits/second to T1 rates), the route with the lowest metric reflects the most desirable path to a destination.

IGRP Updates

A gateway server running IGRP sends an IGRP update broadcast every 90 seconds. It declares a route inaccessible if it does not receive an update from the first gateway in the route within three update periods (270 seconds). After five update periods (450 seconds), the gateway server removes the route from the routing table. IGRP uses flash update and poison reverse to speed up the convergence of the routing algorithm.

Other IGRP Capabilities

IGRP includes several features that improve overall routing performance:

- To avoid routing loops, IGRP uses split-horizon partitioning and update hold-down.
- If multiple equal-cost routes exist to a particular destination, the gateway server running IGRP splits the load to that destination in a round-robin fashion. (See "Load-Balancing" later in this chapter.)

- To specify load-sharing that takes into account the bandwidth of different routes, IGRP offers the **variance** router subcommand (see "Load-Balancing" later in this chapter).

Monitoring IGRP Operation

To display IGRP transactions on the console terminal, use the privileged EXEC command **debug igrp**.

Removing Routing Table Entries

To remove a dynamic route from the routing table, use the privileged EXEC command **clear route**. This command takes one required argument, *network*, which is the network address portion of the routing table entry to be removed. Note that routing entries you remove with the **clear route** command may reappear unless you shut down dynamic routing with the **no router** configuration command.

To remove a directly connected route from the routing table, use the **shutdown** subcommand of the **interface** configuration command.

To remove a static route from the routing table, use the **no route** configuration command; see "Removing a Static Route" earlier in this chapter for more information.

Using Administrative Distances

In a large network, some routing protocols and some gateways can be more reliable than others as sources of routing information. By specifying administrative distance values, you enable the gateway server to intelligently discriminate between sources of routing information.

Defining Administrative Distance

An *administrative distance* is a rating of the trustworthiness of a routing information source, such as an individual gateway or a group of gateways. Numerically, an administrative distance is an integer between 0 and 255. The values 0 and 1 are reserved for connected interfaces and static routes, respectively. In general, the higher the value, the lower the trust rating. An administrative distance of 255 means the routing information source cannot be trusted at all and should be ignored.

The gateway server always uses the best routing source available: the routing source with the lowest administrative distance. For example, consider a gateway server using IGRP and RIP. Suppose you trust the IGRP-derived routing information more than the RIP-derived routing information. If you set the administrative distances accordingly, the gateway server uses the IGRP-derived information and ignores the

RIP-derived information. However, if you lose the source of the IGRP-derived information (say, to a power shutdown in another building), the gateway server uses the RIP-derived information until the IGRP-derived information reappears.

You can also use administrative distance to rate the routing information from gateways running the same routing protocol.

Assigning Administrative Distances

To define an administrative distance, use the **distance** router subcommand. This command has one required argument, *weight*, and an optional pair of arguments, *address* and *mask*.

The argument *weight* is an integer from 10 to 255 that specifies the administrative distance. (Values 0 through 9 are reserved for internal use.) Used alone, *weight* specifies a default administrative distance that the gateway server uses when no other specification exists for a routing information source. Weight values are relative; there is no quantitative method for choosing weight values.

The optional argument pair *address* and *mask* specifies a particular gateway or group of gateways to which the weight value applies. The argument *address* is an Internet address that specifies a gateway, network, or subnet. The *mask* argument (in dotted-decimal format) specifies which bits, if any, to ignore in the *address* value; a set bit in *mask* instructs the gateway server to ignore the corresponding bit in the *address* value.

To remove an administrative distance value, use the **no distance** subcommand. This subcommand takes the same arguments as **distance**.

Examples of Using Administrative Distances

In the example below, the **router** configuration command sets up IGRP routing in autonomous system number 109. The **network** subcommands specify routing on networks 192.31.7.0 and 128.88.0.0. The first **distance** subcommand sets the default administrative distance to 255, which instructs the gateway server to ignore all routing updates from gateway servers for which an explicit distance has not been set. The second **distance** subcommand sets the administrative distance for all gateway servers on the Class C network 192.31.7.0 to 90. The third **distance** subcommand sets the administrative distance for the gateway server with the address 128.88.1.3 to 120.

```
router igrp 109
network 192.31.7.0
network 128.88.0.0
distance 255
distance 90  192.31.7.0  0.0.0.255
distance 120  128.88.1.3  0.0.0.0
```

The order in which you enter **distance** subcommands can affect the assigned administrative distances in unexpected ways. For example, the following subcommands assign the gateway server with the address 192.31.7.18 an administrative distance of 100, and all other gateway servers on subnet 192.31.7.0 an administrative distance of 200.

```
distance 100 192.31.7.18  0.0.0.0
distance 200 192.31.7.0  0.0.0.255
```

However, if you reverse the order of these subcommands, all gateway servers on subnet 192.31.7.0 are assigned an administrative distance of 200, even the gateway server at address 192.31.7.18.

```
distance 200 192.31.7.0  0.0.0.255
distance 100 192.31.7.18  0.0.0.0
```

Assigning administrative distances is a problem unique to each network and is done in response to the greatest perceived threats to the connected network. Even when general guidelines exist, the network manager must ultimately determine a reasonable matrix of administrative distances for the network as a whole.

Redistributing Routing Information

In addition to running multiple routing protocols simultaneously, the gateway server can redistribute information from one routing protocol to another. For example, you can instruct the gateway server to re-advertise HELLO-derived routes using the RIP protocol, or to re-advertise static routes using the IGRP protocol.

However, the metrics of some routing protocols do not translate easily to the metrics of other protocols. For example, the RIP metric is a hop count, the HELLO metric is a delay, and the IGRP metric is a combination of five quantities. Also, exchanging routing information between different routing protocols can create routing loops, which can seriously degrade network operation (see "Avoiding Routing Loops" later in this chapter). The gateway server provides configuration commands to maximize routing flexibility under these constraints.

Supported Metric Translations

The gateway server does not support all possible redistributions because of the incompatibilities among routing protocols. The following paragraphs describe the supported redistributions for each routing protocol. These descriptions assume that you have not defined a default redistribution metric, which replaces metric conversions (see "Setting Default Metrics" later in this section).

RIP can redistribute static routes and HELLO-derived routes. RIP assigns static routes a metric of 1 (directly connected) and converts HELLO metrics in accordance with Table 12-1.

Table 12-1. RIP and HELLO Metric Transformations

From HELLO	To RIP	From RIP	To HELLO
0	0	0	0
1-100	1	1	100
101-148	2	2	200
149-219	3	3	300
220-325	4	4	325
326-481	5	5	481
482-713	6	6	713
714-1057	7	7	1057
1058-1567	8	8	1567
1568-2322	9	9	2322
2323-3440	10	10	3440
3441-5097	11	11	5097
5098-7552	12	12	7552
7553-11190	13	13	11190
11191-16579	14	14	16579
16580-24564	15	15	24564
24565-30000	16	16	30000

Note: Values are derived from the mapping function defined by Dave Mills and the "gated" developers at the Cornell University Theory Center.

EGP can redistribute static routes and RIP-, HELLO-, and IGRP-derived routes. EGP assigns the metric 3 to all static and derived routes. EGP cannot re-advertise EGP-derived routes.

HELLO can redistribute static routes and RIP- and IGRP-derived routes. HELLO assigns static routes a metric of 100 (directly connected) and converts RIP metrics in accordance with Table 12-1. HELLO advertises IGRP-derived routes with a metric equal to the phase delay portion of the IGRP metric or to 100, whichever is larger. Ethernets have a phase delay of 1 millisecond and serial links have a phase delay of 20 milliseconds; HELLO assigns a metric of 100 to routes using these media.

IGRP can redistribute static routes and information from other IGRP-routed autonomous systems. IGRP assigns static routes a metric that identifies them as directly connected. IGRP does not change the metrics of routes derived from IGRP updates from other autonomous systems. Note that IGRP can redistribute other routing protocols if a default metric is in effect (see "Setting Default Metrics" later in this section).

Using the **redistribute** Subcommand

By default, the gateway server does not exchange information among different protocols. If you want to pass routing information among routing protocols, use the **redistribute** router subcommand. This command takes one required argument, *protocol*, and one optional argument, *autonomous-system*. The argument *protocol*

specifies a routing information source using one of the keywords **static, rip, egp, hello,** or **igrp.** If you specify **igrp,** you must also specify *autonomous-system*, the autonomous system number.

For example, to redistribute RIP-derived information using the HELLO protocol, enter:

```
router hello
redistribute rip
```

To end redistribution of information from a routing protocol, use the **no redistribute** subcommand. This subcommand takes the same arguments as **redistribute.**

Setting Default Metrics

The **default-metric** router subcommand, used in conjunction with the **redistribute** router subcommand, causes the current routing protocol to use the same metric value for all redistributed routes. (Redistributed routes are those routes established by other routing protocols.) A default metric helps solve the problem of redistributing routes with incompatible metrics; whenever metrics do not convert, using a default metric provides a reasonable substitute and enables the redistribution to proceed.

The **default-metric** subcommand has two forms, depending on the routing protocol specified in the **redistribute** subcommand. For RIP, EGP, and HELLO, which use scalar, single-valued metrics, the subcommand takes one required argument, *number.* This argument is the default metric value (an unsigned integer) appropriate for the specified routing protocol.

For IGRP, the **default-metric** subcommand takes five required arguments: *bandwidth*, *delay*, *reliability*, *loading*, and *mtu*. The argument *bandwidth* is the minimum bandwidth of the route in kilobits/second, *delay* is the route delay in tens of microseconds, *reliability* is the likelihood of successful packet transmission expressed as a number between 0 and 255 (255 is 100% reliability), *loading* is the effective bandwidth of the route in kilobits/second, and *mtu* is the minimum MTU (Maximum Transmission Unit) of the route.

For example, consider a gateway server in autonomous system 109 using both the HELLO and IGRP routing protocols. To advertise IGRP-derived routes using the HELLO protocol and to assign the IGRP-derived routes a HELLO metric of 10,000, the configuration commands are:

```
router hello
default-metric 10000
redistribute igrp 109
```

Filtering Redistributed Routing Updates

The gateway server provides the **distribute-list** configuration command for filtering incoming and outgoing routing updates. Filtering can isolate sources of misleading information and help maintain consistency when redistributing information among

multiple routing protocols. This command takes one required argument, *access-list*, and a keyword. The argument *access-list* is the number of an access list as described in Chapter 10, "Adding Security Measures". The keywords **in** and **out** specify whether the filtering applies to incoming or outgoing routing updates, respectively.

When you apply an access list to incoming routing updates, the gateway server ignores updates with addresses not permitted by the access list. In this way, you can eliminate updates from certain gateways or groups of gateways.

When you apply an access list to outgoing routing updates, the gateway server broadcasts update information only for routes that are permitted by the access list. You can use this function to filter the information redistributed among routing protocols.

To end the filtering of routing updates, use the **no distribute-list** command. This command takes the same arguments as **distribute-list.**

Example of Static Route Redistribution

Consider a network attached to a gateway that does not use a dynamic routing protocol. Suppose the gateway address is 192.31.7.65 and the network address is 192.1.2.0. The following command sets up a static route on a **cisco Systems** gateway server:

```
route 192.1.2.0  192.31.7.65
```

To configure the gateway server to use IGRP in autonomous system 109, the command is:

```
router igrp 109
```

Then, the commands to re-advertise the static route are:

```
network 192.31.7.0
redistribute static
```

Now suppose that of several static routes defined, you want to redistribute only those connecting to networks 192.31.7.0 and 192.1.2.0. This task requires the **distribute-list** command and the definition of access lists. The following commands perform the task:

```
distribute-list 3 out
access-list 3 permit 192.31.7.0  0.0.0.255
access-list 3 permit 192.1.2.0 0.0.0.255
```

cisco Systems recommends that any redistributed static routes appear on a single gateway as static routes to minimize the likelihood of creating a routing loop.

Example of RIP and HELLO Redistribution

Consider a wide area network at a university that uses RIP as an interior routing protocol. Assume the university wants to connect its wide area network to a regional network, 128.1.1.0, which uses HELLO as the routing protocol. The goal in this case is to advertise the networks in the university network to the gateways on the regional network. The commands for the interconnecting gateway server are:

```
router hello
network 128.1.1.0
redistribute rip
default-metric 10000
distribute-list 10 out
```

In this example, the **router** command starts a HELLO routing process. The **network** subcommand specifies that network 128.1.1.0 (the regional network) is to receive HELLO routing information. The **redistribute** subcommand specifies that RIP-derived routing information be advertised in the HELLO routing updates. The **default-metric** subcommand assigns a HELLO delay of 10,000 to all RIP-derived routes.

The **distribute-list** command instructs the gateway server to use access list 10 (not defined in this example) to limit the entries in each outgoing HELLO update. The access list prevents unauthorized advertising of university routes to the regional network.

This example could have specified automatic conversion between the RIP and HELLO metrics. However, in the interest of routing table stability, it is not desirable to do so. Instead, this example limits the routing information exchanged to availability information only.

Example of IGRP Redistribution

Each IGRP routing process can provide routing information to only one autonomous system; the gateway server must run a separate IGRP process and maintain a separate routing database for each autonomous system it services. However, you can transfer routing information between these routing databases.

Suppose the gateway server has one IGRP routing process for network 15.0.0.0 in autonomous system 71 and another for network 192.31.7.0 in autonomous system 109, as the following commands specify:

```
router igrp 71
network 15.0.0.0

router igrp 109
network 192.31.7.0
```

To transfer a route to 192.31.7.0 to the database of the first routing process (without passing any other information about autonomous system 109), use the following commands:

```
router igrp 71
redistribute igrp 109
distribute-list 3 out
access-list 3 permit 192.31.7.0  0.0.0.255
```

Using Default Routes

A gateway server may not be able to determine the routes to all other networks. Therefore, to provide complete routing capability, use some gateways as "smart gateways" and give the remaining gateways *default routes* to the smart gateways.

Smart gateways, which are usually exterior gateways using EGP, know about many routes; the remaining gateways store only routes for their own autonomous system. The remaining gateways use the default routes to forward data packets for which they have no explicit routing information. Eventually, the packets reach a gateway with more routing information.

IGRP determines default routes automatically and dynamically. Gateway servers running IGRP automatically pass routes to exterior networks to other gateway servers in the autonomous system. The gateway servers dynamically pick the best default route to use for their otherwise unroutable packets. If a default route disappears, a gateway server automatically selects another route in its place.

Setting a Default Route

If your gateway server or the exterior gateways use a dynamic routing protocol other than IGRP, you must use the **default-network** command to specify candidate default routes. This command takes one required argument, *network-address*, which specifies a network address in dotted-decimal format.

You can specify multiple default networks with the **default-network** command. The gateway server periodically scans its routing table to choose the optimal default network as its default route. The gateway server uses both administrative distance and metric information to determine the default route. This approach adds dynamic flexibility to the selection of default routes and is compatible with all routing protocols, including static routes.

If you give *network-address* a subnet address value (where the host portion is zero and the subnet portion is not zero), the gateway server uses the value as a "subnet default route"--a path of last resort when the gateway server has no specific routing information for a subnetted network.

If *network-address* specifies one of the networks connected to the gateway server, and if the current routing protocol is RIP or HELLO, the gateway server advertises the default route as 0.0.0.0. If the current routing protocol is IGRP, the gateway server advertises the specified directly connected network as an exterior network.

Setting a Static Default Route

You can establish a static default route with the optional keyword **default** of the **route** configuration command, as described in "Defining Static Routes" earlier in this chapter. For example, the following command instructs the gateway server to send all legal but unroutable packets to the smart gateway attached to network 10.0.0.0 through the gateway at address 192.31.7.33:

```
route 10.0.0.0 192.31.7.33 default
```

The second address need not be that of an exterior gateway; it may be a first-hop address or an intermediate-hop address to an exterior gateway. The fundamental requirement of this default route is that all packets encounter a smart gateway on their way to network 10.0.0.0; the smart gateway reroutes the packets to the proper destination.

You can specify only one default route using the **route** command; use the **default-network** command to specify more than one default route.

Avoiding Routing Loops

A routing loop occurs when two or more gateways route a packet back and forth, delaying or preventing progress toward its destination. Routing loops can be short-term problems caused by a temporary inconsistency in the routing tables of two or more gateways. In these cases, the data packet moves around the loop a few times and then proceeds to its destination. Short-term routing loops waste resources and annoy network users, but do not cause serious problems.

Long-term routing loops are a serious problem. In long-term routing loops, a data packet circulates until a gateway decreases its time-to-live field to zero, resulting in the elimination of the packet. Breaking these loops requires manual intervention in the form of skillful route debugging.

The gateway server provides dynamic routing algorithms that can set up loop-free routing in a single autonomous system of any topology. However, to prevent routing loops, the algorithms must have total control of the routing information; if you specify static routes or the redistribution of routing information, you undermine the automatic functioning of the algorithms. In general, if you alter any dynamic routing information, you may introduce routing loops.

The **route** configuration command and the **redistribute** router subcommand are powerful network configuration tools; use them with great care. **cisco Systems** recommends that you use the **distribute-list** filtering command and the **distance** router subcommand to minimize the amount of routing information that a gateway server advertises and accepts.

Load-Balancing

The gateway server can increase overall network efficiency and reliability through load-balancing: the splitting of network traffic over multiple routes, termed *parallel paths*. Load-balancing increases packet throughput and provides redundancy.

To make load-balancing effective, you must specify how the gateway server chooses parallel paths. The gateway server uses a *variance* value and *capacity* values to determine which routes to group as parallel paths.

The variance value is a multiplier that determines a range of metric values. The gateway server accepts a route as a parallel path if its metric falls within the specified range.

The capacity values rate the traffic-handling capacity of the various network segments served by the various reachable gateways. These ratings are relative: a capacity of 20 indicates twice the traffic-handling capability of a capacity of 10.

For example, suppose you have two serial lines to a particular destination: a 56 kilobits/second line and a 9.6 kilobits/second line. To maximize throughput, you want to use both lines, not just the faster line. You would specify a variance multiplier that caused the gateway server to classify these serial lines as parallel paths.

Specifying Variance

By default, the gateway server chooses only routes with the same metric as parallel paths. The gateway server splits traffic evenly among these parallel paths.

You can use the **variance** router subcommand to have the gateway server use parallel paths with different metrics. This subcommand takes one required argument, *multiplier*, which is an integer that specifies a range. A metric value falls in the range if it lies between the lowest-cost metric value (numerically smallest, indicating the highest bandwidth) and the lowest-cost metric value times *multiplier*. The lowest-cost metric value depends on the group of candidate parallel paths. The default *multiplier* value is 1.

To return the effective variance value to the default, use the **no variance** subcommand.

For the two serial lines in the last example, the bandwidths were 56 kilobits/second and 9.6 kilobits/second. Assuming IGRP is the routing protocol, a *multiplier* value of 6 would group these lines as parallel paths. This value derives from the facts that IGRP metrics are inversely proportional to bandwidth and 56 is roughly six times 9.6.

cisco Systems recommends that you use the **variance** subcommand with caution. Values greater than 3 can cause routing instabilities in complex networks.

Setting Capacity

The IGRP routing protocol provides bandwidth and delay information that supports allocating load according to route capacity; other routing protocols do not provide this kind of information.

To supply capacity information for use with non-IGRP protocols, use the **capacity** router subcommand. This subcommand takes one required argument, *weight*, and an optional pair of arguments, *address* and *mask*.

The argument *weight* is an integer that indicates relative capacity to process network traffic; the higher the number, the greater the traffic-handling capacity. The default *weight* for all routing protocols is 10.

Used alone, *weight* becomes the default capacity for the routing protocol specified in the last **router** command. The gateway server uses this default value when it has no specific capacity information for a gateway.

The optional argument pair *address* and *mask* specifies a particular gateway or group of gateways to which the weight value applies. The argument *address* is an Internet address that specifies a gateway, network, or subnet. The argument *mask* (in dotted-decimal format) specifies which bits, if any, to ignore in the *address* value; a set bit in *mask* instructs the gateway server to ignore the corresponding bit in the *address* value. To remove capacity information, use the **no capacity** subcommand. This subcommand takes the same arguments as **capacity**.

To display capacity-related information, use the EXEC command **show protocols**.

Splitting Load Among Parallel Paths

IGRP automatically splits network traffic among parallel paths in proportion to the different metrics. Other protocols must use the weights defined by the **capacity** router subcommand to make load-proportioning decisions. All **cisco Systems**-supported routing protocols balance network traffic over parallel paths where the paths have the same metric.

However, proportional load-splitting does not always make sense. Consider a T1 line (1.544 megabits/second) and a 9.6 kilobits/second line: the bandwidths differ by about 160 to 1. In this case, it makes no sense to attempt to balance data sent over two such unequal circuits; the 9.6 kilobits/second line would add almost nothing to the overall bandwidth. The upper limit for load-sharing with a 9.6 kilobits/second line is a 19.2 kilobits/second line or a 56 kilobits/second line.

Note that the **capacity** router subcommand binds the capacity number to a gateway, not to a network advertised by that gateway. If a gateway connects to one network with a 9,600 baud line and to another with a T1 line, the **capacity** subcommand does not allow the gateway server to discriminate between the networks. In this situation, use IGRP to provide the proportional load-sharing.

An Example of Load-Balancing

In the following example, consider a network that has mostly 56 kilobits/second lines and some 19.2 and 9.6 kilobits/second lines. Approximating the relative capacities of the 56, 19.2, and 9.6 kilobits/second lines as 5:2:1, we can assign the weights 20, 8, and 4, respectively. The corresponding configuration commands are:

```
router rip
capacity 20
capacity 20 128.1.2.3  0.0.0.0
capacity  8 128.1.2.4  0.0.0.0
capacity  4 128.1.2.5  0.0.0.0
```

The **router** command starts a RIP routing process. The first **capacity** subcommand sets the default capacity to 20; the remaining three **capacity** subcommands set capacities of 20, 8, and 4 for three individual gateways.

If the three gateways advertise paths to the same network, the gateway server splits traffic to that network in a 5-2-1 pattern. The gateway server sends five packets to 128.1.2.3, then two packets to 128.1.2.4, then one packet to 128.1.2.5, and then repeats the pattern.

Monitoring and Debugging Routing Operations

The gateway server provides **show** and **debug** commands that report useful information for observing routing activities and tracking down problem sources. See Chapter 18, "Network Monitoring and Troubleshooting", for descriptions of **show** and **debug** commands in general.

Displaying the Routing Table

You can display the current state of the gateway server routing table with the EXEC command **show route**. In the following example output, most of the information is self-explanatory; the exception is the metric information enclosed in square brackets. The first number in the brackets is the administrative distance of the information source; the second number is the metric for the route.

```
chaff#show route

Codes: I - IGRP derived, R - RIP derived, E - EGP derived
       C - connected, S - static, X - Chaos derived, H - HELLO derived
       * - candidate exterior route

Gateway of last resort is 192.31.7.18 to network 10.0.0.0
S Net 9.0.0.0 via 192.31.7.25, 1236 uses
I*Net 10.0.0.0 [100/504100] via 192.31.7.18, 15 sec, 40 uses, Ethernet0
R Net 15.0.0.0 [120/1] via 192.31.7.18, 24 sec, 0 uses, Ethernet0
C Net 192.31.7.0 is subnetted (mask is 255.255.255.240), 6 subnets
I     192.31.7.80 [100/2200] via 192.31.7.21, 71 sec, 0 uses, Ethernet0
I     192.31.7.64 [100/1200] via 192.31.7.21, 71 sec, 13 uses, Ethernet0
I     192.31.7.96 [100/1300] via 192.31.7.21, 71 sec, 282 uses, Ethernet0
C     192.31.7.16 is directly connected, 149301 uses, Ethernet0
C     192.31.7.48 is directly connected, 0 uses, Serial0
I     192.31.7.32 [100/502100] via 192.31.7.18, 15 sec, 0 uses, Ethernet0
```

Displaying Routing Information for a Network or Subnet

You can display detailed routing information for a network or subnet by specifying the network or subnet address as an argument of the **show route** command. For example:

```
chaff#show route 10.0.0.0

Routing entry for 10.0.0.0 (mask 255.0.0.0)
  Known via "igrp 109", distance 100, metric 504100
  Last update from 192.31.7.18, 23 seconds ago
  Routing Descriptor Blocks:
  * 192.31.7.18, from 192.31.7.18, 23 seconds ago, 40 uses, via Ethernet0
      Route metric is 504100, traffic share count is 1
      Total delay is 41000 microseconds, minimum bandwidth is 20 Kbit
      Reliability 255/255, minimum MTU 1008 bytes
```

Displaying Routing Protocol Parameters and Status

You can display the parameters and current state of the active routing protocol process with the EXEC command **show protocols**. Example output follows:

```
Routing Protocol is "igrp 109"
  Sending updates every 90 seconds, next due in 59 seconds
  Invalid after 270 seconds, hold down for 280, flushed after 630
  Outgoing distribution access list is not set
  Incoming distribution access list is not set
  Allowed variance in multi-path metrics is 1
  Redistributing: static, igrp 109
  Routing for Networks:
    192.31.7.0
  Routing Information Sources:
    Gateway         Distance  Capacity  Last Update
    192.31.7.18          100        10      22:26:38
    192.31.7.20          100        10      6d17
    192.31.7.21          100        10      22:38:13
    192.31.7.33          100        10      0:01:28
  Distance and Capacity: (defaults are 100 and 10)
    Gateway                Mask    Distance  Capacity
   192.31.7.0        0.0.0.255         100        10
```

The information displayed by **show protocols** is useful in debugging routing operations. For example, the information in the Routing Information Sources section of the **show protocols** output can help you identify a gateway suspected of delivering bad routing information.

Using the **debug** and **logging** Commands

The **debug** and **logging** commands enable you to record useful routing information. Using the **debug** command, you can instruct the gateway server to log any combination of RIP, EGP, HELLO, and IGRP routing events as well as routing table events to the console terminal. The **logging** command can record routing transactions in a log file on a time-sharing host. See Chapter 18, "Network Monitoring and Troubleshooting", for more information on the **debug** and **logging** commands.

cisco Systems

Chapter 13
Using Other Internet Services

The **cisco Systems** gateway server supports several Internet services and options required for proper operation.

ICMP Messages

The **cisco Systems** gateway server supports the Internet Control Message Protocol (ICMP), an integral part of the Internet Protocol. ICMP provides message packets to report changes in packet processing. See RFC 792 for a complete description of ICMP message packets.

If the gateway server cannot route a packet, the gateway server sends an ICMP *Host Unreachable* message to the source. The gateway server cannot route a packet if it lacks routing information, encounters an inoperative interface, receives an 1822 Host Down message, or fails an administrative access control.

The gateway server does not send ICMP *Network Unreachable* messages because the source host may misinterpret the message. For example, if the source host is unaware of subnets on a network, it may interpret an ICMP Network Unreachable message as meaning a whole network is unavailable when in fact a single subnet is down. To avoid misinterpretations, the gateway server sends only ICMP Host Unreachable messages.

Note that during an attempt to create a TCP connection, the gateway server responds to ICMP Host Unreachable and Network Unreachable messages by trying different routes. However, the gateway server ignores these messages during an active connection because they may indicate a temporary condition. The gateway server determines on its own when to try alternate routes or abandon a connection.

If the gateway server receives a non-broadcast packet addressed to itself and that packet uses a protocol the gateway server does not recognize, it sends an ICMP *Protocol Unreachable* message to the source.

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

If the only route for a packet is out the same interface it came in on, and if the packet source is on the same network or subnet as the receiving interface, the gateway server sends an ICMP *Host Redirect* message to the packet source. The ICMP Host Redirect message indicates which local gateway the source host should use. To turn off ICMP Host Redirect messages, use the **no redirects** subcommand of the **interface**

configuration command. The gateway server ignores incoming ICMP Host Redirect messages; the gateway server assumes it has better information.

If the gateway server encounters a problem while processing the Internet header options of a packet, it sends an ICMP *Parameter Problem* message to the source of the packet and discards the packet.

If a packet hold queue overflows and the gateway server must discard packets, it sends ICMP *Source Quench* messages to the sources of incoming packets.

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

If the gateway server receives an ICMP *Timestamp Request* or *Information Request* message, it responds with an ICMP *Timestamp Reply* or *Information Reply* message, respectively. Because it does not keep universal time, the gateway server sets the "nonstandard value" bit in ICMP Timestamp Reply messages.

During the process of obtaining configuration information from network hosts, the gateway server sends broadcast ICMP *Mask Request* messages to determine subnet definitions for the local networks. A gateway server responds to an ICMP Mask Request message with the ICMP *Mask Reply* message.

The gateway server can waste much time retransmitting ICMP message packets if one or more hosts ignore them. To mitigate this problem, the gateway server limits the rate at which it sends ICMP Host Redirect, Source Quench, Host Unreachable, and Network Unreachable messages to one per second per interface.

Boot Protocol Messages

The Boot Protocol (BootP), described in RFC 951, specifies a method for determining the Internet address of a host given its Ethernet hardware address. The basic mechanism is similar to RARP (see "Address Resolution" in Chapter 11, "Understanding Internet Addresses, Broadcasts, and Address Resolution"), but is UDP-based rather than a distinct Ethernet protocol. This difference enables gateways to route BootP messages to non-local networks, whereas RARP messages cannot leave the local Ethernet-based network. Thus, instead of requiring a RARP server for each network segment, one BootP server can serve an entire network.

Using BootP, gateway servers can assist isolated hosts by forwarding BootP messages to server hosts. The gateway server forwards BootP messages with a delay field value greater than five to the helper address. To specify a helper address, use the **helper-address** interface subcommand, as described in "Selecting and Setting Broadcast Information" in Chapter 7, "Configuring the Network Interfaces".

Internet Header Options

The **cisco Systems** gateway server supports the Internet header options *Strict Source Route*, *Loose Source Route*, *Record Route*, *Security*, *Stream ID*, and *Time Stamp*.

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

You can specify several of the Internet header options in the extended command mode of the privileged EXEC command **ping**. To start the extended command mode, type **y** at the extended commands prompt. To display a list of the options you can specify, type **?** at the extended commands prompt.

To specify the Loose Source Route option when making a Telnet connection, use the optional keyword **/route** with the **connect** or **telnet** command. This keyword takes an argument, *path*, which is a list of host names or addresses separated by spaces. The list forms a loose source route to the ultimate destination; the list must include the host name or address of the ultimate destination. Incoming TCP connections with source routing use the source route information to make the connection.

Telnet Options

The gateway server supports the Telnet options *Echo*, *Binary Transmission*, *Suppress Go Ahead*, *Terminal Type*, and *Send Location*. You can specify a detailed display of Telnet option negotiations by using the optional keyword **debug** with the **connect** or **telnet** command when you make a Telnet connection. The information displayed is useful in tracking down Telnet-related problems.

Host-Name-to-Address Conversion

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

Defining Static Name-to-Address Mappings

To define a static host-name-to-address mapping, use the **host** configuration command. This command takes one required argument, *name*, one optional argument, *port*, and up to eight address arguments (the first address argument is required). The argument *name* is the host name, and the argument *port* is the port number (in decimal). If you do not specify a port number, the gateway server uses the Telnet protocol port (decimal 23).

The following examples illustrate the **host** command.

```
host chaff 192.31.7.18
host cisco-gw 10.2.0.2 192.31.7.33
```

Using Dynamic Name Lookup

You can specify that the Domain Name System (DNS) or IEN-116 Name Service automatically determine host-name-to-address mappings. The **name-server, service domain, domain-name,** and **service ipname** commands establish different forms of dynamic name lookup.

To specify one or more hosts that supply name information, use the **name-server** configuration command. This command takes one to six arguments that specify the addresses of name-serving hosts. If you do not use the **name-server** command, the gateway server uses the all-ones broadcast address (255.255.255.255).

By default, the gateway server uses the Domain Name System to determine the addresses corresponding to host names. The **no service domain** configuration command disables use of the Domain Name System; the **service domain** command re-enables use of the Domain Name System.

To define a default domain name, use the **domain-name** configuration command. This command takes one required argument, *name*, which is the default domain name. The gateway server uses the default domain name to complete unqualified domain names (names without a dotted domain name such as ".CISCO.COM"). The **no domain-name** command removes the default domain name, if any.

To specify the use of the IEN-116 Name Service to determine the addresses corresponding to host names, use the **service ipname** configuration command. The **no service ipname** command disables use of the IEN-116 Name Service.

Removing Name-to-Address Mappings

To remove one or all of the host names from the cache, use the privileged EXEC command **clear host**. This command takes one argument, *name*. If you use a host name as the *name* argument, the gateway server removes that host-name-to-address mapping from the cache. If you use an asterisk (*) as the argument, the gateway server removes all host-name-to-address mappings from the cache.

Examining the Name-to-Address Cache

To display the name-to-address cache, use the EXEC command **show hosts**. See Chapter 18, "Network Monitoring and Troubleshooting", for an example of **show hosts** output.

The "Little Services"

The gateway server provides the TCP and UDP "little services" Echo and Discard. These services are described in RFC 862 and RFC 863.

cisco Systems

Chapter 14
Using DDN Protocols

cisco Systems supports three approaches for connecting the gateway server to a Packet Switch Node (PSN) or an Interface Message Processor (IMP) on the Defense Data Network (DDN): the 1822-LH/DH protocol, the HDH (1822-J) protocol, and the DDN X.25 Standard. This chapter describes these DDN protocols.

Note that every DDN gateway supports the Exterior Gateway Protocol (EGP) for exchanging routing information with the DDN core gateways. See "Dynamic Routing Using EGP" in Chapter 12, "Routing With the Gateway Server", for details on the **cisco Systems** EGP implementation.

Using the 1822-LH/DH Protocol

The 1822-LH/DH protocol is the original hardware specification and software protocol for attaching hosts to PSNs on the DDN. Many existing DDN attachments still use 1822-LH/DH. Newer attachments must use the DDN X.25 Standard described later in this chapter.

The 1822-LH/DH hardware specification has two forms, *Local Host* and *Distant Host*. These forms differ only in the jumper configuration on the 1822 interface; there is no software difference. See "DDN 1822-LH/DH (CSC-A) Interface Hardware" in Chapter 7, "Configuring the Network Interfaces", for a description of the interface hardware.

You can display information about the 1822 hosts with which the gateway server has communicated in the past five minutes with the privileged EXEC command **show imp-hosts**. This command displays the host address, the number of seconds of inactivity, the current RFNM (Request For Next Message) count, the queued packets count, the total number of packets sent, and the 1822 operational status. Example command output follows.

```
Host          Seconds   RFNM   Queued      Sent   State
10.1.0.11         140      0        0         1      up
10.2.0.9            3      0        0         1    down
10.1.0.2            1      0        0     10891      up
10.3.0.11         408      0        0        74      up
10.7.0.20          22      0        0       655      up
10.1.0.17           5      0        0        37      up
10.2.0.37          68      0        0       982      up
```

To display PSN-related events such as PSN up/down transitions, use the privileged EXEC command **debug psn.**

Using the HDH Protocol

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 the Multi-port Communications Interface described in "Multi-port Communications Interface Hardware" in Chapter 7, or the CSC-D (formerly called the CSC-H) serial interface described in "Serial Network Interface Hardware" in Chapter 7.

The gateway server supports both the *message* and *packet* modes of HDH. To set the mode, use the **hdh** subcommand of the **interface** configuration command. This subcommand takes one argument, either the keyword **packet** (the default) or the keyword **message.**

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.**

The **show imp-hosts** and **debug psn** commands described in "Using the 1822-LH/DH Protocol" earlier in this chapter also apply to gateway servers using the HDH protocol.

Using the DDN X.25 Standard

The DDN X.25 Standard is the required protocol for use with DDN PSNs. An earlier X.25 variant, DDN X.25 Basic, permitted hosts to communicate only if both hosts used DDN X.25 Basic. DDN X.25 Standard removes this limitation and enables hosts to communicate with hosts using HDH or 1822-LH/DH. The Defense Communications Agency (DCA) has certified the **cisco Systems** DDN X.25 Standard implementation for attachment to the Defense Data Network.

The DDN X.25 hardware is the Multi-port Communications Interface described in "Multi-port Communications Interface Hardware" in Chapter 7, or the CSC-D serial interface described in "Serial Network Interface Hardware" in Chapter 7.

To enable DDN X.25 encapsulation on an interface, use the **encapsulation ddnx25** interface subcommand.

See Chapter 15, "Using X.25 Protocols", for more information on **cisco Systems** X.25 support.

cisco Systems

Chapter 15
Using X.25 Protocols

The International Telegraph and Telephone Consultative Committee (CCITT) published the X.25 Recommendation for connections between data terminal equipment (DTE) and data communications equipment (DCE or network) in 1984. The Defense Data Network (DDN) and the International Standards Organization (ISO) require the use of X.25 protocol for computer communications.

The X.25 model is a telephone network for computer data communication. 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 can provide data communications using protocols such as the Internet Protocol (IP), DECnet, and XNS.

X.25 Level 2 (LAPB)

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. **cisco Systems** provides LAPB in all gateway servers with serial interfaces; you can use LAPB instead of the default HDLC encapsulation. Note that the additional overhead of LAPB reduces the effective data bandwidth of the link.

Using LAPB under noisy conditions can result in greater throughput. When LAPB detects a damaged frame, the gateway server immediately retransmits the frame instead of waiting for host timers to expire. However, if the line is not noisy, the lower overhead of HDLC encapsulation is more efficient.

Starting LAPB Encapsulation

The X.25 Recommendation distinguishes between two types of X.25 hosts: data terminal equipment (DTE) hosts and data communications equipment (DCE) hosts. A gateway server using LAPB encapsulation can act as a DTE or DCE device at the protocol level.

Note: For the simple case of connecting a gateway server to a public network, specify X.25 (Level 3) encapsulation. This encapsulation method sets LAPB encapsulation for a DTE device. To communicate over a leased line using LAPB encapsulation, one host must be configured as a DTE and the other as a DCE.

To set DTE or DCE operation, use the **encapsulation** subcommand of the **interface** configuration command. This subcommand takes one keyword as an argument: **lapb** to set DTE operation, or **lapb-dce** to set DCE operation.

The **lapb** and **lapb-dce** options require the data to be in Internet Protocol format. 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. 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**.

Setting LAPB Parameters

After you establish LAPB encapsulation, you can set LAPB parameters with the **lapb** 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*. See the CCITT X.25 Recommendation for more information on X.25 parameters.

Setting the Retransmission Timer

The retransmission timer determines how long a transmitted frame can remain unacknowledged before the gateway server polls for an acknowledgment. To set the limit for the retransmission timer (the LAPB T1 parameter), use the **lapb t1** interface subcommand. This subcommand takes a time value in milliseconds as its argument; the default is 3,000 milliseconds.

For X.25 networks, the gateway server 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 a maximum-sized frame to complete one round trip on the link. If the timer setting is too small, the gateway server will poll before the acknowledgment frame can return, which results in an effective loss of bandwidth. If the timer setting is too large, the gateway server 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 1.5. 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 the Hold Timer

The hold timer determines how long to delay before sending an acknowledgment frame for a received data frame. (Whenever possible, the gateway server merges acknowledgments with packets carrying other information.) To set the limit for the hold timer (the LAPB TH parameter), use the **lapb th** interface subcommand. This subcommand takes a time value in milliseconds as its argument; the default is 2,000 milliseconds. The hold timer value must be less than or equal to the retransmission timer (T1) value.

To cause default LAPB behavior, set the hold timer value to that of the retransmission timer. To force an acknowledgment for every information frame received, set the hold timer value to 0. Note that smaller hold timer values increase responsiveness at the expense of higher line overhead.

Setting Frame Parameters

To specify the maximum number of bits a frame can hold, use the **lapb n1** interface subcommand. This subcommand takes a number of bits, which must be a multiple of eight, as its argument; the default is 12,000 bits (1,500 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. This subcommand takes a retransmission count as its argument; the default is 20 retransmissions.

To specify the maximum permissible number of outstanding frames, called the *window*, use the **lapb k** interface subcommand. This subcommand takes a frame count from 1 to 7 as its argument; the default is 7 frames.

Using LAPB Over Leased-Line Links

You can use LAPB as a packet encapsulation method over a leased-line link if both ends support LAPB. You must configure one end as a DTE device and the other as a DCE device.

If a LAPB link becomes congested, the gateway server stores outgoing frames in a hold queue until the link returns to normal or the hold queue becomes full. To set the size of the hold queue, use the **hold-queue** interface subcommand; see "Controlling Interface Hold Queues" in Chapter 7, "Configuring the Network Interfaces". If the hold queue overflows, the gateway server sends an ICMP Source Quench message to the source of the overflow-causing frame and discards the frame.

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).

```
interface serial 3
encapsulation lapb-dce
lapb t1 4000
```

Displaying LAPB Statistics

To display operation statistics for an interface using LAPB encapsulation, use the EXEC command **show interface**. 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 CCITT X.25 Recommendation.

Debugging LAPB Encapsulation

To debug LAPB encapsulations, you must have a good understanding of the CCITT 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 O 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 O REJSENT (2) REJ P/F 1
Serial0: LAPB I REJSENT (2) DM F (C)
Serial0: LAPB I DISCONNECT (2) SABM (C)
Serial0: LAPB O CONNECT (2) UA
.
.
.
Serial0: LAPB T SABMSENT 357964 0
Serial0: LAPB O SABMSENT (2) SABM P
```

In the example output, each line represents a LAPB frame entering or exiting the gateway server. 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 send and receive sequence numbers. 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.

X.25 Level 3 (Packet Level)

X.25 Level 3 (the packet level), corresponds to Level 3 (the network level) of the OSI reference model. This protocol specifies methods for the delivery of user data, flow-control handling, out-of-band signaling, and addressing. Note that network addresses are also called *X.121 addresses* in accordance with the CCITT X.121 Recommendation. In addition to an Internet address, a gateway server using X.25 must have an X.121 address.

X.25 Level 3 can transport protocol data as well as user data. User data is unstructured, whereas protocol data conforms to the protocol format. For user data, the X.121 address specifies the final destination. For protocol data, the encapsulated data specifies the final address.

The DDN X.25 Protocol

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 data in the form of an Internet packet. 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.

To establish DDN X.25 Basic Service, use the **x25** or **x25-dce** keyword with the **encapsulation** interface subcommand.

To establish DDN X.25 Standard Service, use the **ddnx25** or **ddnx25-dce** keyword with the **encapsulation** interface subcommand. This action causes the gateway server 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 gateway server receives an Internet packet with a non-zero 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.

Mapping Addresses

To transport packets using X.25 Level 3, the gateway server must map network-protocol addresses to X.121 addresses and vice versa. For example, Figure 15-1 illustrates hosts A and B that want to communicate via gateways X and Y, which have an X.25 link between them.

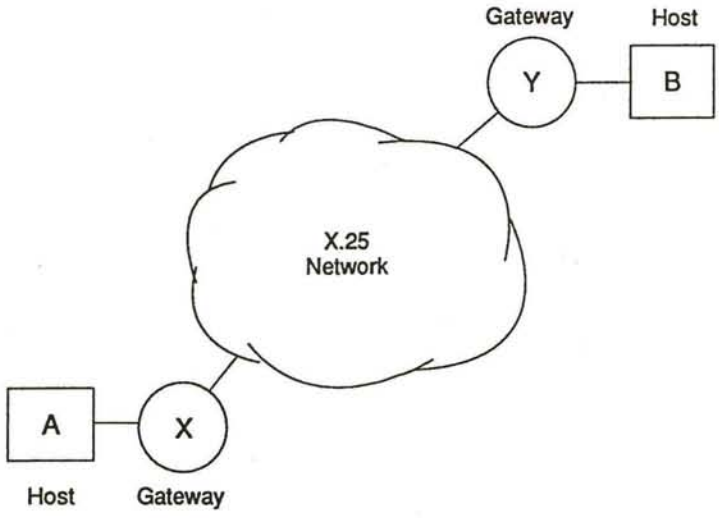


Figure 15-1. Communicating Gateway Servers Through an X.25 Network

To send a packet to host B, host A must first send the packet to gateway X. From the destination address in the packet and from its routing information, gateway X determines that it must send the packet to gateway Y over the X.25 network. Gateway X must then determine the X.121 address for gateway Y to open a virtual circuit. Gateway X uses its network-protocol-to-X.121 address map to convert the Internet address of gateway Y to the equivalent X.121 address. Gateway X can now make the call to create a virtual circuit.

The DDN X.25 Standard 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 15-1. 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.)

Table 15-1. DDN Internet/X.121 Address Conventions

Network Class	Address Format
A	N.H.L.I
B	N.N.H.I
C	N.N.N.HI

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 gateway server can convert Class B and Class C addresses as well. As indicated in Table 15-1, 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.

For more information, see "Defense Data Network X.25 Host Interface Specification", available from the Network Information Center at SRI International in Menlo Park, California.

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 gateway server 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 gateway server then sends broadcast packets to hosts with marked entries. For more information, see "Setting Address Mappings" later in this chapter.

To display the network-protocol-to-X.121 address mapping, use the EXEC command **show x25-map**. See "Displaying Address Mappings" later in this chapter for more information.

Starting X.25 Level 3 Encapsulation

A gateway server using X.25 Level 3 encapsulation can act as a DTE or DCE device on general X.25 networks and on the DDN. To set the operation type, use the **encapsulation** interface subcommand. This subcommand takes one keyword as an argument: **x25** to set DTE operation on a public X.25 network, **ddnx25** to set DTE operation on the DDN X.25 network, **x25-dce** to set DCE operation on a public X.25 network, or **ddnx25-dce** to set DCE operation on the DDN X.25 network. The usual operation type is **x25** or **ddnx25**.

Setting X.25 Level 3 Parameters

After you establish X.25 Level 3 encapsulation, you can set X.25 Level 3 parameters with the **x25** subcommand of the **interface** configuration command. This subcommand usually 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 virtual circuit channel, period of time, byte count, or other quantity, depending on *parameter*. Some **x25** keywords take an additional keyword and/or multiple values. See the CCITT X.25 Recommendation for more information on X.25 parameters.

If you connect a gateway server 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 LAPB Parameters" earlier in this chapter affect X.25 Level 3 operations.

Setting the Interface Address

The gateway server determines the X.121 address of an X.25 interface using the strategy described in "Mapping Addresses" earlier in this chapter. To change the X.121 address for non-DDN networks, use the **x25 address** interface subcommand. This subcommand takes a variable-length X.121 address as its argument.

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. This subcommand takes as arguments a protocol keyword, a protocol address, an X.121 address, and up to six option keywords. The protocol keyword is **ip**, **decnet**, **chaos**, **xns**, or **pup**, and the two address values specify the network-protocol-to-X.121 mapping. The six option keywords add certain features to the mapping specified:

- **reverse** specifies reverse charging for outgoing calls.

- **accept-reverse** causes the gateway server to accept an incoming Reverse Charge call. If this option is not present, the gateway server clears Reverse Charge calls. (Facilities are described in "Setting X.25 Parameters on a Per-Call Basis" later in this section.)

- **broadcast** causes the gateway server to direct any broadcasts to the specified X.121 address.

- **cug** *number* specifies a Closed User Group number (from 1 to 99) for the mapping in the outgoing call. (See "Setting X.25 Parameters on a Per-Call Basis" later in this section.)

- **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.

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.

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): permanent virtual circuits, incoming calls, incoming and outgoing (two-way) calls, and 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 with the upper limit 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 X.25 device must start with the lower limit of the incoming call range and search 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 most 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 15-2. 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.

Table 15-2. Range Limit Keywords for the Virtual Circuit Channel Sequence

Keyword	Limit Type
lic	Lower limit, incoming call range
hic	Upper limit, incoming call range
ltc	Lower limit, two-way call range
htc	Upper limit, two-way call range
loc	Lower limit, outgoing call range
hoc	Upper limit, outgoing call range

The default lower limit for all channel ranges on the gateway server 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.

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. This subcommand takes a virtual circuit channel number, a protocol keyword, and a protocol address as its arguments. The virtual circuit channel number must be less than the lower limit of the incoming call range in the virtual circuit channel sequence (set using the **lic** keyword). The protocol keyword is **ip, decnet, chaos, xns, or pup.** The protocol address is that of the host at the other end of the PVC. To cancel a PVC, use the **no x25 pvc** 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 establish a PVC.

Maintaining Virtual Circuits

The gateway server can clear a switched virtual circuit (SVC) after a set period of inactivity. To set this period, use the **x25 idle** interface subcommand. This subcommand takes a time in minutes as its argument; the default is 0, which causes the gateway server to keep the SVC open indefinitely.

To clear all virtual circuits at once, use the privileged EXEC command **clear x25-vc.** This command takes an interface type keyword (**ethernet, serial,** or **ddn-1822**) and a unit number as arguments to identify the interface with which the virtual circuits are associated.

To clear a particular virtual circuit, the **clear x25-vc** command takes the two arguments described above and a logical circuit number (LCN) value as a third argument to specify the circuit.

To increase throughput across networks, you can establish up to four switched virtual circuits to a host. To specify the maximum number of switched virtual circuits that can be open simultaneously to one host, use the **x25 nvc** interface subcommand. This subcommand takes as its argument a circuit count from 1 to 4; the default is 1.

Configuring the X.25 Level 3 Retransmission Timers

The X.25 Level 3 retransmission timers determine how long the gateway server must wait before retransmitting various Request packets. You can set these timers independently using the **x25** subcommand keywords listed in Table 15-3. 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 15-3. Retransmission Timer Keywords and Defaults

Keyword (DTE/DCE)	Affected Request Packet	Default Time (seconds) (DTE/DCE)
t20 / t10	Restart Request	180 / 60
t21 / t11	Call Request	200 / 180
t22 / t12	Reset Request	180 / 60
t23 / t13	Clear Request	180 / 60

Setting X.25 Packet Sizes

X.25 networks use maximum input and output packet sizes set by the network administration. You can set the gateway server input and output packet sizes to match those of the network with the **x25** subcommand keywords **ips** and **ops**, respectively. Each keyword takes a byte count from 1 to 1024 as its argument; the default for both is 128 bytes. Larger packet sizes are better, because smaller packets require more overhead processing.

Note: Set **ips** and **ops** 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 gateway must break the packet into two or more X.25 packets with the "more data" bit set. The receiving device collects all packets with the "more data" 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. This subcommand takes either 8 or 128 as its argument; the default is 8. 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 **x25** subcommand keywords **win** (for *input window*) and **wout** (for *output window*). Each keyword takes a packet count as its argument. The packet count for **win** and **wout** can range from 1 to the window modulus; the default is 2.

The **win** limit determines how many packets the gateway server can receive before sending an X.25 acknowledgment; the number is one less than **win**. The **wout** limit determines how many sent packets can remain unacknowledged before the gateway server 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 gateway server 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 gateway server loading. To enable this option, use the **x25 th** subcommand with a packet count from 1 to the value of **win** as its argument. The gateway server 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 1, the gateway server can send an acknowledgment per input packet. The default is 0, which disables the option.

Suppressing the Calling Address

To omit the calling address in outgoing calls, use the **x25 suppress-calling-address** interface subcommand. This option is necessary for networks that do not expect the calling address unless it is a subaddress.

Accepting Reverse Charge Calls

To instruct the gateway server to accept all Reverse Charge calls, use the **x25 accept-reverse** interface subcommand.

Forcing Packet-Level Restarts

To force a packet-level restart when the link level is restarted, use the **x25 linkrestart** interface subcommand. This option is necessary for networks that expect this behavior.

Setting X.25 Parameters on a Per-Call Basis

To override interface settings on a per-call basis, use the **x25 facility** interface subcommand. This subcommand takes a keyword and up to two values as arguments. The keyword may be **cug, packetsize, reverse,** or **windowsize.**

The **cug** (Closed User Group) keyword takes as its argument a number from 1 to 99, as agreed between the local system and the X.25 network; this number provides an extra measure of security and is enforced by the network. The **packetsize** keyword takes an input packet size and an output packet size, both in bytes, as arguments. The **reverse**

keyword takes no argument; this keyword requests that all calls be placed such that the remote system pays usage charges. The **windowsize** keyword takes an input window byte count and an output window byte count as arguments. The packet and window sizes correspond to the **ips, ops, win, and wout** parameter keywords described earlier in this chapter.

The gateway server places facility keywords and values in outgoing Call Request packets; the receiving host can accept the new parameters, offer new ones, or simply refuse the call. To disable an **x25 facility** subcommand, enter the same subcommand preceded by the keyword **no.**

The default parameters used for X.25 operation are the network defaults. Facilities parameters take precedence over the network defaults, but may be changed through negotiation with the called system. Facilities specified using the **x25** map subcommand override facilities specified for the network. This approach permits customization of facilities on a per-system basis.

Monitoring X.25 Level 3 Operations

The gateway server provides three **show** commands to provide information on address mappings, interface operation, and virtual circuit operation.

Displaying Address Mappings

To display the network-protocol-to-X.121 address mappings, use the EXEC command **show x25-map.** The following is example output:

```
Serial0: 10.2.0.22 000000220200  DDN, 1 LCN: 4*
Serial0: 10.1.0.1 000000010100  INTERFACE
Serial1: 128.1.0.1 000000010000 INTERFACE
Serial1: 128.1.0.2 000000020000 PERMANENT BROADCAST
Serial1: 128.1.0.3 000000030000 PERMANENT, 2 LCN: 1023*, 1024
```

Each line of output shows the interface name, 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 gateway server network interface), and *DDN* (derived using the DDN address conversion scheme). If broadcasts are enabled for an address mapping, the keyword *BROADCAST* also appears on the output line. Finally, if greater than zero, each line also shows the number of LCNs and a list of the LCNs for the protocol/X.121 address. An asterisk marks the current LCN.

Displaying X.25 Interface Parameters and Statistics

To display parameter information for an interface using X.25 Level 3 protocol, use the EXEC command **show interface.** For X.25 interfaces, the following is example 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 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 gateway server 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 the formats "sent/received" and "successful+failed".

Displaying X.25 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:

```
LCI: 1,  State: D1,  Interface: Serial0
Started  0:55:03,  last input 0:54:56,  output 0:54:56
Connected to [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 if 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 non-zero 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 Operation

To debug X.25 encapsulations, it helps to have a working knowledge of the CCITT X.25 Recommendation.

When installing an X.25 link, you can use the privileged EXEC commands **debug x25** and **debug x25-events** to see the protocol interactions. The **debug x25** command enables the monitoring of all X.25 traffic; the **debug x25-events** also enables the monitoring of all X.25 traffic but does not display information about X.25 data or acknowledgment packets. The following is example output:

```
Serial0: X25 I R1 RESTART (5) 8 lci 0 cause 7 diag 250
Serial0:  X25 O R1 RESTART CONFIRMATION (3) 8 lci 0
Serial0: X25 O P2 CALL REQUEST (19) 8 lci 1
From(14): 31250000000101 To(14): 31109090096101
Facilities: (0)
Serial0: X25 O 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 gateway server 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)*] specifies the number of digits in the address. The *Facilities* field indicates the length (in bytes) of the requested facilities and the facilities contents.

cisco Systems

Chapter 16
Using DECnet Protocol

This section describes the **cisco Systems** implementation of DECnet for the gateway server.

DECnet Restrictions

DECnet support on the gateway server includes local and wide-area DECnet routing over Ethernets and serial lines. The following restrictions apply:

- The gateway server supports only DECnet Phase IV; it does not provide Phase III compatibility.
- The gateway server uses HDLC framing rather than DEC's DDCMP framing for point-to-point lines. You can construct a network using both **cisco Systems** and DEC equipment; however, you must ensure that each point-to-point line has the same type of equipment on both ends.
- The gateway server (and DEC routers) cannot transfer the packets of DEC protocols that do not use DECnet. These protocols include MOP, which supports a remote console and upload/download, and LAT, the DEC terminal server protocol.

Starting the DECnet Process

When you start DECnet processing, the gateway server:

- Starts the DECnet router process, which periodically sends Hello and Routing messages on all DECnet-enabled interfaces. These messages track the availability of hosts on the local and non-local networks.
- Enables the processing of incoming DECnet messages.
- Changes the Ethernet hardware address as required by DECnet. (DECnet does not use a MAC-level address resolution protocol such as ARP or Probe.) The Ethernet hardware address of a host must be AA00.0400.XXYY, where "XXYY" is an encoded version of the DECnet address.

DECnet Addresses

The DECnet protocol associates addresses with machines, not interfaces. Therefore, a gateway server can have only one DECnet address; that is, every DECnet-enabled interface on the gateway server must have the same Ethernet address. (You do not have to set each interface address manually; the **decnet address** command automatically assigns an address to each interface for which you entered a **decnet cost** command. These commands are described later in this chapter.) To provide sufficient routing information, the routing table lists each host and the interface through which the gateway server can access that host.

The Designated Router

The routers on a DECnet dynamically determine which of them is to act as the *designated router*. The designated router is the default router for all end nodes on the Ethernet. When the gateway server starts DECnet operation, it cannot know if it is the designated router until it establishes communications with all other routers on the network. To prevent confusion among the routers, a DECnet gateway server cannot start functioning as the designated router for an interval equal to three times the Hello message interval (which is 15 seconds by default).

Gateway Server Startup Precautions

The **cisco Systems** DECnet implementation permits changing the DECnet address and node type without rebooting (unlike the DEC implementation). If you make one of these changes, the gateway server automatically clears the portion of the routing table that becomes invalid. Most implementations of ARP can discover an address change and distribute the new address. However, to be safe, set the node type before the address, and do not change either after the gateway server starts routing DECnet packets.

If your gateway server configures itself from non-volatile memory, or if it uses BootP to obtain its interface addresses, you can start DECnet operation without taking any special precautions. Otherwise, you must be careful to avoid address and parameter inconsistencies.

If you configure the gateway server from a configuration file loaded over the network, set all DECnet parameters before starting the DECnet router process. This approach prevents parameter inconsistencies.

Starting DECnet operation will change the Ethernet hardware address of the gateway server, preventing communication if other hosts try to use the old Ethernet address. To avoid problems from address inconsistencies, place the DECnet configuration commands before the interface configuration commands in the configuration file. Doing so instructs the gateway server to start DECnet operation before the gateway server passes Ethernet traffic to other hosts on the network. Also, if you load the operating system software for the gateway server from a network host, you must have non-volatile memory to avoid problems arising from the change of address.

Configuring DECnet Operation

The parameters in the **cisco Systems** implementation of DECnet are a subset of the parameters you can alter in DEC's Network Control Protocol (NCP). **cisco Systems** uses the same names, the same range of allowable values, and the same defaults wherever possible. Note that you must use the configuration commands to set DECnet parameters; the **cisco Systems** DECnet implementation does not set parameters by communicating with NCP.

Using the **decnet** Configuration Command

To configure general DECnet operation, use the **decnet** configuration command. This command takes two required arguments, *parameter* and *value*. The argument *parameter* is one of several keywords described below; *value* is an address, a decimal number, or a keyword, depending on *parameter*.

All DECnet parameters except interface cost are global to the gateway server. The **decnet** commands that specify global settings may appear anywhere in a configuration file. To specify interface cost, use the **decnet cost** subcommand of the **interface** configuration command; see "Setting an Interface Cost Value" later in this chapter.

To set the address of the gateway server and start DECnet operation, use the keyword **address**. This keyword takes as its value an address in DECnet format *X.Y*, where *X* is the area number and *Y* is the node number. There is no default gateway server address; you must specify this parameter for DECnet operation.

To set the maximum cost and hop count value for an **area**-type gateway server (see the **node-type** keyword, later in this section), use the keywords **area-max-cost** and **area-max-hops**. These parameters determine the maximum cost and number of hops for a route that the gateway server may consider usable; the gateway server treats as unreachable any route with a cost or number of hops greater than the values you specify. For the keyword **area-max-cost**, the value is a cost from 1 to 1,022; the default is 1,022. For the keyword **area-max-hops**, the value is a hop count from 1 to 30; the default is 30.

To specify how often the gateway server sends Hello messages, use the keyword **hello-timer**. This keyword takes as its value a time from 1 to 8,191 seconds; the default is 15 seconds. The gateway server broadcasts Hello messages on all local networks. Other hosts on the network use the Hello messages to identify the hosts with which they can communicate directly.

To determine the largest node number allowed in the current area, use the keyword **max-address**. This keyword takes as its value a node number from 1 to 1,023; the default is 255. This parameter controls the sizes of internal routing tables and of messages sent to other nodes. In general, all routers on the network should use the same value for this parameter.

To set the largest area number that the gateway server can handle, use the keyword **max-area**. This keyword takes as its value an area number from 1 to 63; the default is 63. Like **max-address**, this parameter controls the sizes of internal routing tables and of messages sent to other nodes. All routers on the network should use the same **max-area** value.

To set the maximum cost and hop count value for a **routing-iv**-type gateway server (see the **node-type** keyword, later in this section), use the keywords **max-cost** and **max-hops**; the gateway server ignores routes that have a cost or hop count greater than the corresponding value of these parameters. For the keyword **max-cost**, the value is a cost from 1 to 1,022; the default is 1,022. For the keyword **max-hops**, the value is a hop count from 1 to 30; the default is 30.

To set the limit on the number of times a packet can pass through a router, use the keyword **max-visits**. This keyword takes as its value a number from 1 to 63; the default value is 63. If a packet exceeds the limit set by this parameter, the gateway server discards the packet. DEC recommends that the value of the **max-visits** parameter be at least twice that of the **max-hops** parameter.

To specify the node type for the gateway server, use the keyword **node-type**. This keyword takes another keyword, **area** or **routing-iv**, as its value. If you specify **area**, the gateway server participates in the DECnet routing protocol with other area routers, as described in the DEC documentation. If you specify **routing-iv** (the default), the gateway server acts as a standard DECnet Phase IV routing node and ignores Level 2 routing packets. The gateway server in this case identifies the closest area router and sends all packets destined for other areas to it.

To set a priority value for use in determining the default router, use the keyword **router-priority**. This keyword takes as its value a number from 0 to 127; the default is 64. The DECnet protocol chooses the router with the highest priority value as the default router. If several routers have the same number, the protocol selects the one with the highest address.

To specify how often the gateway server sends routing messages, use the **routing-timer** keyword. This keyword takes as its value a time from 1 to 65,535 seconds; the default is 40 seconds. Routing messages list all the hosts that the gateway server can reach. Other gateways use this information to construct local routing tables. DEC calls this parameter the "broadcast routing timer", because they have a different timer for serial lines; the **cisco Systems** DECnet implementation does not make this distinction.

Setting an Interface Cost Value

To enable DECnet operation for a gateway server interface, you must set a cost for the interface. DECnet adds this cost to the metrics of routes that use the interface. Most DECnet installations have an individualized routing strategy for using costs. Therefore, check the routing strategy used at your installation to ensure that costs you specify are consistent with those set for other hosts on the network.

To set a cost value for an interface, use the **decnet cost** subcommand of the **interface** configuration command. This subcommand takes one required argument, *cost-value*, which is an integer from 1 to 25. There is no default cost for an interface. The cost you specify applies to the interface identified in the **interface** command (see Chapter 7, "Configuring the Network Interfaces").

Disabling and Re-enabling DECnet

To disable all DECnet processing, use the **no decnet** configuration command. This command does not restore the gateway server address to its previous Ethernet address. When DECnet processing is disabled, the gateway server discards all DECnet packets and does not include DECnet packets in the statistics displayed with the EXEC command **show traffic.**

To restart DECnet processing on the gateway server, use the **decnet address** configuration command.

To disable DECnet processing for a specific interface, use the **no decnet cost** interface subcommand. When DECnet processing is disabled for an interface, the gateway server discards all DECnet packets that interface receives and does not include these DECnet packets in the statistics displayed with the EXEC command **show traffic.** The **no decnet cost** interface subcommand also ends Hello and Routing broadcasts from the interface. However, the gateway server continues to route DECnet packets from other interfaces to the disabled interface; these packets exceed their **max-visits** values and are discarded.

To restart DECnet processing for an interface, use the **decnet cost** interface subcommand.

Monitoring DECnet Operation

To display the current values of parameters set with the **decnet** configuration command, as well as the DECnet routing table, use the EXEC command **show decnet.** Sample output follows.

```
DECnet configuration parameters:
  address: 58.1, node-type: area, max-address: 255, max-area: 63
  hello-timer: 45, routing-timer: 450
  max-cost: 1022, max-hops: 30, area-max-cost: 1022, area-max-hops: 30
  max-visits: 63, router-priority: 64
DECnet interface parameters:
  Ethernet1 cost: 5, we are designated router
  Ethernet2 cost: 5, we are designated router
  Ethernet3 cost: 5, we are designated router
  Ethernet4 cost: 5, we are designated router
```

```
DECnet area table:
  area 3 9 hops, cost 13, via 7.910 Ethernet1
  area 6 7 hops, cost 11, via 7.910 Ethernet1
  area 7 1 hops, cost 5, via 7.910 Ethernet1 prio 64 expires 105 area vec #64
  area 7 1 hops, cost 5, via 7.900 Ethernet1 prio 65 expires 45 area vec #64
  area 8 8 hops, cost 12, via 7.910 Ethernet1
  area 17 9 hops, cost 13, via 7.910 Ethernet1
  area 46 7 hops, cost 15, via 7.910 Ethernet1
  area 56 6 hops, cost 32, via 7.910 Ethernet1
  area 58 0 hops, cost 0 (LOCAL)
  area 60 1 hops, cost 5, via 60.1 Ethernet2 prio 64 expires 31 area vec #64
  area 61 2 hops, cost 15, via 7.900 Ethernet1
  area 62 1 hops, cost 5, via 62.1 Ethernet4 prio 126 expires 133 area vec #70
  area 63 1 hops, cost 5, via 63.61 Ethernet3 prio 64 expires 39 area vec #64

DECnet route table:
  area router 0 hops, cost 0, (LOCAL)
  58.1 0 hops, cost 0, (LOCAL)
  58.31 1 hops, cost 5, direct Ethernet1 expires 45
```

To display general DECnet statistics, use the EXEC command **show traffic**. Sample output of the DECnet-specific portion of **show traffic** follows.

```
DECnet statistics:
  Total: 3190088 received, 0 format errors, 0 unimplemented
         0 not a gateway, 0 no memory, 14 no routing vector
  Hellos: 2109401 received, 0 bad, 1500739 other area, 213712 sent
  Level 1 routing: 0 received, 0 bad, 438743 other area, 178100 sent
  Level 2 routing: 228482 received, 0 not primary router, 178100 sent
  Data: 413462 received, 0 not long format, 0 too many visits
        413308 forwarded, 154 no route, 0 encapsulation failed
```

The command output includes the following counts:

- *unimplemented*: a count of unimplemented control messages. A non-zero count indicates that your DECnet installation uses some messages other than Hello and Routing, or that it uses some option bits not used in the **cisco Systems** implementation. Please report a non-zero count to **cisco Systems.**

- *not a gateway*: a count of packets received before the gateway server began DECnet processing. This count is normally non-zero.

- *no memory*: a count of the times the gateway server did not have sufficient memory to perform an operation. Please report a non-zero count to **cisco Systems.**

- *no routing vector*: a count of the times the gateway server received a routing update packet from a host before the first Hello message from that host. This count is normally non-zero.

- *other area*: a count of Hello messages and Level 1 routing updates sent by hosts in a different area. Only area routers process Hello messages and Level 1 routing updates from other areas. This count is normally non-zero for a gateway server on a network that has more than one area.

- *not long format*: a count of data packets received with addresses in the DECnet short format. All packets on an Ethernet use the long address format; please report a non-zero count to **cisco Systems.**

- *no route*: a count of packets addressed to the gateway server itself and to hosts in a non-local area when no area router is in operation.

To enable logging of DECnet routing activity, use the privileged EXEC command **debug decnet.**

cisco Systems

Chapter 17
Using Other Protocols

The **cisco Systems** gateway server supports three network protocols in addition to TCP/IP, the DDN protocols, and DECnet. This chapter describes the three protocols: Chaosnet, XNS, and PUP.

Using the Chaosnet Protocol

Chaosnet is a local area network protocol developed at the Massachusetts Institute of Technology in the mid-1970s. Several Artificial Intelligence workstation manufacturers use the Chaosnet protocol in their networking software products.

The **cisco Systems** gateway server supports full Chaosnet routing and a small subset of Chaosnet host functions, including the status, uptime, and dump-routing-table services. The gateway server can route Chaosnet packets over Ethernets and synchronous serial lines.

Chaosnet Addressing

Chaosnet addresses are 16-bit quantities written as octal numbers. The higher-order eight bits of the address are the Chaosnet network number and the lower-order eight bits are the host number.

The **cisco Systems** Chaosnet implementation assumes that the Chaosnet network corresponds one-to-one with a subnetted Internet network. For example, Chaosnet subnet 1 must correspond to Internet subnet 1 and Chaosnet host 360 (octal) must correspond to Internet host 240 (decimal). Because a Chaosnet network comprised of **cisco Systems** gateway servers assumes a single underlying Internet network, the gateway server does not route Chaosnet packets from one Internet network to another.

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

Chaosnet Routing

To start the Chaosnet router process, use the **router chaos** configuration command (see Chapter 12, "Routing With the Gateway Server", for more information about **router**). The router process routes Chaosnet packets and sends Chaosnet routing updates. For example, the following commands start Chaosnet routing on network 128.88.0.0:

```
router chaos
network 128.88.0.0
```

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

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

```
redistribute rip
```

See Chapter 12 for more information on protocol-independent routing issues such as administrative distance and redistribution.

Monitoring Chaosnet Operation

To display routing entries obtained from the Chaosnet routing protocol, use the EXEC command **show route**. In the command output, Chaosnet entries are marked by "X" in the first column.

To display Chaosnet-specific ARP entries as 16-bit octal addresses, use the EXEC command **show arp**.

To display statistics on Chaosnet protocol operation, use the EXEC command **show traffic**.

To enable logging of Chaosnet routing activity including service requests, use the privileged EXEC command **debug chaos**.

To enable logging of Chaosnet packet transactions, use the privileged EXEC command **debug chaos-packet**.

Using the XNS Protocol

The gateway server provides a subset of XNS (Xerox Network Service) protocol support (IDP, Echo, and XNS RIP). The gateway server implementation of XNS can coexist with the other protocols that the gateway server supports.

XNS can be routed across 10-megabits/second Ethernet, 3-megabits/second Ethernet, and synchronous serial lines running HDLC, LAPB, or X.25. The **cisco Systems** XNS implementation uses the XNS RIP protocol to maintain routing information across an XNS network. This implementation also provides the XNS Echo protocol through a user interface for debugging purposes and as a server for remote diagnostics.

To initialize XNS support, use the **xns address** configuration command. This command takes one optional argument, *ethernet-address*, which specifies an address to use as the XNS system address. If you do not specify *ethernet-address*, the gateway server uses the first 10-megabits/second Ethernet interface address available, if any. The **no xns address** command turns off XNS support.

To specify the XNS network to which an interface is connected, use the **xns network** subcommand of the interface configuration command. This subcommand takes an XNS network number as its argument. XNS network numbers must be unique across all interfaces.

To display each interface that supports XNS and the XNS RIP routing table, use the EXEC command **show xns**. This command takes one optional argument, *number*, which is an XNS network number. If you specify a *number* value, **show xns** displays information for that network only.

To display XNS protocol statistics, use the EXEC command **show traffic**.

To send XNS Echo packets, use the privileged EXEC command **ping** and specify the keyword **xns** when prompted for a protocol. The gateway server also prompts you for an XNS protocol address; enter an address to begin the exchange.

Using the PUP Protocol

PUP (PARC Universal Protocol) was developed at Xerox's Palo Alto Research Center in the mid-1970s. PUP originally ran on the experimental 3-megabits/second Ethernet, the precursor to the IEEE 802.3 and Ethernet standards. The protocol is still used today by some Xerox workstations.

cisco Systems gateway servers support full PUP routing and a minimal set of PUP host functions, primarily the PUP Echo server. PUP packets may be routed over 10-megabits/second Ethernet, 3-megabits/second Ethernet, and synchronous serial lines.

PUP Addressing

PUP addresses are 16-bit quantities written as two octal numbers separated by a pound sign (for example, 10#371). The high eight bits of the address constitute the PUP subnet number, and the low eight bits constitute the PUP host number.

The **cisco Systems** PUP implementation assumes that the PUP network corresponds one-to-one with a single subnetted Internet network. For example, PUP subnet 1 must correspond to Internet subnet 1 and PUP host 360 (octal) must correspond to Internet host 240 (decimal). Because a PUP network comprised of **cisco Systems** gateway servers assumes a single underlying Internet network, the gateway server does not route PUP packets from one Internet network to another.

The gateway server forms PUP addresses by combining the low eight or fewer bits of the Internet subnet field with the low eight or fewer bits of the Internet host field. This scheme does not assume any particular class of Internet addresses or subnetting scheme. However, **cisco Systems** recommends that you use at least eight bits of subnet number and eight bits of host number.

PUP Routing

When you start the PUP router process, the gateway server begins routing PUP packets and sending out PUP routing updates. The name of the PUP routing protocol is "gwinfo". To start the routing process, use the **router gwinfo** configuration command (see Chapter 12, "Routing With the Gateway Server", for more information about **router**). For example, to enable PUP routing on network 128.88.0.0, the commands are:

```
router gwinfo
network 128.88.0.0
```

PUP routing does not replace Internet routing; an Internet routing protocol, such as RIP, must run concurrently with PUP on the gateway server. To ensure routing table consistency, the Internet routing protocol must have a greater administrative distance than the PUP routing protocol. Also, the PUP router must be configured to re-advertise subnet routes derived from the Internet routing protocol.

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

```
redistribute rip
```

See Chapter 12 for more information on protocol-independent routing issues such as administrative distance and redistribution.

Monitoring PUP Operation

To display routing entries obtained from the PUP routing protocol, use the EXEC command **show route**. In the command output, these entries are preceded by "G".

To display PUP-specific ARP entries as two 8-bit octal numbers separated by a pound sign, use the EXEC command **show arp**.

To display statistical summaries of PUP packets when PUP routing is enabled, use the EXEC command **show traffic**.

To enable logging of PUP routing activity to the console terminal, use the EXEC privileged command **debug pup-packet.**

To enable logging of PUP gwinfo routing exchanges to the console terminal, use the privileged EXEC command **debug gwinfo.**

To send PUP Echo packets, use the privileged EXEC command **ping** and specify the keyword **pup** when prompted for a protocol. The gateway server also prompts you for a PUP protocol address; enter an address to begin the exchange.

cisco Systems

Chapter 18
Network Monitoring and Troubleshooting

This chapter describes how to monitor and troubleshoot your network and gateway servers.

Monitoring and Troubleshooting

This section describes the **show, debug,** and **logging** commands for monitoring the gateway server and network events.

Displaying Information With the **show** Command

The EXEC command **show** displays data structures, configuration parameters, and usage statistics for the gateway server. Descriptions of all the various **show** command options follow.

show ?

The **show ?** command lists all the **show** command options.

show access-lists

The **show access-lists** command displays the contents of the access-control lists.

show arp

The **show arp** command displays the ARP cache. The following example shows the command output:

```
Address         Idle (min)   Hardware Addr    Type    Interface
192.31.7.20              -   0260.8CEE.0134   ARPA    Ethernet #0
192.31.7.17              0   2424.2408.0065   ISO1    Ethernet #0
192.31.7.19              0   0800.0900.46FA   ARPA    Ethernet #0
```

Most of the output is self-explanatory. The *Idle* column shows the number of minutes since the gateway server last received an ARP Reply packet from the host. The *Type* column identifies the encapsulation method; ARPA is the standard Ethernet encapsulation, and ISO1 is the IEEE 802.3 encapsulation.

show bridge

The **show bridge** command displays spanning-tree parameters and bridging tables.

show buffers

The **show buffers** command displays statistics for the buffer pools on the gateway server. The gateway server has one pool of queuing "elements" and four pools of packet buffers of different sizes. For each pool, the gateway 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. The following example shows the command output:

```
Buffer elements:
     250 in free list (250 max allowed)
     688904 hits, 0 misses, 0 created

Small buffers, 80 bytes (50 total):
     49 in free list (150 max allowed)
     1361382 hits, 30 misses, 1 trims, 1 created

Middle buffers, 576 bytes (total 25):
     24 in free list (75 max allowed)
     49996 hits, 2 misses, 1 trims, 1 created

Big buffers, 1500 bytes (total 10):
     10 in free list (30 max allowed)
     18049 hits, 0 misses, 0 trims, 0 created

Large buffers, 8192 bytes (total 4):
     4 in free list (10 max allowed)
     2592 hits, 2 misses, 1 trims, 5 created

34 failures (0 no memory)
```

In the command output, *hits* are successful buffer allocations and *misses* are failures caused by a buffer not being in the pool. The gateway server may react to a miss by creating a new buffer to satisfy a request. A large number of misses may indicate surges of broadcast activity.

The command output also shows the number of buffers created *(created)* and destroyed *(trims)*. *Failures* is the total number of unserviced buffer requests.

show configuration

The privileged **show configuration** command displays the contents of the non-volatile memory, if present and valid. The non-volatile memory stores the configuration information in the gateway server in text form as configuration commands. See "Saving, Examining, and Moving Configuration Information" in Chapter 6, "Advanced Gateway Server Configuration".

show controller {serial | mci | ether}

The **show controller** command displays current internal status information for the CSC-S or CSC-T serial interfaces, Multi-port Communications Interface, or Type 2 Ethernet interface.

show debugging

The **show debugging** command displays the current settings of the **debug** and **undebug** command options.

show decnet

The **show decnet** command displays DECnet configuration information and the DECnet routing table. See Chapter 16, "Using DECnet Protocol", for more information.

show egp

The **show egp** command displays a summary status of the EGP neighbors of the gateway server.

show hardware

The **show hardware** command displays the configuration of the system hardware, the software version strings, the names and sources of configuration files and/or boot images, and the Internet addresses of the interfaces. The following example shows the command output:

```
System Bootstrap, version 3.0(1), copyright (c) 1987, cisco Systems, Inc.
System Software, Version 6.1(200), compiled Mon 06-Sep-87 17:56
Copyright (c) 1987, cisco Systems, Inc.
Chaff uptime is 2 days, 6 hours, 47 minutes
Software booted from 192.31.7.19
Host config file is "chaff-confg", booted from 192.31.7.19
Network config file is "network-confg", booted from 192.31.7.19
Local IP address(es) 192.13.7.18, 15.0.240.2

CSC2 (68020) processor with 1024K bytes of processor memory.
1 Ethernet/IEEE 802.3 interface.
1 Serial network interface.
32K bytes of non-volatile configuration memory.
Configuration register is 0x304
```

show hosts

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

```
Default domain is CISCO.COM
Name/address lookup uses IP name service
Name servers are 255.255.255.255

Host      Flags        Age    Port      Address(es)
FUZZ      (temp,??)    21       -       192.31.7.65
MATHOM    (temp,OK)     0       -       192.31.7.17
CHAFF     (perm,OK)     9       -       192.31.7.18
FRED      (perm,OK)    72       -       192.31.7.20
NINE      (perm,OK)    94       -       192.1.1.129
```

In the command output, a *temp* entry in the *Flags* field is entered by a name server; the gateway server may remove the entry after 72 hours of inactivity. A *perm* entry is entered by a configuration command and is not timed out. Entries marked *OK* are believed to be valid; entries marked *??* are considered suspect and subject to revalidation.

The *Age* column indicates the number of hours since the gateway server last referred to the cache entry. The *Port* column identifies the TCP port specified with the **host** configuration command; the default port is 23, the Telnet port. The *Address(es)* column shows the address of the host. One host may have up to eight addresses.

show imp-hosts

The **show imp-hosts** command displays information about the hosts with which the gateway server has communicated via its CSC-A (1822-LH/DH) interface or serial interface using HDH (1822-J) encapsulation during the past five minutes. See Chapter 14, "Using DDN Protocols", for more information.

show interface [*type unit*]

The **show interface** command displays statistics for the network interfaces on the gateway server. Specify the arguments *type* and *unit* to display statistics for a particular network interface. The argument *type* can be **ethernet, serial,** or **ddn-1822.** The argument *unit* specifies the *n*th interface of type *type*. The following example shows the command output:

```
Ethernet 0 is up, line protocol is up, hardware address 0260.8C00.3319
     Internet address is 192.31.7.18, subnet mask is 255.255.255.240
     Broadcast address is 255.255.255.255
     MTU 1500 bytes, bandwidth 10000 Kbit, delay 1000 usec, rely. 255/255
     Encapsulation is ARPA, no access checking
     Address determined by configuration file
     Last input 0:00:00, output 0:00:00, output hang never
     Output queue length is 0 (maximum 100)
     Five minute input rate 2505 bits/sec, 7 packets/sec
     Five minute output rate 2593 bits/sec, 9 packets/sec
          21225 packets input, 2515593 bytes, 32 no input buffers
          Received 2 broadcasts, 0 runts, 0 giants
          0 input errors (0 CRC, 0 frame, 0 overrun, 0 abort)
          22980 packets output, 1399316 bytes, 0 congestion drops
          0 output errors (0 collisions), 0 interface resets

Serial 0 is up, line protocol is up
     Internet address is 192.31.7.34, subnet mask is 255.255.255.240
     Broadcast address is 255.255.255.255, helper address is 192.31.7.31
     MTU 1500 bytes, bandwidth 20 Kbit, delay 20000 usec, rely. 255/255
     Encapsulation is HDLC, no access checking
     Address determined by configuration file
     Last input 0:00:13, output 0:00:14, output hang never
     Output queue length is 0 (maximum 100)
     Five minute input rate 1 bits/sec, 0 packets/sec
     Five minute output rate 44 bits/sec, 0 packets/sec
          3326 packets input, 79750 bytes, 0 no input buffers
          Received 113 broadcasts, 0 runts, 0 giants
          0 input errors (0 CRC, 0 frame, 0 overrun, 0 abort)
          1585 packets output, 108330 bytes, 0 congestion drops
          0 output errors (0 collisions), 0 interface resets
```

show ip-cache

The **show ip-cache** command displays the routing table cache that the Multi-port Communications Interface uses to rapidly switch Internet packets between its interfaces.

show line [*line-number*]

The **show line** command displays a summary status of the console and virtual terminal lines on the gateway server. Specify the argument *line-number* (octal) to display detailed information about a particular line. The following example shows the command output:

```
Tty  Type   Tx/Rx  A  Modem  Roty  AccO  AccI  Uses  Noise
  0  CTY           -     -     -     -     -    34      8
* 1  VTY           -     -     -     -     -   153      0
  2  VTY           -     -     -     -     -    12      0
  3  VTY           -     -     -     -     -     0      0
  4  VTY           -     -     -     -     -     0      0
  5  VTY           -     -     -     -     -     0      0
```

In the command output example, the *Tty* column lists the line number; an asterisk indicates an active line. The *Type* column identifies the line type; CTY is the console terminal and VTY is a virtual terminal. The *Tx/Rx*, *A*, *Modem*, and *Roty* columns do not apply to the gateway server.

The *AccO* and *AccI* columns indicate the access classes for incoming (virtual terminal) and outgoing (such as Telnet) connections, respectively. The *Uses* column shows the total number of connections made to or from the terminal line since the system was booted. This count helps you evaluate terminal line use. The *Noise* column lists the total number of "noise" characters received. (A noise character is a non-activating character received as a framing error or when the line is inactive.)

show logging

The **show logging** 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 Monitoring Protocol) configuration parameters and protocol activity. See "Capturing Troubleshooting Output With the **logging** Command" later in this chapter for more information.

show memory

The **show memory** 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. The following example shows the summary command output:

```
             Head   Free Base  Total Bytes  Used Bytes  Free Bytes
Processor   5435C       67978       703652      149016      554636
Multibus   100000     1005E8        16384        3024       13360
```

show processes

The **show processes** command displays information about all active processes, including:

- Process ID
- Queue type (high, medium, low)
- Scheduler test (event, time, suspended)
- Total runtime (in milliseconds)
- Count of invocations
- Microseconds per invocation
- Stack utilization
- Controlling terminal
- Process name

The following example shows the command output:

```
PID Q T     PC Runtime (ms)  Invoked   uSecs   Stacks TTY Process
  1 M E   E5BA       759080   213980    3547  249/400   0 Net Background
  2 M E   3994         2780       57   48771  197/400   0 Logger
  3 L T    3E8        32416  2033195      15  336/600   0 TTY Background
  4 M E  15978           36       47     765  306/400   0 BootP Server
  5 H E   3994      5729664  3056835    1000  527/900   0 IP Input
  6 M E  139CC     17954296   868666   20000   29/500   0 IP Protocols
  7 M E  1EBFC        13328   415263      32  235/500   0 TCP Timer
  8 L E  1F466        55688      253  220110  263/500   0 TCP Protocols
  9 L E   F3E4      3749376  1046729    3000  278/400   0 ARP Input
 10 L E   3994            0        1       0  342/400   0 Probe Input
 11 M T    1F4     12305792  4048756    3000   63/500   0 EGP Router
 12 M E  25B06      1120356    48041   23000  289/500   0 IGRP Router
```

Because the gateway server has a 4-millisecond clock resolution, consider the runtimes reliable only after a large number of invocations or a reasonable measured runtime.

show protocols

The **show protocols** command shows the status of any routing processes, including other gateways with which the gateway server is communicating. See Chapter 12, "Routing With the Gateway Server", for more information.

show route [*network*]

The **show route** command displays the gateway server routing table. Specify the argument *network* to display routing information for a particular network, where *network* is a network or subnet address.

The **show route** command can provide valuable diagnostic clues for resolving routing problems. See "Probing Nodes With the **ping** Command" later in this chapter for more information.

show sockets

The **show sockets** command displays the generic socket data structures that the gateway server maintains for demultiplexing Internet protocols such as UDP and IGRP.

show stacks

The **show stacks** command monitors the stack utilization of processes and interrupt routines.

show tcp [*line-number*]

The **show tcp** command displays the status of all TCP connections. Specify the argument *line-number* (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
```

show terminal

The **show terminal** command displays the configuration parameter settings for the current terminal.

show traffic

The **show traffic** command displays a detailed breakdown of the protocol traffic through the system. The following example shows the command output:

```
IP statistics:
     Rcvd:  23151 total, 0 format errors, 0 checksum errors, 0 bad hop count
            0 unknown protocol, 20562 local destination, 0 not a gateway
     Frags: 0 reassembled, 0 timeouts, 0 fragmented, 0 couldn't fragment
     Bcast: 197 received, 1391 sent
     Sent:  22431 generated, 2589 forwarded
            1 encapsulation failed, 0 no route

ICMP statistics:
     Rcvd:  0 checksum errors, 0 redirects, 451 unreachable, 271 echo
            104 echo reply, 0 mask requests, 0 mask replies, 0 other
            0 timestamp, 0 info request
     Sent:  0 redirects, 0 unreachable, 127 echo, 271 echo reply
            0 mask requests, 0 mask replies, 0 quench, 0 timestamp
            0 info reply, 0 time exceeded, 0 parameter problem

UDP statistics:
     Rcvd:  16 total, 0 checksum errors, 0 no port
     Sent:  1045 total, 0 forwarded broadcasts

TCP statistics:
     Rcvd:  19542 total, 0 checksum errors, 4 no port
     Sent:  20917 total

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

IGRP statistics:
     Rcvd:  376 total, 0 checksum errors
     Sent:  356 total

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

ARP statistics:
     Rcvd:  2 requests, 2 replies, 0 revers, 0 other
     Sent:  2 requests, 1 replies (0 proxy), 0 reverse

Probe statistics:
     Rcvd:  1 address requests, 1 address replies, 17 other
     Sent:  2 address requests, 0 address replies (0 proxy)

Unknown packet types rcvd: 0
```

In the command output, most of the categories are self-explanatory; a few are not. A *format* error is a gross error in the packet format, such as an impossible Internet header length. A *bad hop count* occurs when a packet is discarded because its time-to-live (TTL) field was decremented to zero. An *encapsulation failure* usually indicates that the gateway server received no reply to an ARP request and therefore did not send a datagram. A *no route* occurrence is counted when the gateway server discards a datagram it did not know how to route. A *proxy reply* is counted when the gateway server sends an ARP or Probe Reply on behalf of another host.

show users [all]
systat [all]

The **show users** or **systat** command displays information about the active lines of the gateway server, including the line number, connection names, and terminal location. Specify the keyword **all** to display information for both active and inactive lines. The commands enable you to monitor virtual terminal use. The following example shows the command output:

```
 TTY      Host(s)              Idle   Location
con0      CLUTTER                 0   Console terminal
          DROSS                1:35
*vty1     idle                    6   MATHOM.CISCO.COM
```

show x25-map

The **show x25-map** command shows the network-protocol-to-X.121 address mapping table for gateway servers equipped with the commercial or DDN Standard X.25 software option. See Chapter 15, "Using X.25 Protocols", for more information.

show x25-vc [*lcn*]

The **show x25-vc** command displays X.25 circuit information on a gateway server equipped with the commercial or DDN Standard X.25 software option. Specify the argument *lcn*, a logical connection number, to display information for a particular virtual circuit. See Chapter 15, "Using X.25 Protocols", for more information.

show xns

The **show xns** command displays XNS configuration parameters and protocol activity.

Troubleshooting With the **debug/undebug** Commands

The gateway server includes hardware and software to aid in tracking down problems with the gateway server or with other hosts on the network. The privileged EXEC command **debug** enables you to display 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.

You may also send **debug** output to the log file of a UNIX *syslog* server or to a gateway server virtual terminal. See "Capturing Troubleshooting Output With the **logging** Command" later in this chapter for more information.

Descriptions of all the various **debug** command options follow. Each option has a corresponding **undebug** command to disable console logging for that option.

debug ?

The **debug ?** command lists and briefly describes all the **debug** command options.

debug all

The **debug all** command enables all diagnostic output. The **undebug all** command, which turns off all diagnostic output, may be more useful. You can then enable only the diagnostic output you need.

debug arp

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

debug bridge

The **debug bridge** command enables logging of spanning-tree algorithm execution during bridging operation.

debug broadcast

The **debug broadcast** command enables logging of all broadcast packets received. This information is useful for finding the source of a broadcast storm.

debug chaos

The **debug chaos** command enables logging of Chaosnet routing transactions.

debug chaos-packet

The **debug chaos-packet** command enables logging of Chaosnet packet generation, reception, and forwarding and routing transactions.

debug decnet

The **debug decnet** command enables logging of DECnet routing activity.

debug egp

The **debug egp** command enables logging of all EGP transactions.

debug egp-events

The **debug egp-events** command enables logging of major EGP transactions, such as an EGP peer being acquired or discontinued.

debug gwinfo

The **debug gwinfo** command enables logging of PUP gwinfo routing transactions.

debug hdh

The **debug hdh** command enables logging of HDH (HDLC Distant Host) transactions.

debug hello

The **debug hello** command enables logging of HELLO routing transactions.

debug icmp-packet

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

debug igrp

The **debug igrp** command enables logging of IGRP routing transactions.

debug ip-filter *list*

The **debug ip-filter** command enables logging of permit and deny actions taken in accordance with the access list *list*.

debug ip-packet

The **debug ip-packet** command enables logging of general IP debugging information, including packets received, generated, and forwarded.

debug lapb

The **debug lapb** command enables logging of LAPB (X.25 Level 2) transactions. See Chapter 15, "Using X.25 Protocols", for more information.

debug packet

The **debug packet** command enables logging of packets that the gateway server is unable to classify. Examples are packets with an unknown Ethernet link type or IP packets with an unrecognized protocol field.

debug packet-events

The **debug packet-events** command enables logging of protocol events such as certain ICMP messages and gateway-related service requests.

debug psn

The **debug psn** command enables logging of noteworthy PSN events for DDN gateways equipped with CSC-A (1822-LH/DH) interfaces or serial interfaces using HDH (1822-J) encapsulation.

debug pup-packet

The **debug pup-packet** command enables logging of PUP transactions.

debug rip

The **debug rip** command enables logging of RIP routing transactions.

debug routing

The **debug routing** command enables logging of routing table events such as network appearances and disappearances. See "Probing Nodes With the **ping** Command" later in this chapter for more information on resolving routing problems.

debug serial-interface

The **debug serial-interface** command enables logging of serial-interface hardware events for gateway servers equipped with serial network interfaces.

debug tcp

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

debug udp-packet

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

debug x25

The **debug x25** command enables logging of all X.25 circuit activity. See Chapter 15, "Using X.25 Protocols", for more information.

debug x25-events

The **debug x25-events** command enables logging of significant X.25 events such as Call Request and Call Clear messages. Unlike the **debug x25** command, the **debug x25-events** command does not report data or acknowledgment activity, thus reducing the volume of the output. See Chapter 15, "Using X.25 Protocols", for more information.

debug xns-packet

The **debug xns-packet** command enables logging of XNS packet transactions.

debug xns-rip

The **debug xns-rip** command enables logging of XNS RIP routing transactions.

Capturing Troubleshooting Output With the **logging** Command

By default, the gateway server sends **debug** command output and system bug messages to the console terminal. The **logging** configuration command lets you redirect these messages to one or more *syslog* servers on cooperating hosts; the *syslog* format is compatible with 4.3 BSD UNIX. The EXEC command **terminal monitor** causes the current terminal to receive logging output as well.

To identify a syslog server host, use the **logging** command with the argument *address*, which is the address of a syslog server host. You can use this form of the **logging** command more than once to direct logging messages to multiple syslog server hosts. The **no logging** command ends the sending of logging messages to the server host at *address*.

To define the syslog level of the messages, use the **logging** command with a keyword and the optional argument *level*. The keyword **console** limits the messages normally displayed on the console terminal to those with a level at or below *level*; the default level is **informational**. The keyword **trap** controls system bug messages, interface status messages, and other non-debug messages; the default level is **warning**. The argument *level* is one of the following keywords, listed in order from highest to lowest level: **emergencies, alerts, critical, errors, warnings, notifications, informational,** or **debugging**. The **no logging console** or **no logging trap** command inhibits the display of messages with the indicated level.

The EXEC command **show logging** displays the addresses, terminals, and levels associated with the current logging setup.

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/gwlog
```

The **local7** keyword specifies the logging facility to be used by the gateway server. The **debug** keyword specifies the syslog *level* for the gateway server. 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.

Using SNMP

The Simple Network Monitor Protocol (SNMP) provides a means to access and set configuration and runtime parameters for the **cisco Systems** gateway and terminal servers. The **cisco Systems** implementation of SNMP is in accordance with RFC 1028 and supports IDEA 11. (IDEA 11 is not yet a standard; the gateway server supports IDEA 11 for future compatibility.)

By default, SNMP is disabled and neither responds to nor generates any network traffic. To enable SNMP, use the **snmp-server session** configuration command. This command takes one optional argument, *string*, which is a session string. If you specify a session string, the gateway server restricts GET and SET operations to queries with that session string. If you do not specify a session string, the gateway server permits all GET operations and denies all SET operations. To disable SNMP, use the **no snmp-server** command.

To set up an access list to determine which hosts can send requests, use the **snmp-server access-list** configuration command. This command takes an access-list number as its argument. The gateway server discards packets from hosts that the access list denies. To disable access-list operation, use the **no snmp-server access-list** command with the appropriate access-list number.

To set the size of packets that SNMP accepts to a value greater than 484 bytes, use the **snmp-server packet-size** configuration command. This command takes a byte count as its argument; the permitted range for this byte count is that of the system.

To specify the host or hosts that should receive TRAP messages, use the **snmp-server host** configuration command. This command takes a host name as its argument and can be used more than once to specify multiple hosts. To specify that a host should not receive TRAP messages, use the **no snmp-server host** command.

The SNMP specification requires that a TRAP message be generated for each packet with an incorrect session string. However, because this action can result in a security breach, the gateway server by default does not return a TRAP message when it receives an incorrect session string. To override this default, use the **snmp-server trap-authentication** configuration command. To re-enable the default action, use the **no snmp-server trap-authentication** command.

Probing Nodes With the ping Command

The privileged EXEC command **ping** is a basic Internet diagnostic tool. The command sends an ICMP Echo Request datagram to an Internet address and waits for an ICMP Echo Reply message. Most Internet implementations support the ICMP Echo function.

Results from the **ping** command help you evaluate the path-to-host reliability, delays over the path, and whether the host can be reached or is functioning. The **ping** command also helps you determine if routing is functioning properly. Common routing problems include:

- "Open jaw" or "diode" routes (the gateway server can reach a destination, but the destination cannot route back).
- Incorrectly established node or gateway addresses.
- A network interface disconnected but not shut down to disable it as a viable route.

When you enter the **ping** command, the system prompts you for a protocol keyword (**ip**, **pup**, or **xns**), the Internet address or host name, repeat count (default is 5), datagram size (default is 100 bytes), timeout interval (default is 2 seconds), and extended commands (default is none). The following example shows **ping** command output:

```
Protocol [ip]:<RETURN>
Target IP address: 192.31.7.17
Repeat count [5]: 50
Datagram size [100]: 500
Timeout in seconds [2]: <RETURN>
Extended commands [n]: <RETURN>
Type CTRL/^X to abort.
Sending 50, 500-byte ICMP Echos to 192.31.7.17, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent, round-trip min/avg/max = 16/22/36 ms
```

Each exclamation point (!) indicates receipt of an ICMP Echo Reply message. A period (.) indicates the gateway server timed out while waiting for a reply. A *U* indicates receipt of an ICMP Host Unreachable or Network Unreachable message. A *Q* indicates receipt of an ICMP Source Quench message. The output concludes with the success rate and minimum, average, and maximum round-trip times.

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

Diagnosing Processors

To identify the processor card, look for the card with the flat cable labeled "console". For details about the hardware, see Chapter 8, "Configuring the Gateway Server Processors".

To run the memory diagnostic on your processor:

1. Turn off the gateway server.
2. Remove the jumpers from the boot file number field (bits 3, 2, 1, and 0) of the processor configuration register. (Note jumper positions before removal.)
3. Restart the system. Wait for the gateway server to print the two-line banner message and prompt you with a right angle bracket (>).
4. Type **t m** and press RETURN. The gateway server prompts you for additional arguments.
5. Press RETURN to accept each default argument.

Running a single pass of the diagnostic takes only a few minutes. If the program encounters memory problems, it displays appropriate error messages on the console terminal. The following example shows the memory diagnostic.

```
System Bootstrap, Version 3.1(1), copyright (c) 1987, cisco Systems, Inc.
CSC1 processor with 1024 Kbytes of memory
>t m <RETURN>
Memory/Bus Diagnostic
Starting Address [0x1002]? <RETURN>
Ending Address [0x100000]? <RETURN>
Hex argument for variable tests [0xFFFF]? <RETURN>
Select Tests [all]? <RETURN>
Number of passes to run [1000]? <RETURN>
Message Level (0=silence, 1=summary, 2=normal) [2]? <RETURN>

Testing addresses between 0x1002 and 0x100000

Begin pass 0, test 123456789101112  End pass
Begin pass 1, test 123456789101112  End pass
.
.
.
```

Random Access Memory (RAM) on the CSC/1 and CSC/2 processors occupies hexadecimal addresses 00000 through FFFFF. You may be unable to run the memory diagnostic outside this range.

Checking Server Host Configuration

The gateway server normally attempts to load configuration files or boot images from server hosts. Common reasons for failure include:

- No TFTP service is running on the designated server host.
- The desired configuration file does not exist.
- The desired configuration file is not in the directory where the TFTP server expects to find it, or the file protection setting prevents access. See your server system documentation for information on default directories.
- The server host does not recognize the broadcast address that the gateway server used. See Chapter 11, "Understanding Internet Addresses, Broadcasts, and Address Resolution", for information on changing the Internet broadcast address of the gateway server.

Reloading and Crashes

The privileged EXEC command **reload** causes the System Bootstrap program to halt, re-initialize, and reload the system. If the appropriate boot file-name jumpers are set, the system automatically attempts to reload its operating software. If syslog logging is enabled, a message regarding the reload request is logged.

To reload the system, the CSC/1 processor disables its memory refresh and watchdog timers. Part of a spurious parity error message usually appears on the console terminal before the watchdog timer re-initializes the system. The CSC/2 processor simply issues a hardware reset function to reload the system.

An error message on the console terminal announces a system crash. Crash messages include:

```
Address Error
Protection Error
Parity Error
Page Invalid Error
Illegal Instruction
Zero Divide
CHK
TRAPV
Privileged Instruction
Line 1010 Emulator Trap
Line 1111 Emulator Trap
Spurious Bus Error
Undefined Trap
```

Record any information printed with the crash message, such as the error address, then contact **cisco Systems** immediately.

Either software or hardware failures can cause a system crash. To determine whether the hardware is at fault, run the full system memory diagnostics described in "Diagnosing Processors" earlier in this chapter.

Using the System Bootstrap Program

The System Bootstrap program helps initialize the processor hardware and boot the main operating system software. If you remove the jumpers from the boot file number field (bits 3, 2, 1, and 0) of the configuration register, you can start the gateway server in standalone bootstrap mode. The prompt ">" is the prompt of bootstrap mode.

The following bootstrap commands are among the most useful:

- The **h** (help) command prints a summary of the bootstrap commands.
- The **i** (initialize) command causes the bootstrap program to re-initialize the hardware and clear the contents of memory.
- The **b** (boot) command with no argument boots the default software from ROM, assuming no jumpers in the boot field portion of the configuration register. You may include an argument, *filename*, to specify a file to be booted over the network using TFTP. You may also include a second argument, *host*, which is the Internet address or name of a particular server host.
- The **t** (test) command runs various diagnostic programs; see "Diagnosing Processors" earlier in this chapter.

Note: If you start the gateway server with the BREAK disable (bit 8) jumper removed from the configuration register, you can press the BREAK key on the console terminal to force the gateway server into bootstrap mode. Type **c** to continue normal execution of the system software.

cisco Systems

Appendix A
EXEC Command Reference

This appendix summarizes all the EXEC commands. Separate sections list the **show** commands for displaying network status and the **debug** commands for producing debugging information.

Network Management Commands

?
help

Lists available EXEC commands. Type **?** or **help** after a command name to display information about the command.

clear arp

(Privileged) Removes all non-interface entries from the ARP cache.

clear host *name*

(Privileged) Removes a host-name-to-address mapping from the host-name-and address-cache, where *name* is the host name. To clear the entire cache, use an asterisk (*) for *name*.

clear interface *type unit*

(Privileged) Resets or re-initializes a network interface, where *type* is **ethernet, serial,** or **ddn-1822,** and *unit* is the interface unit number.

clear line *line-number*

(Privileged) Ends any connections and associated processes, and resets data structures associated with a line, where *line-number* is an octal line number.

clear route *network*

(Privileged) Removes routing table entries for the specified network, where *network* is an Internet address.

clear x25-vc *interface unit* [*lcn*]

(Privileged) Clears one or all X.25 virtual circuits, where *interface* is **ethernet, serial,** or **ddn-1822** and *unit* is the interface unit number. To clear one virtual circuit, use *lcn* to specify the logical circuit number. To clear all virtual circuits on the interface, omit *lcn*.

configure

(Privileged) Enters configuration command mode. See Appendix B, "Configuration Command Reference", for a summary of commands available in this mode.

{connect|telnet} *host* [*port*] **[debug] [/route:***path***]**

Starts a Telnet connection, where *host* is the name or Internet address of the host and optional *port* is a decimal TCP port number; the default is the default Telnet server port (decimal 23). Optional **debug** enables debugging output of Telnet option negotiations. Optional **/route:** specifies loose source routing, where *path* is a list of host names or Internet addresses that specify network nodes, including the final destination.

disconnect [*identifier*]

Closes a connection, where *identifier* is the connection name or number. The default is the current connection.

enable
disable

Enters privileged command mode, changing the system prompt to the host name followed by a pound sign (#). The privileged **disable** command ends privileged mode operation.

exit
quit

Logs off the gateway server.

name-connection *number*

Prompts for a new name for a connection, where *number* is the connection number.

ping

(Privileged) Sends an ICMP Echo Request message to an Internet address and waits for an ICMP Echo Reply message, testing the reliability of the path to a host, the delays over that path, and whether that host can be reached or is functioning.

reload

(Privileged) Halts and re-initializes the gateway server.

resume [*identifier*]

Resumes a previous connection, where *identifier* is the connection name or number. The default is the most recently used connection.

send *connection*

(Privileged) Sends a message to a connection, where *connection* is the connection number. To send a message to all connected terminals, use an asterisk (*) for *connection.*

terminal escape-character *character*

Sets the escape character for the current terminal line, where *character* is the decimal representation of the ASCII character or the character itself.

terminal length *screen-length*

Sets the screen length of the current terminal, where *screen-length* is the number of lines on the screen. The default is 24; 0 disables pausing between screens of output.

terminal monitor
terminal no monitor

Copies debugging output to the current virtual terminal, enabling remote monitoring of network events at a gateway server. The **terminal no monitor** command ends this copying.

terminal notify
terminal no notify

Sets the current line with multiple connections to inform a user when output is pending on a connection other than the current connection. The **terminal no notify** command ends this notification.

terminal pad *decimal-number count*
terminal no pad *decimal-number*

Sets padding by the current terminal of the specified output character, where *decimal-number* is the decimal representation of the ASCII character, and *count* is the number of NUL bytes sent after that character. The **terminal no pad** command removes padding for the specified output character.

write erase

(Privileged) Clears the contents of the non-volatile memory.

write memory

(Privileged) Copies current configuration information to the non-volatile memory.

write network

(Privileged) Sends a copy of the current configuration information to a server host; prompts for a destination host and a file name.

write terminal

(Privileged) Displays the current configuration information.

show Commands

show ?

Displays the available **show** command options.

show access-lists

Displays the contents of the access-control lists.

show arp

Displays the ARP cache.

show bridge

Displays spanning-tree parameters and bridging tables.

show buffers

Displays statistics for the buffer pools on the gateway server.

show configuration

(Privileged) Displays the contents of the non-volatile memory, if present and valid.

show controller {serial | mci | ether}

Displays internal status information for the CSC-S or CSC-T serial interfaces, Multi-port Communications Interface, or Type 2 Ethernet interface.

show debugging

Displays the current settings of the **debug** and **undebug** command options.

show decnet

Displays DECnet configuration information and the DECnet routing table.

show egp

Displays a summary status of the EGP neighbors of the gateway server.

show hardware

Displays the configuration of the system hardware, the software version strings, the names and sources of configuration files and/or boot images, and the Internet addresses of the interfaces.

show hosts

Displays information relating to name and address translation: default domain name, style of name lookup service, name server addresses, and contents of the host-name-and-address cache.

show imp-hosts

Displays information about the hosts with which the gateway server has communicated via its CSC-A (1822-LH/DH) interface or serial interface using HDH (1822-J) encapsulation during the past five minutes.

show interface [*type unit*]

Displays statistics for network interfaces on the gateway server. To specify a particular interface, use *type* and *unit*, where *type* is **ethernet, serial,** or **ddn-1822,** and *unit* is the interface unit number.

show ip-cache

Displays the routing table cache that the Multi-port Communications Interface uses to rapidly switch Internet packets between its interfaces.

show line [*line-number*]

Displays a summary status of the console and virtual terminal lines on the gateway server. To display detailed information about a particular line, use *line-number* (octal) to specify the line.

show logging

Displays the state of syslog error and event logging, including host addresses and whether console logging is enabled. Also displays SNMP configuration parameters and protocol activity.

show memory

Displays statistics for the memory free pool.

show processes

Displays information about all active processes.

show protocols

Displays the status of routing processes, including other gateways with which the gateway server is communicating.

show route [*network*]

Displays the gateway server routing table. To display routing information for a particular network, use *network* to specify the network or subnet address in dotted-decimal format.

show sockets

Displays the generic socket data structures the gateway server maintains for demultiplexing certain Internet protocols.

show stacks

Monitors the stack utilization of processes and interrupt routines.

show tcp [*line-number*]

Displays the status of all TCP connections. To display the status of the TCP connections for a particular line, use *line-number* (octal) to specify the line.

show terminal

Displays the configuration parameter settings for the current terminal.

show traffic

Displays a detailed breakdown of the protocol traffic through the system.

show users [all]

Displays information about the active lines of the gateway server. To display information about both active and inactive lines, use **all**.

show x25-map

Displays the network-protocol-to-X.121 address mapping table for gateway servers equipped with the X.25 software option.

show x25-vc [*lcn*]

Displays X.25 virtual circuit information on gateway servers equipped with the X.25 software option. To display information for a particular virtual circuit, use *lcn* to specify the logical connection number.

show xns

Displays XNS configuration parameters and protocol activity.

systat [all]

Displays information about the active lines of the gateway server. To display information about both active and inactive lines, use **all**.

where

Displays a list of the current Telnet sessions associated with the (virtual) terminal, including names and numbers.

debug Commands

debug ?

(Privileged) Lists and briefly describes all the **debug** command options. There is a corresponding **undebug** command for each debug option.

debug all

(Privileged) Enables all diagnostic output.

debug arp

(Privileged) Enables logging of ARP and Probe protocol transactions.

debug bridge

(Privileged) Enables logging of spanning-tree algorithm execution during bridge operation.

debug broadcast

(Privileged) Enables logging of all broadcast packets received.

debug chaos

(Privileged) Enables logging of Chaosnet routing transactions.

debug chaos-packet

(Privileged) Enables logging of Chaosnet packet generation, reception, forwarding, and routing transactions.

debug decnet

(Privileged) Enables logging of DECnet routing activity.

debug egp

(Privileged) Enables logging of all EGP transactions.

debug egp-events

(Privileged) Enables logging of major EGP transactions, such as an EGP peer being acquired or discontinued.

debug gwinfo

(Privileged) Enables logging of PUP gwinfo routing transactions.

debug hdh

(Privileged) Enables logging of HDH (HDLC Distant Host) transactions.

debug hello

(Privileged) Enables logging of IP HELLO routing transactions. This command does not affect DECnet HELLO debugging messages.

debug icmp-packet

(Privileged) Enables logging of ICMP transactions only.

debug igrp

(Privileged) Enables logging of IGRP routing transactions.

debug ip-filter *list*

(Privileged) Enables logging of permit and deny actions related to the access list *list*.

debug ip-packet

(Privileged) Enables logging of general IP debugging information, including packets received, generated, and forwarded.

debug lapb

(Privileged) Enables logging of LAPB (X.25 Level 2) transactions.

debug packet

(Privileged) Enables logging of packets that the gateway server is unable to classify. Examples are packets with an unknown Ethernet link type or IP packets with an unrecognized protocol field.

debug packet-events

(Privileged) Enables logging of protocol events such as certain ICMP messages and gateway-related service requests.

debug psn

(Privileged) Enables logging of noteworthy PSN events for DDN gateways equipped with CSC-A (1822-LH/DH) interfaces or serial interfaces using HDH (1822-J) encapsulation.

debug pup-packet

(Privileged) Enables logging of PUP transactions.

debug rip

(Privileged) Enables logging of RIP routing transactions.

debug routing

(Privileged) Enables logging of routing table events.

debug serial-interface

(Privileged) Enables logging of serial-interface hardware events.

debug tcp

(Privileged) Enables logging of significant TCP transactions such as state changes, retransmissions, and duplicate packets.

debug udp-packet

(Privileged) Enables logging of UDP-based transactions.

debug x25

(Privileged) Enables logging of all X.25 circuit activity.

debug x25-events

(Privileged) Enables logging of significant X.25 events, such as Call Request and Call Clear messages.

debug xns-packet

(Privileged) Enables logging of XNS packet transactions.

debug xns-rip

(Privileged) Enables logging of XNS RIP routing transactions.

Appendix B Configuration Command Reference

This appendix summarizes all the commands for configuring the gateway server. Separate sections list the subcommands for the **interface**, **line**, and **router** configuration commands.

Configuration Commands

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

Adds a permit or deny condition to an access list, where *list* is an integer from 1 through 99, **permit** or **deny** specifies the type of condition, *address* is the Internet address to be tested, and *mask* is a mask in Internet format that specifies bits to be ignored in the test. The **no access-list** command removes an access list.

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

Adds a permit or deny condition to an extended access list for filtering traffic through a network interface, where *list* is an integer from 100 through 199; **permit** or **deny** specifies the type of condition; *protocol* is **ip**, **tcp**, **udp**, **icmp**, or **chaos**; *source* is an Internet source address; *source-mask* is a mask in Internet format of source address bits to be ignored; and *destination* and *destination-mask* are the destination Internet address and mask.

For **tcp** and **udp**, optional *operator* is **lt**, **gt**, **eq**, or **neq**, and *operand* is the decimal destination port number. For **chaos**, *operator* is **eq** or **neq**, and *operand* is a Request For Connection packet contact string.

arp *internet-address ethernet-address type*
no arp *internet-address*

Adds a permanent entry to the ARP cache, where *internet-address* is the Internet address corresponding to the hardware address specified by *ethernet-address*, which is a dotted triple of hexadecimal digits (such as "0800.0C00.1F5C"), and *type* is **arpa**, **iso1**, or **iso2**.

[no] autonomous-system *number*

Specifies the autonomous system number for the gateway server, where *number* is a 16-bit decimal number.

banner *c message c*
no banner

Specifies a message that the gateway server displays when an EXEC session begins, where *c* is a delimiting character and *message* is one or more lines of text. You cannot use the delimiting character in the message. The **no banner** command removes the message.

boot {system | network | host} *filename address*

Changes default file names and may specify a server host for netbooting configuration files and boot image files for the gateway servers with the non-volatile memory option. For **system,** *filename* is the file name of the operating software to load and *address* is the Internet address of the network host holding that file. If this command appears in the non-volatile configuration memory, it overrides the processor configuration register settings, unless those settings specify the use of default (ROM) operating software.

For **network,** *filename* is the new name for the network configuration file and optional *address* is the Internet address of a specific network host or a subnet broadcast address.

For **host,** *filename* is the host configuration file name and optional *address* is the Internet address of a specific network host or a subnet broadcast address.

bridge *number* [*keyword value*]
no bridge *number*

Sets bridging parameters, where *number* is an integer from 1 to 10 that identifies a spanning tree, and *keyword* and *value* are as follows:

Keyword	Value
forward-delay	*seconds*
hello-time	*seconds*
max-age	*seconds*
priority	*8-bit-number*

The **no bridge** command halts bridging activity for all interfaces related to spanning tree *number.*

decnet *parameter value*
no decnet

Sets DECnet Phase IV configuration parameters, where *parameter* and *value* are as follows:

Parameter	Value	Valid Range*	Default
address	*X.Y*	Area, node numbers	None
area-max-cost	*cost*	1-1022	1022
area-max-hops	*hops*	1-30	30
hello-timer	*seconds*	1-8191	15
max-address	*node*	1-1023	255
max-area	*area*	1-63	63
max-cost	*cost*	1-1022	1022
max-hops	*hops*	1-30	30
max-visits	*visits*	1-63	63
node-type	*keyword*	**area, routing-iv**	**routing-iv**
router-priority	*priority*	0-127	64
routing-timer	*seconds*	1-65535	40

*Unsigned decimal numbers, except for **address** and **node-type.**

For details of these parameters, see Chapter 16, "Using DECnet Protocol". The **no decnet** command disables DECnet processing.

default-network *network-address*

Specifies candidate default routes for non-IGRP gateway servers and/or exterior gateways, where *network-address* is the Internet address of the network.

domain-name *name*
no domain-name

Defines a default domain name for use in completing unqualified domain names. (Do not include in *name* the period that separates an unqualified name from the domain name.) The **no domain-name** command removes the default domain name.

egp-neighbor *gateway autonomous-system* **[primary]**

Defines a neighbor or primary gateway for a gateway server using EGP, where *gateway* is the Internet address of the neighbor gateway, *autonomous-system* is the autonomous system number of the gateway server, and **primary** identifies *gateway* as a core gateway.

host *name* [*port*] *address1 address2 ... address8*

Defines a host-name-to-address mapping, where *host* is the host name and each *address* value is an Internet address for that host. Optional *port* is the decimal port number; the default is the Telnet port (decimal 23). The host-name-to-address mapping should be constant across an autonomous system and specified in the network configuration files.

hostname *name*

Specifies the name used in prompts and default configuration file names, where *name* is the host name for the gateway server. This command preserves case distinction in *name*.

interface *type unit*

Identifies a network interface to which subsequent interface subcommands apply, where *type* is **ethernet, serial,** or **ddn-1822,** and *unit* is an integer that specifies the *n*th interface of type *type* (starting with 0 for the first interface of that type). See also "Interface Subcommands" later in this appendix.

line *first-line* [*last-line*]

Identifies a terminal line or lines to which subsequent line subcommands apply, where *first-line* is the octal number of the terminal line or the first line in a contiguous group and optional *last-line* is the octal number of the last line in the group. See also "Line Subcommands" later in this appendix.

[no] logging *address*

Sends debugging output and system bug messages to a syslog server host, where *address* is the address of that host. The **no logging** command ends the sending of logging messages to the specified host.

logging {console|trap} *level*
no logging {console|trap}

Defines the syslog level of messages displayed on the console terminal as those with a level at or below *level*, where **console** limits debugging messages (the default *level* is **informational**) and **trap** limits nondebugging messages (the default *level* is **warnings**). Optional *level*, in order from highest to lowest level, is **emergencies, alerts, critical, errors, warnings, notifications, informational,** or **debugging.** The **no logging console** or **no logging trap** command re-enables display of all syslog message levels.

name-server *server-address1 server-address2 ... server-address6*

Specifies up to six name-serving hosts to use for name and address resolution, where each *server-address* value is the Internet address of a name-serving host.

primary-neighbors *number*

Establishes the maximum number of primary gateways with which the gateway server may simultaneously exchange EGP information, where *number* is a decimal integer.

route *network gateway* **[default]**
no route *network gateway*

Defines a static routing entry, where *network* is the Internet address of the target network or subnet, *gateway* is the Internet address of a gateway that can reach that network, and **default** marks the route as a path toward a "smart" gateway. The **no route** command removes the specified static routing entry.

route *network interface unit*
no route *network interface unit*

Establishes multiple networks or subnets on an interface, where *network* is the Internet address of the target network or subnet, *interface* is **ethernet, serial,** or **ddn-1822,** and *unit* is an integer that specifies the *n*th interface of this type.

router *protocol* [*autonomous-system*]
no router *protocol*

Establishes a routing process, where *protocol* is **rip, egp, hello, igrp, chaos,** or **gwinfo,** and *autonomous-system* is a 16-bit decimal number (used only by IGRP). The **no router** command shuts down a routing process. See also "Router Subcommands" later in this appendix.

[no] service *keyword*

Enables the specified network service or function, where *keyword* is as follows:

Keyword	Service/Function	Default
config	TFTP autoloading of configuration files	Enabled
domain	Domain Name System for host-name-to-address translation	Enabled
ipname	IEN-116 Name Service for host-name-to-address translation	Disabled
iso2	IEEE 802.2-compatible (SNAP format) ARP requests	Disabled
router	Automatic routing	Enabled
probe	Probe protocol for IEEE 802.3 address resolution	Enabled
subnet-zero	Routing updates for subnet zero	Disabled
route-cache	High-speed switching of Internet Protocols on Multi-port Communications Interfaces	Enabled

no snmp-server

Disables SNMP server operation on the gateway server.

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

Specifies an access list to use when the gateway server determines which hosts can send SNMP requests.

[no] snmp-server host *name*

Specifies which host or hosts can receive SNMP TRAP messages.

snmp-server packet-size *size*

Specifies the maxiumum allowable size for SNMP packets.

[no] snmp-server session [*string*]

Enables SNMP server operation on the gateway server, where *string* specifies a session string for use in permitting or denying GET and SET operations. If *string* is omitted, all GET operations are permitted and all SET operations are denied.

[no] snmp-server trap-authentication

Enables sending of TRAP messages when the gateway server processes a packet with an incorrect session string.

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

Configures the gateway server to act as a TFTP-server host, where *filename* is the name of the gateway server ROM "file" and *access-list* is the number of an access list that specifies which hosts may receive the ROM software. With TFTP service enabled, the gateway server responds to TFTP read requests for configuration file *filename* by supplying its ROM contents as an operating system file. By default, the gateway server does not act as a TFTP server.

xns address [*ethernet-address*]
no xns address

Initializes XNS support in the gateway server. If specified, the gateway server uses *ethernet-address* as the system XNS address. Otherwise, the gateway server uses the first 10-megabits/second Ethernet interface address found. The **no xns address** command turns off XNS support.

Interface Subcommands

[no] access-group *list*

Assigns an access list to a network interface, where *list* is an access-list number from 1 through 199. The **no access-group** command removes the access-list assignment.

address *address subnet-mask*

Sets the interface address and subnet mask, where *address* is an Internet address and *subnet-mask* is a mask of address bits in Internet format that specifies the network portion of the address.

bandwidth *kilobits*

Specifies the bandwidth value for an interface, where *kilobits* is the bandwidth in kilobits/second. This subcommand sets an informational parameter; it does not affect hardware or media settings.

bridge-group *number*

Identifies an interface as part of the spanning tree (bridging group) *number*.

broadcast-address *address*

Sets the Internet broadcast address of an interface, where *address* is an Internet address for the gateway server to use in all broadcasts. The default is 32 bits of all ones.

clockrate *rate*

Sets the internal clock rate of a serial port on the Multi-port Communications Interface to *rate*, where *rate* is 1.2, 2.4, 4.8, 9.6, 19.2, 38.4, 56, or 64 kilobits/second.

decnet cost *cost-value*
no decnet cost

Defines the cost added to the metric for routes that use the interface, where *cost-value* is a decimal number from 1 through 25; there is no default. The **no decnet cost** command disables DECnet processing for the interface.

delay *tens-of-microseconds*

Specifies the delay value for an interface or a network segment, where *tens-of-microseconds* is the delay in tens of microseconds. This subcommand sets an informational parameter; it does not affect hardware or media settings.

encapsulation *method*

Specifies a method of data encapsulation. For Ethernet interfaces, *method* is **arpa**, **iso1**, or **iso2**. For serial line interfaces, *method* is **hdlc**, **lapb**, **x25**, **ddnx25**, **lapb-dce**, **x25-dce**, **ddnx25-dce**, **multi-lapb**, or **multi-lapb-dce**.

hdh {packet|message}

Sets the mode for the HDH protocol (HDLC Distant Host or 1822-J), where **packet** or **message** specifies the mode.

helper-address *address*

Configures an interface to forward UDP broadcasts to a specified address, where *address* is an Internet directed broadcast address or a host address.

hold-queue *number*

Sets the hold queue limit, where *number* is a decimal integer.

[no] interleave

Improves throughput if one or both ports on a CSC-S interface is running at 20 kilobits per second or less. The **no interleave** command disables the throughput enhancement.

lapb *parameter value*

Sets X.25 Level 2 parameters, where *parameter* and *value* are as follows:

Parameter	Value	Valid Range	Default
k	*frames*	1-7	7
n1	*bits*	1-12000	12000
n2	*times*	1-255	20
th	*milliseconds*	1-T1	2000
t1	*milliseconds*	1-64000	3000

For details of these parameters, see Chapter 15, "Using X.25 Protocols".

[no] loopback

Sets the appropriate hardware flags to start software and/or hardware loopback operation for an interface. The **no loopback** command restores the interface to normal operation.

mtu *bytes*

Sets the maximum transmission unit (MTU) of an interface, where *bytes* is a decimal integer that specifies the MTU size. For Ethernet and serial interfaces, the maximum (and default) is 1500; for DDN 1822 interfaces, the maximum (and default) is 1006.

[no] redirects

Re-enables the sending of ICMP Host Redirect messages. The **no redirects** command disables this function.

[no] shutdown

Disables an interface. The **no shutdown** command re-enables an interface.

x25 *parameter value*

Sets X.25 Level 3 parameters, where *parameter* and *value* are as follows:

Parameter	Value	Valid Range	Default
address	*X.121-address*		none
hic	*channel*	1-1024	1024
hoc	*channel*	1-1024	1024
htc	*channel*	1-1024	1024
idle	*minutes*		0
ips	*bytes*	1-1024	128
lic	*channel*	1-1024	1
loc	*channel*	1-1024	1
ltc	*channel*	1-1024	1
modulo	*modulus*	8, 128	8
nvc	*count*	1-4	1
ops	*bytes*	1-1024	128
t10	*seconds*		60
t11	*seconds*		180
t12	*seconds*		60
t13	*seconds*		60
t20	*seconds*		180
t21	*seconds*		200
t22	*seconds*		180
t23	*seconds*		180
th	*packets*	0-*win*	0
win	*packets*		2
wout	*packets*		2

For details of these parameters, see Chapter 15, "Using X.25 Protocols".

x25 accept-reverse

Enables acceptance of Reverse Charge calls from the network attached to the interface.

[no] x25 facility cug *number*

Uses a Closed User Group number from 1 to 99, as agreed by the calling and called networks to specify an extra measure of security.

[no] x25 facility {packetsize|windowsize} *in-value out-value*

Overrides current interface settings on a per-call basis, where **packetsize** overrides the input and output packet size parameters, and **windowsize** overrides the input and output window sizes. For both keywords, *in-value* and *out-value* specify the input and output values.

[no] x25 facility reverse

Requests that all calls be placed such that the remote system pays for usage charges.

x25 linkrestart

Forces a packet-level restart when the link level restarts, as required by some networks.

[no] x25 map *protocol protocol-address X.121-address* [*flags*]

Specifies a network-protocol-to-X.121 address mapping for the interface, where *protocol* is **ip, decnet,** or **xns**; *protocol-address* is a protocol-specific address; and *X.121-address* is the corresponding X.121 address. Optional *flags* can be any combination of the following:

accept-reverse
broadcast
cug *number*
packetsize *in-size out-size*
reverse
windowsize *in-size out-size*

For descriptions of these options, see Chapter 15, "Using X.25 Protocols".

[no] x25 pvc *circuit protocol protocol-address*

Specifies a permanent virtual circuit (PVC) with the virtual circuit number *circuit* connecting to the host at address *protocol-address* using protocol *protocol*, where *protocol* is **ip, decnet,** or **xns.**

x25 suppress-calling-address

Causes outgoing calls to omit the calling address, as required by some networks that expect no calling address unless it is a subaddress.

xns network *number*

Specifies the number of the XNS network to which the interface is connected.

Line Subcommands

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

Restricts connections on the line to Internet addresses in the specified access list, where *list* is an access-list number from 1 through 99, **in** applies to incoming connections, and **out** applies to outgoing Telnet connections. The **no access-class** command removes access restrictions on the line for the specified connections.

escape-character *decimal-number*
no escape-character

Defines the escape character, where *decimal-number* is the decimal representation of the ASCII character or the character itself. The **no escape-character** command reinstates the default escape character, CTRL-^.

exec-timeout *minutes*
no exec-timeout

Sets the interval that the EXEC waits for user input before resuming the current connection, where *minutes* is the number of minutes. The default is 10; 0 specifies no timeouts. The **no exec-timeout** command removes the timeout definition.

length *screen-length*

Sets the terminal screen length, where *screen-length* is the number of lines on the screen. The default is 24; 0 disables pausing between screens of output.

location *text*
no location

Enters information about the terminal for use in network monitoring, where *text* is the information. The characters included in *text* begin with the first non-blank character after *location* and end at the end of the line; no quote marks are needed. The **no location** command removes the information.

login

Prompts the user to give a password before an EXEC process is created on the line.

notify
no notify

Sets a line with multiple connections to inform a user when output is pending on a connection other than the current connection. The **no notify** command ends this notification.

pad *decimal-number count*
no pad *decimal-number*

Sets padding of the specified output character, where *decimal-number* is the decimal representation of the ASCII character, and *count* is the number of NUL bytes sent after that character. The **no pad** command removes padding for the specified output character.

password *password*
no password

Provides security for the line, where *password* is the line password. For line 0, *password* is the password for the privileged command mode. The gateway server prompts for the password whenever a user enters the *enable* command. The **no password** command removes the password.

no session-timeout

Sets the interval that the gateway server waits for user input before closing the connection to a remote computer and returning the terminal to the idle state, where *minutes* is the number of minutes. The default is 0, indicating no session timeout. The **no session-timeout** command removes the timeout definition.

terminal-type *text*
no terminal-type

Records the type of terminal connected to the line for use in the Telnet terminal-type negotiation, where *text* is the terminal type. The **no terminal-type** command removes the terminal-type record.

vacant-message *c message c*
no vacant-message

Defines a message to be printed on the screen of a vacant terminal, where *c* is a delimiting character and *message* is one or more lines of text. You cannot use the delimiting character in the message. The **no vacant-message** command removes the message.

Router Subcommands

[no] capacity *weight* [*address mask*]

Rates the traffic-handling capacity of a gateway server, where *weight* is an integer that reflects the rating. The higher the number, the higher the traffic-handling capacity; the default is 10. Specify a particular gateway or group of gateways with *address* and *mask*, which are Internet addresses. The **no capacity** command removes capacity information.

default-metric {*number*|*bandwidth delay reliability loading mtu*}

Used with **redistribute**, assigns the same metric value to all derived routes. If **redistribute** specifies RIP, EGP, or HELLO, *number* is an unsigned decimal number. If **redistribute** specifies IGRP, *bandwidth* is the minimum bandwidth of the route in kilobits/second, *delay* is the route delay in tens of microseconds, *reliability* is a number between 0 and 255 indicating the likelihood a packet will be transmitted (255 is 100% reliability), *loading* is the effective bandwidth of the route in kilobits/second, and *mtu* is the minimum MTU (Maximum Transmission Unit) of the route in bytes. The **default-metric** command should precede the **redistribute** router subcommand.

[no] distance *weight* [*address mask*]

Defines an administrative distance, where *weight* is an integer from 10 through 255. Specify a particular gateway or group of gateways with *address* and *mask*, which are Internet addresses. The **no distance** command removes an administrative distance.

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

Specifies filtering of routing updates, where *access-list* is the number of an access list, **in** applies filtering based on the source address of incoming routing updates, and **out** applies filtering to the contents of outgoing routing updates.

[no] hold-routes

Disables aging of routes under EGP routing protocol. The **no hold-routes** command re-enables the aging of routes.

[no] network *address* [*passive*]

Sends routing information to a directly connected network, where *address* is the Internet address of the network or a subnet. Optional **passive** suppresses the broadcast of routing updates on the specified network.

[no] redistribute *protocol* [*autonomous-system*]

Redistributes routing information from another protocol, where *protocol* is **static, rip, egp, hello,** or **igrp,** and *autonomous-system* is a 16-bit decimal number (used only by IGRP).

[no] variance *multiplier*

Sets a range of metric values the gateway server uses to establish parallel paths, where *multiplier* is an integer. The default *multiplier* value is 1. The **no variance** subcommand reinstates the default value.

Appendix C
ASCII Character Set

Many of the configuration commands described in this manual require the decimal representation of an ASCII character. Table C-1 provides translations from ASCII character to decimal number. To find the decimal equivalent of the ASCII character, add the row heading and column heading numbers together.

Table C-1. ASCII-to-Decimal Translation Table

	0	1	2	3	4	5	6	7	8	9
0	^@	^A	^B	^C	^D	^E	^F	^G	^H	^I
10	^J	^K	^L	^M	^N	^O	^P	^Q	^R	^S
20	^T	^U	^V	^W	^X	^Y	^Z	ESC	FS	GS
30	^^	US	SP	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	DEL		

Note: In the first four rows, the initial "^" character represents holding down the CTRL key while pressing the key indicated.

Appendix D
Network Acronym List

This appendix spells out the acronyms used in this manual. The list includes some names that appear to be acronyms but are not; a brief description explains these names.

ARP	Address Resolution Protocol
ARPA	*See* DARPA
BootP	Boot Protocol
CCITT	International Telegraph and Telephone Consultative Committee
CSU/DSU	Customer Service Unit/Digital Service Unit
DARPA	Defense Advanced Research Projects Agency
DCA	Defense Communications Agency
DCE	Data Communications Equipment
DDN	Defense Data Network
DNS	Domain Name System
DoD	The United States Department of Defense
DTE	Data Terminal Equipment
EGP	Exterior Gateway Protocol
FTP	File Transfer Protocol
HDLC	High-Level Data Link Control
HELLO	A routing protocol (not an acronym)
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IGRP	Interior Gateway Routing Protocol

IMP	Interface Message Processor
IP	Internet Protocol
ISO	International Standards Organization
LAPB	Link Access Procedure, Balanced
MAC	Medium Access Control
MAU	Medium Attachment Unit
MTU	Maximum Transmission Unit
Probe	Address resolution protocol developed by Hewlett-Packard (not an acronym)
PSN	Packet Switch Node
RARP	Reverse Address Resolution Protocol
RFC	Request For Comments
RIP	Routing Information Protocol
SAP	Service Access Point
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Monitoring Protocol
SNAP	Subnet Access Point
TCP	Transmission Control Protocol
TELNET	Terminal-level communication protocol (not an acronym)
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol
XNS	Xerox Network Service

cisco Systems

Index

Symbol

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

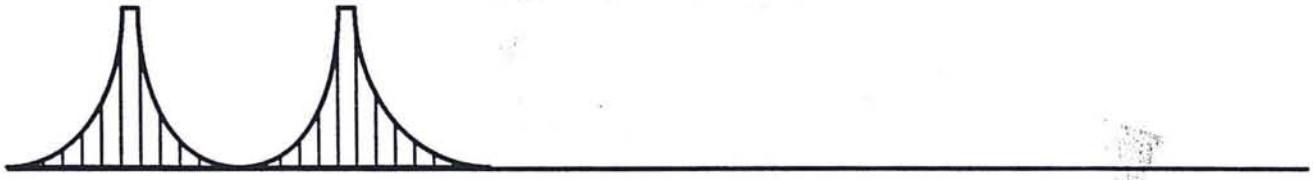
T

U

V

W

X



cisco Systems
Gateway Server 8.0 Release Note

GS-10, Version C, December 11, 1989

Notice of Restricted Rights

Online versions available at `ftp.cisco.com`: gs.psf (PostScript); gs.psf.Z (compressed PostScript)

Contents

List of Figures

List of Tables

Revision History

This is Version C of the *Gateway Server 8.0 Release Notes.* The following corrections and/or changes have been made to the Version B release note:

- Information about obtaining online copies of this document has been added to the inside title page.
- An additional caveat was included for users of Netronix Token Ring firmware in section 1.5.
- A note about the ROM version levels needed to use secondary bootstrap has been added in section 2.1.
- Use of the **debug broadcast** and **keepalive** commands has been clarified in section 2.3.
- Additional information about the TACACS server provided by **cisco Systems** has been added to section 2.6.
- Configuration restrictions for using secondary addressing have been added to section 3.1.3.
- Clarification of the options addressed by the IP Security Option has been added to section 3.2.
- New CLNS global configuration commands have been added to section 3.8.6. (Note that the old syntax for these commands is still supported.)
- A correction for how the Novell network number is written has been added to section 3.9.1.
- A workaround for using Token Ring source-route packets with Novell IPX routing has been added to section 3.9.1.
- A description of Novell access lists has been added to section 3.9.
- The **multiring** command defaults have changed; the new description of the defaults is in section 4.2.1.
- The facilities and parameters supported by X.25 switching have been clarified in section 5.2.
- The description about the bandwidth field display of the **show interface** command has been clarified in section 6.1.
- A section describing known watchdog timer mechanism problems on the CSC/2 Processor card has been added to chapter 6.

A new section has been added in chapter 1 that describes modifications to the software for the 8.0(9) software release.

Readers familiar with Version B of the release note will also notice minor grammatical and formatting changes throughout the text; these were made to enhance readability.

1 Overview

This document summarizes the functions added to the gateway server since the July 1988 version of the *Gateway System Manual.* Software versions documented include Releases 7.0, 7.1 and 8.0. This documents supersedes all previous release notes for the *Gateway System Manual* dated July 1988.

Consult the *Terminal Server Release Note* for information about recent releases of the terminal server software.

The following sections describe the features of each software release.

1.1 Release 7.0 Features

New functionality in Release 7.0 includes:

- Support for SNMP, the Simple Network Management Protocol.
- Routing of the Xerox Network Service (XNS) protocol is supported. This includes support for XNS-based PC protocols such as 3COM 3+.
- Support for the high performance Multi-port Communications Interface (MCI) card which provides high switching rates for routing IP and bridging other protocols.
- Level 2 bridging is possible on systems using the MCI interfaces. This means that protocols that cannot otherwise be routed can be bridged instead. Level 2 bridging supports the spanning tree protocol for building large bridged networks.

Other changes to the 7.0 software include:

- **show ip** *keyword* EXEC commands added for IP protocol support.
- **clear arp** clears ICMP redirect and IP fast switch caches.
- **debug broadcast** command added to display debugging information about all broadcasts.
- **x25 hold-vc-timer** and **x25 ip-precedence** configuration commands added for X.25 support.
- **debug x25-vc** *lcn* extended for X.25 LCN-specific debugging.
- **boot** configuration command extended.
- **keepalive** configuration command added to monitor the network interface.

- **logging buffered** configuration command has been added to the **logging** command to allow logging activity to be stored in a buffer.
- Ability to disable sending ICMP redirect and unreachable messages.
- Support for DOD Basic IP Security Option.
- Sequence numbers have been added before each *syslog* message.
- Proxy ARP can be disabled.
- Unsolicited ARP replies are sent when an interface address changes.
- Ability to perform ARP for hosts that don't support ARP themselves.
- Networks can be set up for passive (listen only) routing.
- The default network in RIP and HELLO is subject to access checking.
- EGP has been rewritten to support multiple autonomous systems.
- DECnet access list support has been added.

1.2 Release 7.1 Features

New functionality in Release 7.1 includes:

- IEEE 802.5 Token Ring support.
- Routing of AppleTalk protocols.
- Introduction of the TRouter, a combination router and terminal server.
- Support for the Serial-port Communications Interface (SCI card), a 4-port serial interface card based on the MCI card.

Other changes to the 7.1 software include:

- SNMP community variables can be set as read-only or read-write.
- Support for **cisco**-specific SNMP variables.
- SNMP traps are supported.
- **show x25** *keyword* EXEC command extended for X.25 support.
- Blacker Front-End supported for DDN networks.
- **lapb th** parameter removed from the X.25 command set.
- **ip** configuration commands have been added or modified to support multi-protocol systems.

- Two IP addresses allowed per interface.
- The **interface** command **no ip route-cache** replaces the **no service route-cache** command.
- **no ip routing** command disables routing of the IP protocol.
- Bridging of IP datagrams supported.
- Single address IP access lists extended from 20 to 50 wildcard items.
- IP Source routing can be disabled.
- ICMP mask replies can be disabled.
- **arp** configuration command extended to support IP protocols.
- IP can select which UDP and ND broadcasts are forwarded.
- Routing updates can be disabled on a per interface basis.
- Default subnets are supported.
- **neighbor** command added to support split-horizon partitioning and point-to-point routing updates.
- XNS can now select which types of broadcasts are forwarded.
- XNS can now be routed over X.25 circuits.
- **show xns** *keyword* EXEC command has been extended to support the XNS protocol.
- **show decnet** *keyword* EXEC command extended for DECnet routing support.
- DECnet fast switching rates supported.
- MCI dynamically disables noisy network interfaces.
- Support for encrypting CSU/DSUs is available by configuring a pulse time.

The gateway server enhancements of Release 7.1 are supported only by the CSC/2 processor. The software level of the CSC/1-based systems has been frozen at Release 7.0.

1.3 Release 8.0 Features

Major new functionality in Release 8.0 includes:

- ISO Connectionless Services (CLNS) routing.
- Novell IPX routing.
- Apollo Domain routing.
- Multi-ring support for Token Ring.
- Token Ring local source-route bridging.
- IP accounting statistics.
- Fast IP re-routing when redundant routers fail.
- Performance improvements for IP and DECnet access lists.
- IP Security Option (RFC 1038).
- Border Gateway Protocol (BGP).
- X.25 switching.
- TACACS user and password verification.
- Improved handling of RIP defaults.
- Generalized PUP routing.
- AppleTalk network number access lists.
- XNS access lists.
- Novell access lists.
- Transparent bridging access lists.
- CSC/2 auxiliary console port support.

1.4 Release 8.0 Performance Enhancements

A number of changes were made to the 8.0 software to enhance the system's switching performance for IP, DECnet, and bridging.

IP fast switching was enhanced by adding IP access lists and accounting. There is now only a minimal performance loss with access lists and accounting.

It should be noted that the load-balancing behavior of parallel interfaces running IP fast switching is on a statistical basis. The load is shared, but on a per host

basis, not on a per packet basis as it is without fast switching. If you are using load balancing for increased bandwidth between individual host pairs, you may wish to disable IP fast switching. IP fast switching generally doesn't result in a significant performance gain for line speeds under 56 Kbps.

Interfaces routing DECnet now default to fast switching. DECnet access lists are also handled at high speed.

When bridging multicast or broadcast traffic, the necessary flooding is done rapidly in a system with two interfaces on a single MCI card. Multi-port bridges must do some data copying and so are inherently slower than two-port bridges.

1.5 Release 8.0 Caveats

For people familiar with earlier versions of the **cisco** router software, the following list highlights any unexpected behavior of an 8.0 system.

- Version 2.0 of the Token Ring firmware is required for proper operation of the Release 8.0 Token Ring support.
- If you are *routing* AppleTalk or Novell and *bridging* anything else, you need Version 1.5 of the MCI interface firmware.
- Version 2.12 of the LLC Token Ring firmware from Netronix is required for Novell.
- Many Release 8.0 images cannot be netbooted successfully by earlier versions of ROM software due to their increased code size. Large images are booted in a two-step process that requires the initial loading of a bootstrap image. Two boot files are now required instead of just one.
- Configuration restrictions relating to the concurrent use of DECnet, XNS, and Novell protocols are now enforced by the software.

1.6 Release 8.0(9) Maintenance Release

The following sections describe caveats about and modifications to the software for the 8.0(9) maintenance release. Important information about changes to commands, defaults and displays are printed in *italic* typeface.

1.6.1 Release 8.0(9) Caveats

- There is a problem in establishing static IP routes if the system is not configured with a routing process. The **ip route** global command gets parsed as an

ip route-cache interface subcommand. To work around this, specify a routing process and provide a network address that is not connected to the router. It will then be possible to assign and save IP static routes. The correction for this problem will be available in the next maintenance release.

- There is a special circumstance in which netbooting system images may fail. Use of the **boot system** command from non-volatile configuration memory with a specific host address, or a manual boot from the ROM bootstrap monitor using a specific address, will fail when the path to the requested network does not use the first active IP interface in the system. The following alternatives are available to work around this situation. A correction for this situation will be available in the next maintenance release.

  1. Use a broadcast address instead; an example follows:

     non-volatile memory: `boot system cisco-gs-80 255.255.255.255`
     monitor: `b cisco-gs-80 255.255.255.255`

  2. Use the processor configuration register bits 0-3 to select a predefined system image name. See the *Gateway System Manual* for more information.

1.6.2 Release 8.0(9) Modifications

SNMP

Software now prevents SNMP from taking over the system. SNMP will now voluntarily relinquish the processor after successfully sending a single SNMP response packet.

Software now frees up storage allocation when trap authentication is disabled.

When reporting the time of the last interface state change, software returns the actual time, not the elapsed time.

IP

Always initialize the time-to-live (TTL) field when sending echo packets. It has been observed that a packet would be echoed, but the reply would never make it back to the sender due to an incorrect TTL value.

When checking to see if the system should absorb a broadcast, it will now ignore the IP address of interfaces that have IP disabled. This problem was noticed when trying to boot a router by using BOOTP to learn its address.

The HELLO routing protocol no longer generates spurious messages when redistributing IGRP information.

The **redistribute** and **distribution-list** subcommands provide correct updates when using EGP routing. When using EGP, write out the **neighbor** subcommand for EGP neighbors in autonomous systems other than our own.

The system no longer sends out ICMP Mask Request messages. These messages can potentially cause problems for SUN workstations.

Software changes have made **offset-list a router** subcommand rather than an **interface** subcommand.

When using TFTP to download a file, buffer overflow errors are now considered fatal, and the transfer will abort. Previously, the system would attempt to execute an incomplete system image which could cause errors.

The IGRP update **debug** messages will correctly display the proper distance to networks.

XNS

Software now only accepts routing updates from the well known XNS socket.

Software prevents XNS network numbers 0 and −1 from being entered into the routing table with a static routing command or via a RIP update.

Appletalk

Software now displays the router name in a format consistent with how Apple displays their routers' names.

Software now properly handles an incoming ZIP request that contains zero network numbers. It also includes the proper number of bytes in a ZIP request. Previous software was sending an extra byte.

Software now verifies AppleTalk packet lengths against the hardware reported length. Sometimes previous software accepted truncated packets.

CLNS

Software now clears all CLNS neighbors when changing the interface encapsulation type.

System error messages have changed to include the correct ISH in the message. Also, some **debug** messages have been cleaned up.

The minimum size of an echo packet has been changed from 30 to 58 bytes.

Traffic statistics for echo replies are now correct.

The command **clns NET** nsap *has been provided to allow configuring the router as an End System only. To enable the router as an Intermediate System, use the*

clns routing *command. The previous CLNS command* **clns routing** nsap *is still supported.*

The **clns route** nsap-prefix **discard** *command has been added, which allows administrators to explicitly throw away packets for some routes. This is useful in preventing routing loops.*

The default time interval value for generating error report PDUs (ERPDU) has been changed from 100 to 10 milliseconds.

Code has been added for updating entries in the CLNS cache lookup table.

Novell

In compliance with Novell IPX specifications, all displays and inputs of network numbers and service types default to hexadecimal.

Fixed software in which occasional incomplete SAP updates would be sent out if the last SAP entry failed the split horizon check.

Several modifications were made in software which manipulates the SAP table. With the previous version, symptoms such as multiple entries for a server, entries not in correct order and incorrect hop counts have been observed. This also permits the proper operation of flash updates.

Corrected formatting of fields used in the **show novell server** command for large network and service type numbers. Also, use display control to give a screenful of server information at a time. Previously, displays would scroll through the whole list without pausing.

Enhanced SAP debugging output to include server names and types.

The correct encapsulation type appears in the **show novell interface** command display for non-MCI/SCI serial cards.

X.25

Forwarded calls are now reported in the **show interface** command display. This is useful when performing X.25 switching.

Software accounting for X.25 calls and clears reports correctly.

X.25 switch now cleans up after itself if it fails to create a TCP connection.

Software supports diagnostic codes as being optional per the specifications.

Software prevents entering the same protocol address in the X.25 map. If the protocol address already exists in the map (e.g. – x25 map ip 131.108.1.1 31370054061), then the user must de-assign 131.108.1.1 before mapping it to another X.121 address.

Miscellaneous

MCI Ethernet interfaces in 8.0 correctly report CRC errors.

Chaosnet support has been fixed. Any enabling of Chaosnet would cause a fatal system error.

When net-booting, the secondary bootstrap image "boot-csc2" no longer generates errors for unknown commands. Previously, this image would complain about numerous commands in the configuration memory it does not support.

Reduced the size of the secondary bootstrap image ("boot-csc2"). Some of the CSC/2 images (containing bridging and X.25) could not be successfully booted over the network and would cause a buffer overflow error.

Enhancements have been added for users running HP Officeshare and other software that use Probe packets for name service. Software continues to absorb Probe Virtual Name Address (VNA) Request and Reply packets if the system is routing IP. If the system receives some other type of Probe packet, it is offered it up for bridging.

The maximum number of networks that can be listed with the **network** configuration command has been raised from 30 to 200. This primarily affects BGP which can require large network tables.

PUP mapping commands are now placed earlier in the configuration file for proper operation of the **router gwinfo** command.

Serial line software has been modified to prevent fatal system errors when a bridged packet is received before the interface is set up for bridging.

The default state for the **multiring** *command on a Token Ring interface is now off. Previously it was set to on, which caused problems for some workstations running Novell Netware or 3COM 3+Open network operating systems. This does not affect any source-routed traffic, only routed protocols. Also, software now allows the* **multiring** *subcommand to be saved in configuration memory.*

New software changes also insure proper display of error messages when the system aborts and has automatic reboot enabled. Truncated PC addresses in error messages have been observed.

1.7 Current ROM Software Revision Levels

The following table describes the current software versions for **cisco** software. Refer to these version numbers when ordering software updates.

System	*Version*	*Description*
GS2	8.0(9)	CSC/2 Gateway Server
	Sets:	GS2-S, GS2-X, GS2-B, GS2-D, GS2-BX, GS2-BD
TS2	8.0(9)	CSC/2 Terminal Server
	Sets:	TS2-S, TS2-X, TS2-D
TR2	8.0(9)	CSC/2 Trouter
	Sets:	TR2-S, TR2-X, TR2-D
PT2	7.1(9)	CSC/2 Protocol Translator
	Sets:	PT2
STS-10	7.1(11)	Small Terminal Server
GS1	7.0(16)	CSC/1 Gateway Server
	Sets:	GS1-S, GS1-X, GS2-B, GS1-D, GS1-BX, GS1-BD
TS1	7.0(16)	CSC/1 Terminal Server
	Sets:	TS1-S, TS1-X, TS1-D

Key

S - Standard system software
X - Standard + Commercial X.25 software
B - Standard + Bridging software
D - Standard + DDN X.25 software
BX - Standard + Bridging + Commercial X.25 software
BD - Standard + Bridging + DDN X.25 software

2 Configuration and Management Commands

This chapter describes the **interface** and general configuration commands added to Releases 7.0, 7.1 and 8.0 to aid in the management of the gateway systems. The commands are divided into these categories:

- Boot configuration commands
- Token Ring interface commands
- General system commands
- Logging commands
- Simple Network Management Protocol (SNMP) commands
- TACACS security commands

As the system software evolves, **cisco** engineers attempt to keep the command structure consistent. At times this involves changing command names. When this occurs, the earlier command will also be recognized for several releases before being removed from the system. In Release 8.0, the configuration commands listed in Table 1 changed form.

Earlier Form	*New Form*	*Comment*
name-server	**ip name-server**	Global IP command
route	**ip route**	Global IP command
default-network	**ip default-network**	Global IP command
proxy-arp-class	**ip proxy-arp**	Interface IP subcommand
proxy-arp-list	**proxy-arp**	Interface IP subcommand
mtu	**ip mtu**	Interface IP subcommand
default-gateway	**ip default-gateway**	TS only Global IP command

Table 1: Changed Configuration Commands

Table 2 lists the EXEC commands that changed form in Release 8.0.

2.1 Netbooting and Boot Configuration Commands

The **cisco** software can boot 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 8.0 release software are so large that two copies of it will not fit in memory. Therefore, the 8.0 netboot algorithm uses a secondary bootstrap system image to netboot the desired system image. (Note that you need the 8.0 ROMs to do this secondary bootstrap.)

Earlier Form	*New Form*
clear route	**clear ip route**
debug appletalk	**debug apple-packet**
debug nbp	**debug apple-nbp**
debug rtmp	**debug apple-routing**
debug zip	**debug apple-zip**
debug gwinfo	**debug pup-routing**
debug chaos	**debug chaos-routing**
debug decnet	**debug decnet-routing**
debug xns	**debug xns-routing**
debug egp	**debug ip-egp**
debug egp-events	**debug ip-egp-events**
debug hello	**debug ip-hello**
debug icmp	**debug ip-icmp**
debug igrp	**debug ip-igrp**
debug rip	**debug ip-rip**
debug routing	**debug ip-routing**
debug tcp	**debug ip-tcp**
debug udp	**debug ip-udp**

Table 2: Changed EXEC Commands

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/1 processors is *boot-csc1*. The secondary bootstrap for CSC/2 processors is *boot-csc2*.

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.

The **boot** command has been extended to allow for lists of filename and host address pairs. Multiple **boot** commands build an ordered list of filenames and hosts which the system scans until it loads the appropriate host or network configuration file or system boot image. For example, these commands:

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

causes a system to look first for *fred-confg* on 192.31.7.24 and if it doesn't load the file, to then try *joe-confg* on 192.31.7.19. If the system finds neither file, a background process tries at ten minute intervals to load one or the other files.

To add a filename and address pair to the end of a list, the command is

boot system|host|network *filename address*

To delete a specific filename and address pair from the list, the command is:

no boot system|host|network *filename address*

To delete an entire list, the command is:

no boot system|host|network

2.2 Token Ring Interfaces

The **cisco** Token Ring interface introduced in Release 7.1 provides interconnection of **cisco** routers and terminal servers to IEEE 802.5 and IBM-compatible Token Ring media. The implementation is based on the TI TMS380 and Intel 80186 controllers. The data rate for the Token Ring is four megabits per second.

2.2.1 Token Ring Protocol Support

The **cisco** Token Ring interfaces support 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 following protocols can be routed over Token Ring as of Release 8.0. The routing support interacts correctly with source-route bridges.

- Novell IPX
- XNS (including Ungermann-Bass and 3COM varieties)
- Internet Protocol
- Connectionless Network Protocol (CLNS, ISO 8473)

Support for IBM-style source-route bridging is discussed in detail in Section 4.2. In Release 8.0, only local source-route bridging is supported. This means that the two rings must not be separated by another network such as an Ethernet, or by serial line. This restriction will be lifted in the next release.

2.2.2 Token Ring Interface Configuration Commands

All Token Ring configuration parameters are determined using subcommands of the **interface** command. (See section 4.2.1 for the Token Ring configuration commands.) The **interface** command for the Token Ring takes the form:

interface tokenring *unit*

where *unit* is the interface unit number. There can be a total of four Token Ring interfaces installed in a gateway server; these are numbered 0 through 3.

The following list describes some default Token Ring parameters.

- Bandwidth defaults to 4000 Kbps.
- Delay defaults to 2500 microseconds.
- Encapsulation defaults to **snap**.
- The MTU defaults to 4464 bytes, the maximum permitted.
- The hold queue defaults to 40 packets.
- Keepalive is enabled by default. Every 10 seconds an interface sends a loopback packet to itself. If an interface misses several such packets, the interface is declared down and is automatically reset.

2.2.3 Monitoring the Token Ring

There are two EXEC **debug** commands for debugging Token Ring problems in the field.

debug token-events

The **debug token-events** command reports changes in the Token Ring interface and ring status.

debug token-ring

The **debug token-ring** command reports several lines of information for each packet sent or received and is intended for low traffic, detailed debugging.

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. The last ring status message is displayed in the EXEC **show interface** display for a Token Ring interface.

The status messages are:

Signal Loss The controller detected an aberration in the signal while receiving a packet.

Hard Error Indicates that the interface is either transmitting or receiving Beacon frames. This generally denotes that the ring is broken. There may be a break in the physical cabling or an inserted interface may have died.

Soft Error The interface has detected an aberration on the ring and is transmitting a Report Error MAC frame. This generally occurs when a new station has inserted into the ring.

Ring Beacon The interface is transmitting Beacons onto the ring. Something is wrong with the ring.

Wire Fault The interface has detected a problem with its lobe wiring.

HW Removal The interface has detected an internal hardware error and removed itself from the ring.

Remote Removal When this interface has tried to insert into the ring it received a Remove MAC frame from another station on the ring. The interface is then de-inserted from the ring.

Counter Overflow An internal counter overflowed. This is not serious.

Only Station This interface has detected that it is the only interface on the ring.

Ring Recovery The interface has detected Claim Token MAC frames on the ring. It generally denotes that the ring is in a recovery mode. It also occurs when new stations insert.

2.3 General System Commands

New and modified general system commands include:

- A new **interface** configuration command:

  pulse-time *seconds*

  was added in Release 7.1 to enable pulsing DTR signals on MCI serial interfaces for a minimum interval of it seconds. 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 CSU/DSUs or other similar devices that use the toggling characteristic of the DTR signal to resynchronize.

- The EXEC **show arp** command has been modified in Release 8.0 to take an optional keyword specifying that only a part of the ARP table be displayed. Examples of keywords include **ip** and **appletalk**. Type **show arp** and a ? (question mark) to list all keywords.

- Beginning with Release 8.0, configuration commands typed in at the console (the **terminal** option of the **configure** command), are now executed on a line at a time basis for quicker feedback about misspellings and other mistakes.

- The CSC/2 processor supports an auxiliary RS-232 DTE port as of Release 8.0. This port can be used to attach to an RS-232 port of a CSU/DSU or protocol analyzer. Remote monitoring of that port can be performed by connecting to the TCP port whose address is 2000 decimal plus the line number of the auxiliary port. For example, if the auxiliary port was line 1 (obtained from the **show users all** display), then the TCP port would be 2001. A special cable must be ordered for use with this auxiliary port. For configuration purposes the auxiliary port is **line aux 0**. The CSC/2 auxiliary port asserts DTR only when a Telnet connection is established. The CSC/1 auxiliary port does not implement DTR. Neither the CSC/1 or CSC/2 console ports use RTS/CTS handshaking for flow control.

- The EXEC **show stacks** command was modified in Release 8.0 to display the reason for the last reboot. If the system was reloaded because of a system failure, a saved system stack trace is displayed. This is useful to **cisco** technical staff in analyzing crashes in the field.

- The EXEC **debug broadcast** command was added in Release 7.0 to display information about all Level 2 broadcast packets received.

- Input throttling was added to Release 8.0 to avoid congesting the system with input. Packets will now be discarded if an interface has too many input packets outstanding in the system. To specify this behavior, the **interface** subcommand **hold-queue** was extended to apply to both input and output queues. The new form of the command is:

  [no] hold-queue *length* **in | out**

  The value for *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** option restores the default values for an interface.

- Serial interfaces using HDLC encapsulation and all Ethernet interfaces periodically send messages to ensure that the network connection is working properly. There is now an **interface** configuration subcommand that disables the sending of such keepalive messages. The subcommand is **[no] keepalive** [*interval*], where *interval* is the time in seconds between keepalives. The default is 10 seconds. When **no keepalive** is specified, no keepalive messages are sent and the network interface line protocol is forced up.

- The **interleave** subcommand of the **interface** configuration command has been removed. This command applied only to the CSC-S card (now obsolete) and has no effect in Version 2.9 (or later) of the system firmware.

- The global configuration command

  enable-password *password*

  was added in Release 8.0 to specify the password for enabling privileged mode. Previously, the password for the console line was used as the **enable** password. This new configuration, if specified, now permits the **enable** password to be different than the console password. If the **enable-password** command is not specified, the earlier behavior prevails.

2.4 The Logging Facility

This section describes enhancements made to the **cisco** software logging facility. The **logging** configuration command can be used to redirect output from the **debug** command, or from asynchronous events such as the transition of an interface going from up to down. Possible destinations for the output include the console terminal, a virtual terminal, or a UNIX host running a syslog server.

The **logging** command now includes these variations:

[no] logging on
[no] logging buffered
[no] logging *ip-address*
[no] logging monitor *level*
[no] logging console *level*
[no] logging trap *level*

Descriptions of each command variation, the levels available, and some notes about configuring a syslog daemon on a UNIX system follow.

[no] logging on

The **logging on** command enables logging to all destinations except the console. The default is to enable logging to all destinations. The **no logging on** command turns this function off.

[no] logging buffered

The **logging buffered** command copies log messages to an internal buffer instead of writing these messages to the console terminal. The messages are displayed using the **show logging** command. The buffer is circular in nature, so that newer messages overwrite older messages. The first message displayed is the oldest message in the buffer. Logging is normally not buffered, and the **no logging buffered** command restores this default.

[no] logging *ip-address*

The **logging** *ip-address* command adds an IP address from the list of syslog servers that should receive logging messages. The **no logging** *ip-address* deletes the IP address from the list.

logging monitor *level*
no logging monitor

The **logging monitor** command determines the minimum severity level of messages to be printed on terminal lines other than the console line. Logging messages at or above this severity level are printed. The terminal lines involved are those for which a terminal monitor command must be given in order for logging to occur. The arguments for *level* are described below.

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

logging console *level*
no logging console

The **logging console** command determines the minimum severity level of messages to be printed on the console line. Logging messages at or above this severity level are printed. The arguments for *level* are described below.

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

logging trap *level*
no logging trap

The **logging trap** command determines the minimum severity level of messages to be sent to syslog hosts specified by the **logging** *ip-address* command. Logging messages at or above this severity level are logged. The arguments for *level* are described below.

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

The *level* Arguments

In the above commands, the argument *level* is one of the following keywords (listed in order from highest to lowest severity):

- **emergencies** — The system is unusable.
- **alerts** — Action must be taken immediately.
- **critical** — Critical conditions.

- **errors** — Error conditions.
- **warnings** — Warning conditions.
- **notifications** — Normal but significant conditions.
- **informational** — Informational messages, only.
- **debugging** — Debug-level messages.

Four categories of the syslog messages are generated by the current software:

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

The EXEC **show logging** command displays the addresses and levels associated with the current logging setup, as well as ancillary statistics.

Setting Up a Syslog Daemon

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.debugging                    /usr/adm/logs/tiplog
```

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

The `debugging` 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.

2.5 Simple Network Management Protocol

Beginning with Release 7.0, the **cisco** gateway server and terminal server provide support for the Simple Network Management Protocol (SNMP). SNMP provides a means of accessing and setting the configuration and runtime parameters of your system. It is compatible with RFCs 1065, 1066 and 1067. The Management Information Base (MIB) supports all of RFCs 1065 and 1066.

2.5.1 Configuring the SNMP Server

SNMP is disabled by default and thus does not respond to or generate any traffic. The following configuration commands have been provided to enable and specify the SNMP server environment.

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

The **snmp-server community** command enables the SNMP server. The optional community string is a password-type argument that, when used, permits access to the SNMP protocol. By default, an SNMP community string permits read-only (RO) access. You can also add an argument to allow read-write (RW) access. If the *string* argument is omitted, SNMP read-only access is permitted for any community string used in the SNMP packets. An optional access list can be specified to control which IP addresses may use the community string. See RFC 1098 for more details about the SNMP server.

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

The **snmp-server access-list** command sets up an access list that validates which host can send requests. Packets from hosts not meeting the access list criteria are ignored.

snmp-server packetsize *bytes*

When receiving a request or generating a reply, the **snmp-server packetsize** configuration command allows control over the largest SNMP packet size permitted. The range is from a default of 484 bytes to a maximum of 8192 bytes.

[no] snmp-server host *name* [*community-string*]

The **snmp-server host** command specifies which host or hosts should receive TRAP messages. One command line is required for each host acting as a TRAP recipient. The *name* argument can be an IP address or a host name. An optional community string can also be specified.

[no] snmp-server trap-authentication

Normally the SNMP specifies that packets with an incorrect community should generate a TRAP message. This can be a security breach. By default, trap messages are not returned when an incorrect community is processed. The **snmp-server trap-authentication** command enables the sending of those trap messages.

no snmp-server

Use the **no snmp-server** command to disable the SNMP server after it has been started.

2.5.2 cisco-Specific SNMP Variables

There is a separate document in RFC 1066 format that describes the **cisco**-specific SNMP variables within the **cisco** portion of the MIB. It describes all of the **cisco**-specific variables and how to access them using SNMP. Below is a list of the major additions in Release 8.0.

- Support for DECnet, XNS, AppleTalk and Novell in **cisco** MIB space. This support is limited to returning the values of the **show** *protocol* **traffic** arrays.
- Support for the retrieval of IP accounting data.
- Support for TFTP loading of configuration information.

2.6 TACACS – User/Password Verification

With Release 8.0, the router and terminal server software have the ability to work with a server host to validate a user ID and password pair before allowing further use of the console or virtual terminal lines. The protocol used is TACACS, developed by the Defense Data Network for controlling access to their TAC terminal servers. **cisco Systems** provides an unsupported, sample TACACS server free of charge that runs on most UNIX systems. (Note that **cisco** will not port this server if it will not work on your box.)

2.6.1 TACACS Line Subcommands

The only command needed to set up a line for TACACS is a variation of the **login** line subcommand:

[no] login tacacs

The **login tacacs** command sets up a line for TACACS password verification. The form **no login** disables all password checking, while the simple form **login** enables the use of line-specific passwords.

2.6.2 TACACS Configuration Commands

The TACACS configuration commands follow:

[no] tacacs-server host *hostname*

The **tacacs-server host** command specifies the TACACS server host. The *hostname* argument specifies a name or address. By giving multiple **tacacs-server host** commands, multiple server hosts can be specified. The list is searched in the order given. The **no** keyword removes the server host.

[no] tacacs-server retransmit *retries*

The **tacacs-server retransmit** command specifies the number of times to go through the server list before giving up. All servers are tried and all will time-out before the retransmit count is increased. The default value of *retries* is 2. The **no** keyword restores the default.

[no] tacacs-server timeout *seconds*

The **tacacs-server timeout** command specifies the amount of time to wait for a server host to reply The default value for *seconds* is 5. The **no** keyword restores the default.

[no] tacacs-server attempts *count*

The **tacacs-server attempts** command controls the number of login attempts to be made on a line set up with the **login tacacs** command before giving up. The default value for *count* is 3. The **no** keyword restores the default.

3 Routing

This chapter describes commands that support routing protocols. The commands are divided into the following sections:

- General IP protocol support
- IPSO — IP Security Option
- IP Routing Enhancements
- Border Gateway Protocol
- DECnet routing
- XNS routing
- AppleTalk routing
- ISO Connectionless Network Service
- Novell IPX routing
- Apollo Domain routing
- PUP protocol

3.1 General IP Protocol Support

This section describes new commands and enhancements to the current commands that support the IP protocols with Release 8.0.

3.1.1 ARP Configuration Command

The **arp** configuration command has an additional optional argument, **alias**, which causes the proxy ARP support to respond as if the **cisco** gateway were that host.

arp *internet-address ethernet-address* **arpa|iso1|iso2 [alias]**

Proxy ARP is normally enabled on all IP interfaces. The **interface** subcommand **no ip proxy-arp** can be used to disable proxy ARP on a per interface basis.

The system now broadcasts an unsolicited ARP reply on an Ethernet interface when it comes up or changes its hardware address (for example, by running DECnet or XNS).

The **arp** command has also been extended to control interface-specific handling of the resolution of IP addresses into 48-bit Ethernet and Token Ring hardware addresses. When used as an **interface** command, the new command syntax is

[no] arp arpa|probe|snap

where **arpa** specifies standard Ethernet style ARP (RFC 826), **probe** specifies the HP-proprietary Probe protocol for IEEE 802.3 networks, and **snap** specifies ARP packets conforming to RFC 1042.

These commands effectively replace the earlier configuration commands **service probe** and **service iso2**. The older commands will be read correctly, but will no longer be written to non-volatile memory.

3.1.2 Global IP Configuration Commands

Following are the global IP configuration commands that specify system-wide parameters of the IP support. Preceding any of these commands with a **no** keyword undoes their effect and/or restores the default condition.

[no] ip accounting-threshold *threshold*

The **ip accounting-threshold** command is discussed in section 3.1.6.

[no] ip default-network *network*

The **ip default-network** command is the new form of the **default-network** command described in the *Gateway System Manual.*

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

The **ip forward-protocol** configuration command controls forwarding of physical and directed IP broadcasts. The command controls which protocols and ports are forwarded on an interface for which an **ip helper-address** command has been specified. **nd** is the ND protocol used by older diskless SUN workstations. **udp** is the UDP protocol. A UDP destination port can be specified to control which UDP services are forwarded. By default both UDP and ND forwarding are enabled if a helper address has been given for an interface. If no ports are specified, TFTP, Domain Name System, IEN-116, Time, NetBios, and BootP datagrams are forwarded by default.

ip host [*port*] *address*

The **ip host** [*port*] *address* command is the new form of the **host** command described in the *Gateway System Manual.*

ip name-server *address*

The **ip name-server** *address* command is the new form of the **name-server** command described in the *Gateway System Manual.*

ip route *network address*

The **ip route** *network address* command is the new form of the **route** command.

[no] ip routing

The command **ip routing** controls the system's ability to do IP routing, assuming that the system is running optional bridging-enabled software. This command is useful for turning off IP routing when setting up a system to bridge (as opposed to route) IP datagrams. See the explanations on bridging options in chapter 4. The default setting is to perform IP routing.

[no] ip source-route

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

3.1.3 Interface-Specific IP Configuration Commands

These are the interface-specific IP configuration commands. An **interface** command line must be given first to specify which network interface is being reconfigured. Preceding any of these commands with a **no** keyword undoes their effect or restores the default condition.

ip access-group *list*

The **ip access-group** command is new form of the **access-group** command described in the *Gateway System Manual.*

[no] ip accounting

The **ip accounting** command is discussed in section 3.1.6.

ip address *address subnet-mask* **[secondary]**

The **ip address** command is a new form of the **address** command (currently described in the *Gateway System Manual*). Additionally, the software supports multiple IP addresses per interface. Secondary addresses are treated like primary addresses except that the system never generates datagrams with secondary source addresses. IP broadcasts and ARP requests are handled properly, as are interface routes in the IP routing table.

Secondary IP addresses can be used in a variety of situations. The following are the most common applications. Other applications are certainly possible.

- There may not be enough host addresses for a particular network segment. For example, your subnetting allows up to 254 hosts per logical subnet, but on one physical subnet you need to have 300 host addresses. Using secondary IP addresses on the routers allows you to have two logical subnets using one physical subnet.
- Many older networks were built using Level 2 bridges. The judicious use of secondary addresses can aid in the transition to a subnetted, router-based network. Routers on an older, bridged segment can be easily made aware that there are many subnets on that segment.
- Secondary addresses may be used when two subnets of a single network might otherwise be separated by another network, a situation not permitted when subnets are in use. In these instances, the first network is *extended* or layered on top of the second network using secondary addresses.

Additionally, the following configuration restrictions must also be adhered to:

- A parallel router is a router which provides an additional physical path between two networks.
- A secondary IP address defines a secondary network which is logically imposed on top of the physical network. (The IP address consists of a network portion and a host portion.)
- If any of a set of parallel routers supports a secondary IP address, all of the routers in that set must support the same secondary network. (The host portion of the secondary IP address should be unique for each router.)
- Be aware that if any router on a network segment uses secondary addresses, all other routers with interfaces on that same segment must also use the same secondary addresses. An inconsistent use of secondary addresses on a network segment can lead quickly to routing loops.

ip broadcast-address *address*

The **ip broadcast-address** command is the new **broadcast-address** command described in the *Gateway System Manual.*

[no] ip directed-broadcast

The **no ip directed-broadcast** command disables forwarding of directed broadcasts on the interface. The default is to forward directed broadcasts.

[no] ip gdp

The **ip gdp** command is discussed in section 3.3.3.

ip helper-address *address*

The **ip helper-address** command is the new form of the **helper-address** described in the *Gateway System Manual.*

ip mask-reply

The **interface** subcommand **ip mask-reply** controls the sending of ICMP Mask Reply messages. The default is not to send Mask Replies.

ip mtu *bytes*

The **ip mtu** command is the new form of the **mtu** command described in the *Gateway System Manual.*

[no] ip proxy-arp

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

[no] ip redirects

The **no ip redirects** command disables sending ICMP redirects on the interface. ICMP redirects are normally sent.

[no] ip route-cache

The configuration command **service route-cache** has been removed. In its place is the **interface** subcommand **ip route-cache.** This command controls the use of a high-speed switching cache for IP routing. The cache is normally enabled.

[no] ip security *arguments*

The **ip security** command is described in section 3.2.2.

[no] ip unreachables

The **no ip unreachables** command disables sending ICMP unreachable messages on an interface. ICMP unreachables are normally sent.

3.1.4 IP Access Lists

There is now no practical limit to the number of wildcard items in a normal or extended IP access list. These access lists are now handled in the fast switching support for increased switching performance.

3.1.5 IP-Related EXEC Commands

Following is a list of the EXEC **show ip** keywords.

show ip accounting [checkpoint]

The **show ip accounting [checkpoint]** command is described in section 3.1.6.

show ip cache

The **show ip cache** command displays the contents of the IP fast switching cache.

show ip egp

The **show ip egp** command displays the status of current EGP neighbors.

show ip interface [*interface-name*]

The **show ip interface** command displays IP-related interface parameters.

show ip protocols

The **show ip protocols** command displays the configuration and status of any routing processes.

show ip route [*network*]

The **show ip route** command displays the IP rotuing table or a specific network in the routing table.

show ip traffic

The **show ip traffic** command displays IP protocol statistics.

ping

The **ping** command described in the *Gateway System Manual* has been extended to accept a data pattern when in verbose mode. The pattern is specified as a 16-bit hexadecimal number. The default pattern is 0xABCD. Patterns such as all ones or all zeros can be used to debug data sensitivity problems on CSU/DSUs.

3.1.6 IP Accounting

IP accounting has been added to the router software. Enabled on a per interface basis, the IP accounting support records the number of bytes and packets switched through the system on a source and destination IP address basis. Only transit IP traffic is measured; traffic generated by the router or terminating in the router is not included in the accounting statistics.

The **interface** subcommand

[no] ip accounting

enables or disables IP accounting for transit traffic outbound on an interface. It does not matter whether or not IP fast switching or IP access lists are being used on that interface; the numbers will be accurate.

The global configuration command

[no] ip accounting *threshold*

defines the maximum number of entries (source and destination address pairs) that the router accumulates. This prevents the IP accounting from possibly consuming all available free memory. This is quite possible in a router that is switching traffic for many hosts. The default threshold value is 512 entries. Overflows are noted.

To maintain accurate accounting totals, the router software maintains two accounting databases. These are an active and a checkpointed database.

The EXEC command

show ip accounting

command displays the active accounting database.

The command

show ip accounting checkpoint

displays the checkpointed database.

The following is an example of the **show ip accounting** display.

Source	Destination	Packets	Bytes
192.31.7.146	36.22.0.120	259	12945
131.108.4.3	192.31.7.50	723	49517
131.108.4.4	192.31.7.50	716	49357
192.31.7.50	131.108.4.4	862	35237
192.33.4.10	131.108.4.3	1	256
131.108.1.27	131.108.2.31	7442	2329847
131.108.2.31	131.108.1.27	8722	635857
131.108.2.37	192.31.7.24	606	53516
192.31.7.24	131.108.2.37	1428	112546

The checkpointed database is created from the active database (and the active database cleared) by the EXEC command

clear ip accounting

The checkpointed database can be cleared by the command **clear ip accounting checkpoint** or by issuing the **clear ip accounting** command again.

Facilities exist for using SNMP to retrieve IP accounting information. See section 2.5.2 on SNMP extensions.

3.2 IPSO — IP Security Option

All aspects of the IP Security Option (IPSO) are set up using configuration commands. This software version addresses both the Basic and Extended security options described in the latest draft of the IPSO.

The following list describes some of the abilities of the system.

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

3.2.1 IPSO Definitions

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

- *level* — The degree of sensitivity of information. For example, data marked TOPSECRET is more sensitive than data marked SECRET. Table 3 lists the level keywords used by the **cisco** software and their corresponding bit patterns.
- *authority* — An organization that defines the set of security levels that will be used in a network. For example, the Genser authority consists of level names defined by DCA. Table 4 lists the authority keywords used by the **cisco** software and their corresponding bit patterns.
- *label* — A combination of a security level and an authority or authorities.

3.2.2 IPSO Commands

All of the IPSO-related commands fall under the interface-specific **ip security** command.

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

The following commands offer minor modifications to the above commands. These commands return to default values whenever one of the above major commands is issued. There are some restrictions on where these commands can be used. These are noted in the command descriptions.

[no] ip security ignore-authorities

The **[no] ip security ignore-authorities** command ignores the authorities field of all incoming datagrams. The value used in place of this field will be the authority value declared for the given interface. This action is only allowed for single-level interfaces.

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

The **[no] ip security implicit-labelling** command accepts datagrams on the interface, even if they do not include a security option. For a single level interface, datagrams without an option assume the label assigned to the interface. For a multi-level interface, the second form of the command must be used. This label, provided by the user, is applied to datagrams without a security option.

[no] ip security extended-allowed

The **[no] ip security extended-allowed** command accepts datagrams on the interface that have an extended security option present. The default rejects the datagram immediately.

[no] ip security add

The **[no] ip security add** command ensures that all datagrams leaving the system on this interface contain a basic security option. If an outgoing datagram does not have a security option present, add one as the first IP option. The security label added to the option field is the label that was computed for this datagram when it first entered the system. Because this action is performed after all the security tests have been passed, this label will either be the same as or will fall within the range of the interface. This action is always enforced on multi-level interfaces.

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

Table 3: IPSO Level Keywords and Bit Patterns

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

Table 4: IPSO Authority Keywords and Bit Patterns

The following commands are used to enable or disable security on a particular interface:

[no] ip security

The **no ip security** command resets an interface to its default state, "dedicated, unclassified Genser." The **ip security** command can also take arguments, as described in the next paragraphs.

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

The **ip security dedicated** command sets the interface to the requested classifications and authorities. All traffic entering the system on this interface must have a security option that exactly matches this label. Any traffic leaving via this interface will have this label attached to it.

ip security multilevel *level1* [*authority1*...] **to** *level2 authority2* [*authority2*...]

The **ip security multilevel** command sets the interface to the requested range of classifications and authorities. All traffic entering or leaving the system must have a security option that falls within this range. Being "within range" requires that the following two conditions be met: First, the classification level must be greater

[no] ip security strip

The **[no] ip security strip** command removes any basic security option that may be present on a datagram leaving the system through this interface. This is performed after all security tests in the system have been passed and is not allowed for multi-level interfaces.

[no] ip security first

The **[no] ip security first** command prioritizes the presence of security options on a datagram. If a basic security option is present on an outgoing datagram, but it is not the first IP option, then it is moved to the front of the options field when this command is used.

Default Values for Minor Keywords

In order to fully comply with the IPSO, the default values for the minor keywords are slightly complicated. The default for all of the minor keywords is *off*, with the exception of **implicit-labelling** and **add**. The default value of **implicit-labelling** is *on*, if the interface is unclassified genser, and otherwise *off*. The default value for **add** is *on* if the interface is not unclassified genser, and otherwise *off*. Table 5 provides a list of all default values.

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

Table 5: Default Security Keyword Values

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

3.2.3 Sample IPSO Configurations

In this example, three Ethernet interfaces are presented. These interfaces are running at security levels of Classified Genser, Secret Genser, and Classified to Secret Genser. See Figure 1.

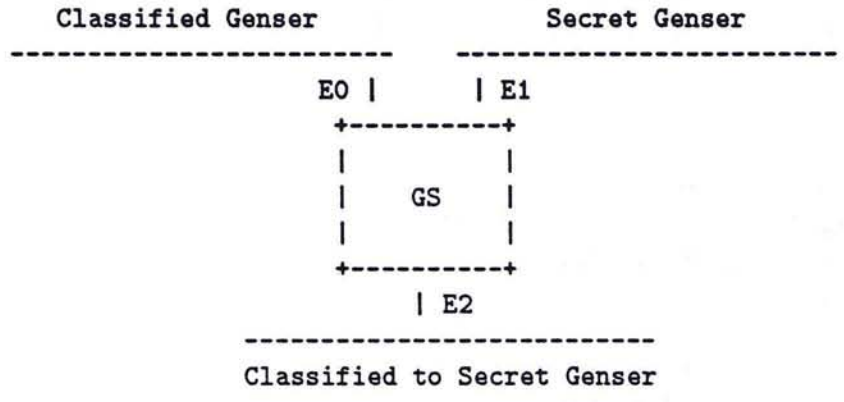


Figure 1: Security Levels

The following configuration commands are necessary to set up this situation:

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

It is possible for the setup to be much more complex. In this next example, there are devices on Ethernet 0 that cannot generate a security option, and so must accept datagrams without a security option. These hosts also crash when they receive a security option, therefore, never place one out on that interface. Furthermore, there are hosts on the other two networks that are using the extended security option to communicate information, so you must allow these to pass through the system. Finally, there is also a host on Ethernet 2 that requires the security option to be the first option present, and this condition must also be specified. The configuration now becomes:

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

3.2.4 IPSO Error Codes

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

Any security failure is also reported to the sending host when allowed by the specification. This calculation on whether to send an error message can be somewhat confusing. It depends upon both the security label in the datagram and the label of the incoming interface. First the label contained in the datagram is examined for anything obviously wrong, if not, it should be assumed to be correct. If there is something wrong, then the datagram should be treated as *unclassified genser*. Then this label is compared to the interface range, and the appropriate action is taken. See Table 6.

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

Table 6: Security Actions

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

- ICMP Parameter problem, code 0 — Error at pointer
- ICMP Parameter problem, code 1 — Missing option
- ICMP Parameter problem, code 2 — ** see text below
- ICMP Unreachable, code 10 — Administratively prohibited

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

3.3 IP Routing Features

This section discusses features available for IP routing. These features include:

- Enhancements to previous software
- EGP routing
- Fast re-routing

3.3.1 Enhancements

Passive routing has been implemented. If a network is flagged with the router subcommand **network** *address* **passive**, then that network number is included in all routing updates to other networks, but does not receive any periodic broadcasts. A finer degree of control can be achieved with the **router** subcommand **passive-interface** *interface* which disables the routing updates sent out a particular interface.

The **offset-list** subcommand of the **router** command can be used to add a positive offset to incoming and outgoing metrics for networks matching an access list.

offset-list *list* **in** | **out** *offset*

If *list* is zero, the *offset* is applied to all metrics. If *offset* is zero, no action is taken. For IGRP, the offset is added to the delay component only. This command is currently implemented for the RIP and HELLO routing protocols as well.

A default subnet can be specified for a network using the **default-network** *network* command. If the router is unable to route a datagram to a subnet of a network and a default subnet exists, then the datagram is delivered to that subnet (if directly connected) or to another gateway along the route to the default subnet. A default subnet is different than a default network; it applies only to subnets of a particular network. Suppose, for example, that network 131.108.0.0 was subnetted on the third octet and that subnet 45, or 131.108.45.0 was to be the default subnet. The command specifying the default would be "default-network 131.108.45.0". The **show ip route** *network* command displays the default subnet for *network*.

The **router** subcommand **neighbor** *address* defines a neighboring router with which to exchange routing information. For exterior routing protocols such as EGP and BGP, this command specifies routing peers. For normally broadcast protocols such as IGRP or RIP, this command permits the point-to-point (non-broadcast) exchange of routing information.

The handling of the default exterior route or gateway of last resort computation has been accelerated. This computation is now done once a minute or on demand and

reacts very quickly to topology changes. It is no longer required that a dynamic routing protocol be running. Care is now taken not to accept a default path to a gateway of last resort if the **cisco** gateway is sourcing a default path. A static route can be used to determine the gateway of last resort in this case.

The **router** subcommand **[no] default-information in|out** controls handling of default route information between multiple IGRP processes. The default is not to pass default route information between IGRP processes.

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

It is now possible for split-horizon partitioning to work correctly on non-broadcast media such as X.25 networks. In such situations routers are sending point-to-point updates to routers specified by the **neighbor** subcommand of the **router** configuration command. Split-horizon partitioning inhibits routers from advertising networks back to the routers from which the networks were originally learned. This facility is particularly useful for Bellman-Ford style routing protocols (IGRP, RIP, and HELLO).

The **router** subcommand **metric weights** *TOS K1 K2 K3 K4 K5* allows the tuning of the IGRP metric calculation for a particular Type of Service (TOS). The TOS parameter currently must always be zero. Parameters *K1* through *K5* are constants in the equation that converts an IGRP metric vector into a scalar quantity.

Software now allows network 0.0.0.0 to be a default network, but never allows it into the main IP routing table. This makes the **cisco** handling of RIP and HELLO defaults compatible with that of other RIP- and HELLO-speaking routers. It is not possible to redistribute network 0.0.0.0. The **distribute-list** mechanism can still be used to suppress information regarding network 0.0.0.0.

3.3.2 EGP Routing

This section describes the configuration of an EGP routing process. The procedure described in the *Gateway System Manual* is no longer valid. In particular, the **primary-neighbors** and **egp-neighbor** commands are no longer supported.

First you must specify an autonomous system for your router using the **autonomous-system** *local-AS* configuration command. *local-AS* is an autonomous system number

assigned by the Defense Data Network Information Center (NIC) at SRI, International in Menlo Park, CA. The *local-AS* number will be included in every EGP message sent by the router.

After the local AS number has been specified, an EGP routing process may be started with a **router** configuration command. For EGP, the command is **router egp** *remote-AS*. The remote AS number is the AS number the router expects its peers to be advertising in their EGP messages. The **cisco** software does not insist that the actual remote AS number match the configured remote AS numbers. The output from **debug egp** will advise of any discrepancies, however. Up to four different EGP routing processes may be created. An EGP routing process may be turned off with the **no router egp** *remote-AS* command.

The **router** subcommand **neighbor** *ip-address* is used to specify the IP address of an EGP peer.

The **router** subcommand **network** *network-number* is used to specify a network to be advertised to the EGP peers of an EGP routing process. Such networks are advertised with a distance of zero. There is no restriction on the network number other than the network must appear in the routing table. The network may be connected, may be statically configured, or may be learned from a dynamic routing protocol.

The following is an example of the configuration for an EGP router process. The router is an autonomous system 109 and is peering with routers in AS 164. We will advertise the networks 131.108.0.0 and 192.31.7.0 to two routers in AS 164, 10.2.0.2 and 10.3.0.11.

```
autonomous-system 109
router egp 164
network 131.108.0.0
network 192.31.7.0
neighbor 10.2.0.2
neighbor 10.3.0.11
```

3.3.3 Fast IP Re-routing

Several changes have been made to the IP routing support in the **cisco** software to enable faster convergence of the various IP routing algorithms and hence quicker fall-back to redundant routers. The total effect is to minimize disruptions to end users of the network in situations where quick recovery is of paramount importance.

Keepalive Timers

With Release 8.0 software, the network administrator can configure the keepalive interval, the frequency at which the **cisco** gateway sends messages to itself (Ethernet and Token Ring) or to the other end (Serial) to ensure a network interface is alive. The interval was 10 seconds; it is now adjustable in one second increments down to one second. An interface is declared down after two update intervals have passed without receiving a keepalive packet.

The new command syntax for the **keepalive interface** subcommand is:

keepalive [*seconds*]
no keepalive

If *seconds* is not specified, a default of 10 seconds is assumed. In the following example the keepalive interval is set to three seconds.

```
interface ethernet 0
keepalive 3
```

Setting the keepalive timer to a low value is very useful for rapidly detecting Ethernet interface failures (transceiver cable disconnecting, cable unterminated, etc.).

The typical serial line failures involve losing carrier detect. Since this sort of failure is typically noticed within a few milliseconds, adjusting the keepalive timer for quicker routing recovery is generally not useful. Caution should be taken when adjusting the keepalive timer for a very low bandwidth serial interface. It is possible to have large datagrams delay the smaller keepalive packets long enough to cause the line protocol to spuriously go down.

Adjustable Routing Timers

The basic timing parameters for IGRP, RIP, and HELLO are now adjustable. Since these routing protocols are executing a distributed, asynchronous routing algorithm, **it is important that these timers be the same for all routers in the network.** Some rather interesting "routing flapping" results if the timers are different. The new command is **timers** and is a **router** subcommand. It takes four parameters, each of which is in seconds and specifies a timer within the routing protocol, as follows:

timers basic *update invalid holddown flush*

update is the rate at which updates are sent. This is the fundamental timing parameter of the routing protocol. *invalid* is an interval of time after which a route is declared invalid; it should be three times the value of *update*. *holddown* is the interval during which routing information regarding better paths is suppressed. It should be at least three times the value of *update*. *flush* is the amount of time that must pass before the route is removed from routing table; it must be at least the sum of *invalid* and *holddown*.

The current and default timer values can be seen by inspecting the output of the EXEC **show ip protocols** command. It is strongly recommended that the relationships of the various timers be preserved as described above.

In the following example, updates are broadcast every 5 seconds. If a router is not heard from in 15 seconds, the route is declared unusable. Further information is suppressed for an additional 15 seconds. At the end of the suppression period the route is flushed from the routing table.

```
router igrp 109
timers basic 5  15  15  30
```

Note that by setting a short update period, you run the risk of congesting slow-speed serial lines; however, this is not much of a concern on faster-speed Ethernets and T1-rate serial lines. Also, if you have many routes in your updates, you can also cause the routers to spend an excessive amount of time processing updates. You can set these timers too short. Experimentation is recommended.

Gateway Discovery Protocol

A third enhancement for rapid re-routing is the introduction of a Gateway Discovery Protocol (GDP) in Release 8.0. This protocol allows hosts to dynamically detect the arrival of new routers as well as determine when a router goes down. Host software is needed to take advantage of this protocol. Unsupported, example GDP servers can be obtained from **cisco Systems**.

GDP is not a standard; it is a protocol designed by **cisco Systems** to meet the customers' needs. Work is in progress to establish GDP or something similar to GDP as a standard means of discovering routers. The current GDP implementation is described next in more detail.

Use of GDP is controlled by the **ip gdp** subcommand of the **interface** command. The following keywords of the **ip gdp** subcommand are supported:

[no] ip gdp
ip gdp priority *number*
ip gdp reporttime *seconds*
ip gdp holdtime *seconds*

The command **ip gdp** enables GDP processing on an interface. The default parameters are a reporting interval of five seconds for broadcast media such as Ethernets and 0 seconds (never) for non-broadcast media such as X.25, a priority of 100, and a hold time of 15 seconds. The **priority**, **reporttime**, and **holdtime** keywords are used to change those default values. When enabled on an interface, GDP updates report the primary and secondary IP addresses of that interface.

For ease of implementation on a variety of host software, GDP is based on the User Datagram Protocol (UDP). The UDP source and destination ports of GDP datagrams are both set to 1997 (decimal).

There are two types of GDP messages, REPORT and QUERY. On broadcast media REPORT packets are periodically sent to the IP broadcast address announcing that the router is present and functioning. By listening for these REPORT packets, a host can detect a vanishing or appearing router. If a host issues a QUERY packet to the broadcast address, the routers each respond with a REPORT sent to the host's IP address. On non-broadcast media, routers send REPORT messages only in response to QUERY messages. The protocol provides a mechanism for limiting the rate at which QUERY messages are sent on non-broadcast media.

Figure 2 shows the format of the GDP REPORT packet format. A GDP QUERY packet has a similar format, except that the Count field is always zero and no address information is present.

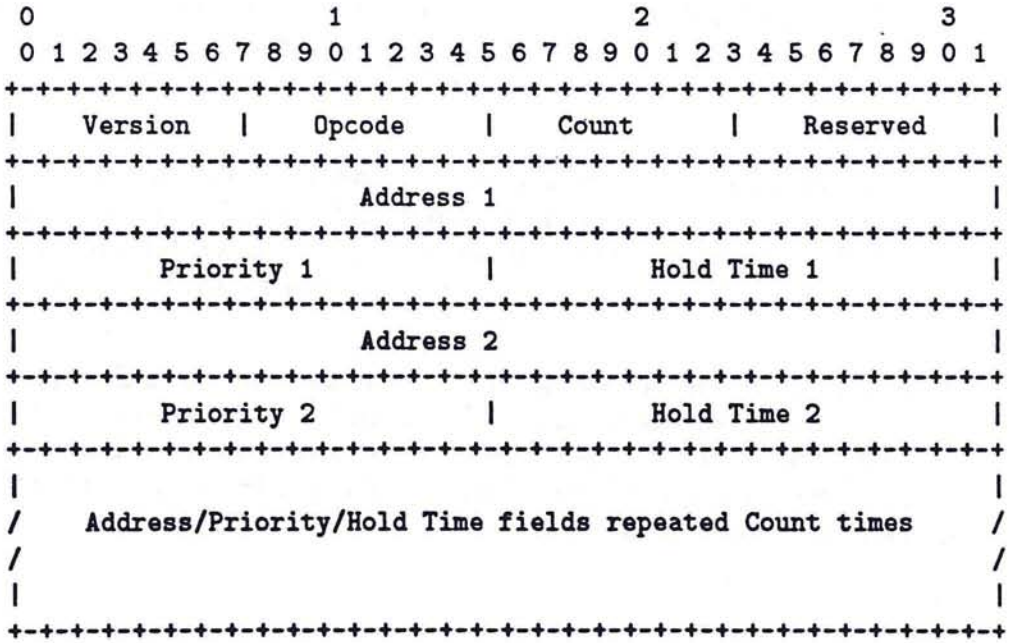


Figure 2: GDP REPORT Message Format

The fields in the REPORT and QUERY messages are described next.

- Version: 8-bits
  The Version field is 8-bits of protocol version number. The current GDP version number is 1. If an unrecognized version number is found, the GDP message must be ignored.

- Opcode: 8-bits
  The Opcode field describes the GDP message type. Unrecognized opcodes must be ignored. Opcode 1 is a REPORT message and opcode 2 is a QUERY message.

- Count: 8-bits
  The Count field contains the number of address, priority, and hold time tuples in this message. A QUERY message has a Count field value of zero. A REPORT message has a Count field value of 1 or greater.

- Reserved: 8-bits
  The Reserved field is reserved and must be set to zero.

- Address: 32-bits
  An IP address of a router on the local network segment. There are no other restrictions on this address. If a host encounters an address that it believes is not on its local network segment, then the host should quietly ignore that address.

- Priority: 16-bits
  The Priority field indicates the relative quality of the associated address. The numerically larger the value in the Priority Field, the *better* the address should be considered.

- Hold Time: 16-bits
  On broadcast media, the number of seconds the associated address should be used as a router without hearing further REPORT messages regarding that address. On non-broadcast media, such as X.25, this is the number of seconds the requestor should wait before sending another QUERY message.

Numerous actions can be taken by the host software listening to GDP packets. One very simple possibility is to flush the host's ARP cache whenever a router appears or disappears. A more complex possibility would be to update a host routing table based on the coming and going of routers. The particular course of action taken depends on the host software and the customer's requirements.

3.4 Border Gateway Protocol (BGP)

The Border Gateway Protocol, or BGP, is defined by RFC 1105. It is an experimental replacement for EGP, the Exterior Gateway Protocol introduced with software Release 8.0. Using BGP it is possible to set up a distributed routing core (as opposed to EGP's single routing core) that automatically guarantees the loop-free exchange of routing information on an Autonomous System (AS) or Routing Domain (RD) level. BGP also allows the possibility of policy based routing on a Routing Domain (RD) and network level.

As of 1 October 1989, BGP is considered an experimental protocol. BGP usage documents are being written as the possibilities of BGP are explored. The **cisco** BGP support is thus subject to change as BGP and inter-Routing Domain technology is advanced.

3.4.1 BGP Terminology

The following are terms used by the Interconnectivity Working Group of the IETF and are currently not defined in any issued RFC.

- *Routing Domain or RD* - A collection of networks and routers that are administered by a single political entity. RFC 1105 uses the term *Autonomous System* instead of *Routing Domain.*
- *Border Gateway or BG* - A router that passes traffic between Routing Domains. It is not required that the BG speak BGP.
- *Routing Server or RS* - A system that participates in the Border Gateway Protocol with other BGP-speaking entities. This system need not be a router.
- *colocated* - The situation where a BG and an RS are not "in the same box," but share a common (sub)network.
- *coresident* - The situation where the BG and RS are "in the same box." The **cisco** BGP implementation is coresident.
- *Interior Gateway Protocol or IGP* - Any interior routing protocol such as RIP, IGRP, or HELLO.
- *Transit RD* - A Routing Domain that passes along traffic not necessarily destined for the current RD between other RDs. The NSFnet and its regional networks are examples of transit RDs. A *multi-homed RD* could be a transit RD, but does not carry traffic for other RDs. The **cisco** BGP implementation can support a transit RD.
- *Stub RD* - A Routing Domain that cannot carry transit traffic. This is very close to the earlier EGP model of a stub exterior network. Stub RDs are also supported by the **cisco** BGP implementation.

3.4.2 Configuring and Monitoring BGP

BGP is configured in much the same way as EGP. The configuration command **router bgp** *routing-domain* is used to start the BGP-specific commands. The number used for the Routing Domain is taken from the Autonomous System number space. It is used to identify the gateway to other peers and to tag the routing information passed along. The **network** subcommand is used to specify those networks that are to be advertised as originating within the current RD. These networks can be learned from any source (connected, dynamic routing, static route, and so on).

The **neighbor** subcommand is used to identify BGP peers and their relative position in the hierarchy, to assign access lists, and to set up various per peer routing policies. The following is the list of BGP **neighbor** command extensions.

no neighbor *address*
neighbor *address* **up|down|hlink|internal**
[no] neighbor *address* **distribute-list** *list* **in|out**
[no] neighbor *address* **weight** *weight*
[no] neighbor *address* **rd** *number* **deny**
[no] neighbor *address* **rd** *number* **permit** *weight*

The only required **neighbor** subcommand is the one that specifies the direction of the link. The current direction keywords are **up**, **down**, **hlink**, and **internal**. The meaning of these directions is explained in RFC 1105. The **cisco** implementation can initiate or receive BGP connections. If the remote system is not a listed neighbor, the BGP connection is refused.

Access lists can be set up on a per peer basis to filter the network numbers sent or accepted. Networks can also be filtered out on a Routing Domain basis with a permit/deny mechanism. A **permit** assigns a weight to the specified RD. RDs that are not explicitly listed are given a default weight. These weights are summed up and the RD path with the greatest weight is used as the path to a particular network. A **deny** causes the information learned about that network using the specified RD to be ignored.

The following is an example of configuring a BGP router process.

```
router bgp 109
network 131.108.0.0
network 192.31.7.0
neighbor 131.108.200.1  internal
neighbor 131.108.240.67  up
```

The EXEC command **show bgp** displays the entire BGP routing table; **show bgp** *network* displays a particular network in the BGP routing table. Entries in the main routing table (**show ip route**) entered by BGP are marked with a 'B'. An asterisk (*) in front of an entry in the **show bgp** display indicates that the entry is "valid," that is, a route to that network currently exists. The EXEC command **show bgp neighbors** displays detailed information on the TCP and BGP connections to individual neighbors. The **debug ip-bgp** command displays some debugging information on BGP transactions.

3.4.3 BGP and Transit Routing Domains

This section explains the interaction of BGP and IGP routing information in systems where the Border Gateway (BG) and the Routing Server (RS) co-reside. The explanation assumes that there are separate databases or routing tables for BGP and IGP information and that the exchange of information between these databases can be controlled.

The network administrator manually specifies a list of networks in the IGP database as candidates for inclusion in the BGP database. These networks (referred to hereafter as local networks) are known a priori to originate in the RD currently being described. If a local network appears in or disappears from the IGP routing database, it is simultaneously entered into or removed from the BGP database.

To prevent routing information loops, BGP must never be a source of reachability information for a local network. Two things must be done to prevent these loops. First, a local network is never entered into the IGP database from the BGP database. Second, to prevent the IGP from putting a BGP-derived network into the IGP database, the software checks for that network in the BGP database. If that network is known via BGP and it was not one of the local networks, then the route to that network is ignored; it is an external network that is BGP-derived.

The case of a stub RD is relatively easy: The BG advertises the RD's networks to the other BGP speakers in other RDs. The IGP would need to carry at most a default path pointing to the BG.

The case of transit and multi-homed RDs is trickier. Here it is assumed that the IGP is carrying information derived from BGP information. To ensure that the IGP information is sourced by the correct set of exit BGs, each BG takes care to re-advertise via the IGP only those non-local networks learned by non-internal BGP links. Therefore, if a network is learned from an internal link, the peer at the other end of that link is an appropriate exit BG.

If multiple BGs have routes to a particular network, then some sort of common policy must be implemented in the RD to select the "best" set of exit BGs. There is nothing to prevent multiple BGs from serving as exit gateways for the same network; however, the BGs should be coordinating their decisions to ensure a stable,

predictable result. With **cisco** software, only a single exit gateway is chosen using the following heuristic methods: The shortest AS path is preferred. If there are AS paths with the same length, then the BG whose IP address is numerically greatest is chosen as the exit BG for that network. Note that many other heuristics or policies can be chosen; the one described is merely a reasonable factory default.

Assume that because of external events, a different BG within the RD becomes the exit gateway for a particular network. The rapidity with which the IGP can handle the topology recomputation determines the length of any unreachability events. A simple-minded RIP implementation would probably have intolerably long unreachability intervals. More sophisticated IGPs, including **cisco's** IGRP, would be able to keep any unreachability intervals very short.

The entire catenet need not be injected into every transit RD. One or more of the BGs can inject a default route into the IGP to siphon off otherwise unroutable traffic.

3.4.4 Examples of BGP Usage

This section describes some BGP usage scenarios.

Stub RD — Replacement for EGP

Acting as a replacement for EGP in a stub RD is very simple. Consider the case of a regional network attaching to the NSFnet at a Nodal Switching System (NSS). The peer specified is the NSS. The direction is usually **up**, although the actual bilateral agreement between the NSFnet and the regional network can be any direction.

The networks specified by the **network** command are all the networks originating in the regional network. These networks can be learned via any means. Once they are present in the routing table, they will be advertised to the NSS as originating in the stub RD. Since BGP has a third-party facility, the BG can indicate to the NSS that other, non-BG routers on the shared (sub)network are the proper first hop paths for certain networks.

Transit RD — The Two Coast Problem

Consider a geographically dispersed group of networks that have two widely separated points of connection to a transit RD. It is desired that traffic destined for networks close to a point of attachment use the router at that point, rather than going through the more distant router. It is further desired that traffic use the distant router if the near router is unavailable.

For a concrete example, assume the transit RD is the NSFnet. Split the group of networks into two RDs, one called East and one called West, each clustered around

a BG onto the NSFnet. None of the networks in the East and West RDs are in both RDs.

In the simplest case there are four BGP-speaking routers in this setup, two BGs at the East and West NSFnet borders and two BGs at the border between East and West. Let us call these BG-speaking routers West/NSFnet, West/East, East/West, and East/NSFnet.

This example describes an RD setup for the West RD. The setup is symmetric for the East RD and all the same arguments can be made.

The West/NSFnet BG advertises to the NSFnet all networks in the West RD with an RD path of length one and all networks in the East RD with an RD path of length two. This same BG also sources a very high priority default into the IGP. On the other side of the West RD, the West/East BG redistributes information about the networks in the East RD into the West IGP. This same BG sources a lower priority default into the BG. This lower priority default has its metric set such that it is never chosen if the West/NSFnet default is present.

The only networks communicated between the BGP speakers of the East and West RDs are the networks belonging to those two RDs. The use of defaults as described above removes the necessity for passing around information regarding other networks.

In the case where all the BGs are functioning, traffic flows as desired. Since the NSFnet (probably) favors shorter RD paths, datagrams destined for the West RD flow through the West/NSFnet BG. Outgoing traffic in the West RD use the default path sourced by the West/NSFnet BG. Datagrams destined for the East RD flow through the West/East BG.

In the case where the West/NSFnet BG goes down, the NSFnet would fall back to the longer path sourced by the East/NSFnet BG to reach networks in the West RD. Outgoing traffic in the West RD would use the lower priority default being generated by the West/East BG. Datagrams destined for the East RD would continue to use the West/East BG as before.

In the case where connectivity to the East RD fails, information about the networks in the East RD would fade from the West IGP. The default path to the West/NSFnet BG would then take up the traffic and route it over the NSFnet to the East RD.

The major contribution of BGP in this scenario is letting the NSFnet know which paths to East and West are more desirable, which in this case means the shorter RD path.

3.5 DECnet Routing

This section describes enhancements to and new commands for DECnet routing, as part of Releases 7.0, 7.1 and 8.0.

3.5.1 DECnet Enhancements

Changes to the commands that support DECnet follow:

- The new command for enabling DECnet routing is

  [no] decnet routing *decnet-address*

  The software continues to parse earlier commands that used the **address** instead of **routing** keyword, but writes the new form to non-volatile memory. This makes the syntax for enabling a routing protocol consistent across all protocols.

- The **show decnet** command was extended as follows:

  show decnet traffic
  show decnet interface *name*
  show decnet route

- The fast switching of DECnet datagrams by MCI network interfaces was implemented in Release 7.1. The **interface** configuration subcommand

  [no] decnet route-cache

  controls DECnet fast switching. The default is to do DECnet fast switching.

- It is advisable to disable fast switching on slow-speed serial links running at speeds of 19.2 kilobits or slower. This allows the main system to handle congestion problems by buffering more datagrams.

- DECnet access lists are now executed when fast switching is enabled. These access lists no longer have a limit to the number of entries that can be specified.

3.5.2 DECnet Access Lists

Access lists were added to the **cisco** DECnet support in Release 7.0. In the following explanation, it is assumed the reader is familiar with IP access lists as described in the *Gateway System Manual* dated July 1988.

There are two forms of DECnet access lists, one that specifies a single address and one that specifies two addresses. The occasions for use of each form are described next in the paragraphs on **decnet** configuration commands.

access-list *list* **permit|deny** *address mask*
access-list *list* **permit|deny** *source srcmask dest dstmask*

The *list* argument is a customer-selected integer between 300 and 399 that uniquely identifies the access list. The **permit** and **deny** keywords decide the access control action when a match happens with the address arguments.

The single address form of the DECnet access list has a DECnet address followed by a mask, also in DECnet address format, with bits set wherever the corresponding bits in the address should be ignored. DECnet addresses are written in the form *area.node* (e.g., 50.4 is node 4 in area 50). All addresses and masks are in decimal.

The double address form of the DECnet access list has a source DECnet address and mask pair followed by a destination DECnet address and mask pair.

The following keywords and arguments have been added to the **decnet** subcommand of the **interface** configuration command.

decnet access-group *list*

The command **decnet access-group** sets up access control for packets sent out this interface. The access *list* can be either a single or double address DECnet access list. A single address DECnet access list applies to destination addresses in this case.

decnet in-routing-filter *list*

The command **decnet in-routing-filter** provides access control to HELLOs or routing information received on this interface. Addresses that fail this test are treated as unreachable. The access *list* is a single address DECnet access list.

decnet out-routing-filter *list*

The command **decnet out-routing-filter** provides access control to routing information being sent out this interface. Addresses that fail this test are shown in the update message as unreachable. The access *list* is a single address DECnet access list.

3.6 XNS Routing

Beginning with Release 7.1, **cisco** software has supported routing of the XNS protocol in the gateway server products. XNS routing can take place concurrently with any other protocol routing, for example, Internet or DECnet. A minimal set of XNS services have been implemented to support XNS routing.

XNS can be routed across Ethernets, Token Rings or point-to-point serial lines running HDLC, LAPB, or X.25. The XNS RIP protocol is supported for maintaining routing databases across the XNS internet. The XNS Echo protocol is provided via a user interface for debugging purposes and as a server for remote diagnostics.

3.6.1 XNS Configuration Commands

XNS is configured using the **xns** configuration command. This configuration command is global for system-wide XNS parameters and is an **interface** subcommand for interface-specific parameters.

[no] xns routing [*address*]

This global configuration command enables XNS processing. The optional *address* argument is a 48-bit value written as a dotted triple of hexadecimal digits, for example, "0123.4567.abcd". If *address* is omitted, the Ethernet address of the first Ethernet interface found is used. The command **no xns routing** disables all XNS datagram processing.

[no] xns route *network host-address*

This global configuration command adds a static route into the XNS routing table. *network* is the destination XNS network number in decimal. *host-address* is a decimal XNS network number and host number written in the following form. If the host number was "0456.acd3.1243" and the network number was "91", *host-address* would be written as "91.0456.acd3.1243".

xns network *number*

This interface-specific configuration command assigns a decimal XNS network number to an interface and enables that interface to run XNS protocols. Interfaces not enabled to run XNS ignore any XNS datagrams that they receive. Every interface in a system that supports XNS must have a unique XNS network number.

[no] xns helper-address *host-address*

Some XNS protocols use the all-nets broadcast (network number of all ones). The **xns helper-address** command is an interface-specific command that causes such all-nets broadcasts to be forwarded to *host-address*. The *host-address* argument is a dotted combination of the network and host addresses as explained in the **xns route** command.

[no] xns forward-protocol *type*

The **interface** subcommand **xns forward-protocol** allows you to specify XNS protocol types that will be forwarded when a broadcast is received on an interface for which an XNS helper address is set. Broadcasts to all networks (network number of all ones) are forwarded regardless of protocol. Broadcasts to the local network (network all zeros) will be ignored. The *type* argument is a decimal number corresponding to an appropriate XNS protocol. See the documentation accompanying your XNS implementation to determine this number.

[no] xns encapsulation *type*

Because there is no agreed upon mechanism for encapsulating XNS packets on a Token Ring network, many vendors have invented their own encapsulation methods. Encapsulation refers to the kind of envelope in which the XNS packet is wrapped prior to being transmitted.

Use the **interface** subcommand

xns encapsulation *keyword*

to select the encapsulation used on a Token Ring interface. The default *keyword* is **snap**. Other possibilities are **ub** (for Ungermann-Bass) and **3com** (for the 3COM Corporation product).

3.6.2 XNS EXEC Commands

The EXEC command **show xns** supports the following keywords:

show xns interface [*name*]
show xns route [*network*]
show xns traffic

The command **show xns interface** shows interface-specific XNS parameters. **show xns route** displays the entire XNS routing table, or if a network number is specified, a specific network entry. Packet handling statistics are displayed by **show xns traffic**.

The **debug xns-packet** command displays XNS datagram traffic, including the source, destination, and next hop gateway of each datagram.

The **debug xns-rip** command displays XNS routing transactions.

The **ping** command understands the XNS echo protocol and can be used to bounce XNS echo packets off of other XNS hosts. You must have XNS routing enabled to use this function.

3.6.3 XNS Access Lists

XNS access lists have been added for Release 8.0. Two types of access list are available. Simple XNS access lists are numbered from 400 to 499 and filter on the source and destination addresses only. Extended XNS access lists are numbered from 500 to 599 and filter on XNS protocol and socket fields. An XNS access list is assigned with the **interface** subcommand

[no] xns access-group *number*

The command syntax for standard XNS access lists is lengthy. For typographic reasons the command example is shown on multiple lines; it must be on a single line when given as a configuration command.

access-list *number* **deny|permit**
XNS-source-network[*.source-address*] *source-mask*
XNS-destination-network.destination-address destination-mask

The only required parameter for standard XNS access lists is the XNS source network. The rest of the parameters are optional except that the source and/or destination address masks are present only if the corresponding source and/or destination address was entered. The following are some examples of standard XNS access list usage.

```
access-list 400 deny -1 2
```

The above example denies access from source network –1 (all XNS networks) to destination network 2.

```
access-list 400 deny -1.0000.0c00.1111
```

The above example denies access from XNS source address 0000.0c00.1111. The source network does not make any difference.

```
access-list 400 deny 1.0000.0c00.1111 0000.00ff.ffff
```

The above example denies access from all hosts on network 1 that have a source address beginning with 0000.0c.

```
access-list 400 deny 1.1111.1111.1111 0000.0000.0000 2.2222.2222.2222
                     0000.0000.0000
```

The above example denies access from source address 1111.1111.1111 on network 1 to destination address 2222.2222.2222 on network 2

For extended access lists the command syntax is again rather lengthy as a configuration command.

access-list *number* **deny | permit** *xns-protocol*
source-network.source-address source-mask source-socket
destination-network.destination-address destination-mask destination-socket

The source and destination addresses and masks are optional. The protocol number *xns-protocol* is the only required parameter.

```
access-list 500 deny 1 1 1234 2 1234
```

The above denies access to protocol 1 from source network 1, source socket 1234 to destination network 2, destination socket 1234

```
access-list 500 deny 1 1.1111.1111.1111 0000.0000.0000 1234
                       2.2222.2222.2222 0000.0000.0000 1234
```

The above illustrates the use of all possible parameters.

3.7 AppleTalk Routing

Release 7.1 supports the routing of Apple Computer's AppleTalk network protocol over Ethernet, synchronous serial, Token Ring, and X.25 interfaces. LocalTalk (AppleTalk on Apple's proprietary local area network) interfaces are not supported.

The following services are provided by the **cisco** AppleTalk implementation:

- Routing Table Management Protocol (RTMP)
- Name Binding Protocol (NBP)
- Echo Protocol (EP)
- AppleTalk Transaction Protocol (ATP)
- Zone Information Protocol (ZIP)

In Apple literature describing AppleTalk routing, the device doing the routing of AppleTalk is described as a "bridge." **cisco** documentation uses that term to describe a MAC-level switching device and the term "router" to describe a network-level switching device. The term router is used in this document.

3.7.1 AppleTalk Interoperability Issues

The **cisco** AppleTalk software has been tested and verified to work with Apple Macintosh products, the Apple LaserWriter, Centram Systems' TOPS (both Sun and Macintosh), Infosphere's Liaison, and the Kinetics FastPath series.

Earlier versions of the Kinetics software have problems with checksumming errors. This forces the inclusion of a configuration command to disable AppleTalk checksumming. Checksumming is on by default; it can be disabled by giving the global configuration command **no appletalk checksum**.

3.7.2 Setting Up AppleTalk Routing

To set up AppleTalk routing, you need to understand the **cisco** representation of AppleTalk addresses. These addresses are 24-bits long. They consist of a 16-bit network number and an 8-bit node number. The **cisco** AppleTalk software parses and displays these addresses as a sequence of two decimal numbers, first the network, then the host, separated by dots. For example, host 45 on network 3 is written as `3.45`.

The first step is to give the configuration command

appletalk routing

to enable AppleTalk protocol processing. The command **no appletalk routing** disables AppleTalk processing.

Next you need to assign AppleTalk addresses on the interfaces that will be handling the AppleTalk protocol. This is done with the **interface** subcommand

appletalk address *address*

Not all fields of *address* need to be specified. If there is another AppleTalk router on the network, it may be able to supply you with the network number. Node numbers can be negotiated between AppleTalk hosts on an Ethernet.

Unspecified parts of the AppleTalk address are zero, so `34.5` represents a fully-qualified address (net 34, node 5), `0.5` is a partially-qualified address (net unspecified, node 5), `122.0` represents net 122, node unspecified, and `0.0` is completely unspecified. Node numbers automatically assigned by AppleTalk begin at 128. The AppleTalk software will choose another node number if the specified node number is already in use.

You may wish to assign a zone name to a network if there is not another router on the network to provide that information. The **interface** subcommand

appletalk zone *zonename*

sets the zone name for the connected network.

Here is a sample configuration for routing between two Ethernets. Ethernet 0 is on network 1, at node 128. Ethernet 1 is on network 2, at node 154. The two networks are in the "Twilight" and "No Parking" zones, respectively.

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

Here is a variant of the above configuration, except that it has other routers on both networks to provide the zone and network number information. Therefore, software can let the **cisco** router discover the information dynamically.

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

3.7.3 AppleTalk Configuration Command Summary

The general use of the following configuration commands is described in the *Gateway System Manual* dated July 1988.

Following are the top-level configuration commands used to set up AppleTalk routing parameters before routing starts.

appletalk arp interval *milliseconds*
appletalk arp retransmit-count *count*
[no] appletalk checksum
[no] appletalk routing
[no] appletalk route *net gateway*

The following commands are subcommands of the **interface** configuration command. They are used to set interface-specific AppleTalk parameters.

[no] appletalk address *address*
[no] appletalk zone *zonename*
[no] appletalk access-group *list*

3.7.4 AppleTalk Access Lists

Access lists controlling AppleTalk switching by network number have been added. The format of an AppleTalk network access list is

access-list *list* **permit|deny** *network*

where *list* is an integer between 600 and 699 and *network* is an AppleTalk network. A *network* of –1 represents any network.

An AppleTalk access list is assigned to an interface with the **interface** subcommand

appletalk access-group *list.*

The EXEC command

show appletalk traffic

displays the number of packets dropped because of access control.

3.7.5 Monitoring AppleTalk

The EXEC monitoring commands for AppleTalk are explained next.

show apple interface [*interface*]

The **show apple interface** command displays the AppleTalk-specific interface information. An optional interface type can be specified with the *interface* argument. A sample display follows:

```
gorky>show ap int

Ethernet 0 is up, line protocol is up
  AppleTalk address is 258.128, Valid
  AppleTalk zone is Twilight
 --More--
Serial 0 is administratively down, line protocol is down
  AppleTalk protocol processing disabled
 --More--
Ethernet 1 is up, line protocol is up
  AppleTalk address is 11.128, Valid
  AppleTalk zone is No Parking
 --More--
Serial 1 is administratively down, line protocol is down
  AppleTalk protocol processing disabled
```

show arp

The **show arp** command displays the contents of the ARP cache, including the AppleTalk entries. A sample display follows:

```
gorky>show arp

Protocol  Address          Idle (min)     Hardware Addr   Type    Interface
Internet  4.4.4.4                -       0000.0C00.02A0  ARPA    Ethernet1
AppleTalk 258.69               0         0260.8C56.1844  ARPA    Ethernet0
AppleTalk 258.128              -         0000.0C00.029F  ARPA    Ethernet0
AppleTalk 11.128               -         0000.0C00.02A0  ARPA    Ethernet1
AppleTalk 258.179              0         0800.89A0.0033  ARPA    Ethernet0
Internet  131.108.2.91         0         0800.2000.72F1  ARPA    Ethernet0
Internet  131.108.2.85         -         0000.0C00.029F  ARPA    Ethernet0
```

show apple route [*network*]

The **show apple route** command displays either the full routing table or just the entry for the optionally specified *network*. A sample display follows:

```
gorky>show apple route

Codes: R - RTMP derived, C - connected, S - static, 3 routes

C Net 258 directly connected, 1431 uses, Ethernet0, zone Twilight
R Net 6 [1/G] via 258.179, 8 sec, 0 uses, Ethernet0, zone The 0
C Net 11 directly connected, 472 uses, Ethernet1, zone No Parking
R Net 2154 [1/G] via 258.179, 8 sec, 6892 uses, Ethernet0, zone LocalTalk
S Net 1111 via 258.144, 0 uses, Ethernet0, no zone set
```

show apple zones

The EXEC command **show apple zones** displays the zone name table and show which networks comprise each zone. A sample display follows:

```
gorky>show ap zone
Name                                    Network(s)
EtherTalk                               2
Twilight                                258 4 1544
LocalTalk                               2154
No Parking                              11
Total of 4 zones
gorky>
```

Use the EXEC command **show apple traffic** to display AppleTalk-specific traffic statistics.

The EXEC debugging commands for monitoring the AppleTalk protocol are described, next.

debug apple-packet

The **debug apple-packet** command enables per packet debugging output. It is roughly equivalent to turning on all the other AppleTalk debugging information. There will be at least one line of debugging output per AppleTalk packet processed.

debug apple-nbp

The **debug apple-nbp** command enables debugging output from the Name-Binding Protocol routines.

debug apple-routing

The **debug apple-routing** command enables debugging output from the Routing Table Maintenance Protocol routines. It can be used to monitor acquisition of routes, aging of routing table entries, and advertisement of known routes. It also reports conflicting network numbers on the same network if the network is misconfigured.

debug apple-zip

The **debug apple-zip** command enables debugging output from the Zone Information Protocol routines. It reports significant events such as discovery of new zones, zone list queries, etc.

ping

The **ping** command has been extended to send Echo Protocol datagrams to other AppleTalk nodes to verify connectivity and measure round-trip times. When the **ping** command prompts for a protocol, specify **apple**.

3.8 ISO CLNS

Release 8.0 includes support for routing the ISO Connectionless Network Services (CLNS) as defined by ISO 8473 and the End System to Intermediate System (ES-IS) routing protocol defined by ISO 9542. The explanations in this section assume familiarity with those documents.

The **cisco** CLNS implementation is compliant with the Government Open Systems Interconnection Profile (GOSIP) and is compatible with DECnet Phase V.

This section notes a number of CLNS and ES-IS implementation details.

3.8.1 Supported Media

- *Ethernet*: Ethernet or IEEE 802.3 electrical levels can be used; however, the packet format is always 802.3/802.2. The ES-IS "All End Systems" and "All Intermediate Systems" multi-cast addresses are used.
- *Token Ring 802.5*: The ES-IS implementation uses functional addresses rather than multi-casts (group addresses) for ESHs and ISHs. The addresses used are c000.0008.0000 for "All End Systems" and c000.0010.0000 for "All Intermediate Systems."
- *Serial interfaces* up to T1 rate: **cisco's** HDLC encapsulation is used, therefore there must be a **cisco** router at both ends of the line. This encapsulation is the most efficient in terms of CPU and line utilization.
- *X.25 interfaces*: These require the manual entry of NSAP-to-X.121 mapping entries and interoperate with other implementations.

3.8.2 End System to Intermediate System Routing

- Both ESHs and ISHs are received and processed by a router.
- The configuration timer (rate at which ESHs and ISHs are sent) is a system-wide parameter. The default is 60 seconds.
- The holding timer for ESHs and ISHs is also a system-wide parameter. The default is five minutes.
- The **debug clns-esis-packet** and **debug clns-esis-event** commands report all ESHs and ISHs sent and received, and report when ESH or ISH information is aged out due to holding timer expiration.
- ES-IS information can be manually entered or deleted.
- Multiple NSAPs will be sent in one ESH if multiple NSAPs have been assigned to the router.

- The optimization described in Section 6.7 and Annex B, section 3 of the ISO 9542 standard is implemented with care to avoid packet flurries. When a new ESH or ISH is received, the router sends out a "bonus" ESH or ISH in unicast, not multi-cast packets. This is done only if the packet is received as a multi-cast.
- When a new NSAP is configured in the router, "bonus" ESHs or ISHs are generated.
- ESHs and ISHs are sent out without options. Options are ignored in received ESHs and ISHs.
- RDPDUs are sent with address masks. These masks can be optionally disabled. The address mask contains one bit only for each octet that was specified in the static route entry. SNPA masks are never sent.
- RDPDUs received with address masks are honored when the **cisco** software is acting as an ES.
- When operating as an IS, RDPDUs are ignored as per the 9542 ES-IS specification.
- The suggested End System Configuration Timer (ESCT) messages are not sent.

3.8.3 Packet Forwarding

CLNS does the following:

- Normally generates checksums. An interface can be optionally configured to substitute a null checksum.
- Always decrements the packet lifetime by one.
- Generates error Report PDUs (ERPDUs) for a variety of causes. ERPDU generation can be disabled on a per interface basis. It is rate-limited, with a default value of one per interface, per 10 milliseconds.
- Generates redirect PDUs when an IS forwards a packet out the same interface it was received on. RDPDU generation can be disabled on a per interface basis. It is also rate limited, with a default value of one per interface, per 100 milliseconds. The address mask option is present on all RDPDUs sent, unless disabled.
- Supports the padding option and the congestion experienced Global QOS option. The congestion threshold is a per interface parameter.
- Ignores the Record Route option, Source Route option, and the QOS (other than congestion experienced) options. The Security option causes a packet to be rejected with a bad option indication.

3.8.4 The CLNS Routing Table

Only static and ES-IS routing are currently available with the **cisco** CLNS implementation. Dynamic routing will be available in future releases. ES-IS supplies enough information to automatically route between End Systems (hosts) on networks directly connected to the Intermediate System (router). Thus any configuration involving only one router is trivially handled by ES-IS alone.

Configurations with more than one router require static routes, which must be entered manually using the configuration mechanism. A consequence of static routing is that there is no automatic routing around failed links and no load sharing.

The static routes can be configured to create a hierarchical network, with an arbitrary division of hierarchies. Routes can be created which direct all NSAPs for a given AFI, or AFI+Org-ID, or AFI+Org-ID+Area, or complete NSAP. These examples are vaguely analogous to TCP/IP's Autonomous System routes, net routes, subnet routes, and host routes, but since the mechanism does not use any a priori knowledge of the structure of NSAPs, other hierarchies can be used if desirable.

Routes are entered by specifying pairs (NSAP-prefix, next-hop-NET). NET is a Network Entity Title. It can be thought of as an NSAP. When a datagram is routed, the routing table is examined and the best match (longest NSAP-prefix entry which matches the beginning of the destination NSAP) is used. In the following static routing table the next-hop-NETs are listed for completeness, but are not necessary to understand the routing algorithm. Table 8 offers examples of how the routing entries in Table 7 can be applied to various NSAPs.

Entry #	*NSAP Prefix*	*Next Hop NET*
1	47.0005.000c.0001	47.0005.000c.0001.0000.1234.01
2	47.0004	47.0005.000c.0002.0000.0231.01
3	47.0005.0003	47.0005.000c.0001.0000.1234.01
4	47.0005.000c	47.0005.000c.0004.0000.0011.01
5	47.0005	47.0005.000c.0002.0000.0231.01

Table 7: Sample Routing Table Entries

Octet boundaries must be used for the internal boundaries of NSAPs and NETs. Even though a route has been configured, it is not usable until an ISH from the next-hop-NET is received.

3.8.5 CLNS Command Notation

The following subsections describe the various configuration and EXEC commands relating to CLNS. Use the following description to understand the notation used.

Datagram Destination NSAP	*Table entry # used*
47.0005.000c.0001.0000.3456.01	1
47.0005.000c.0001.6789.2345.01	1
47.0004.1234.1234.1234.1234.01	2
47.0005.0003.4321.4321.4321.01	3
47.0005.000c.0004.5678.5678.01	4
47.0005.000c.0005.3456.3456.01	4
47.0005.0023.9876.9876.9876.01	5

Table 8: Hierarchical Routing Examples

```
<nsap-prefix>    ::= <nsap>

<next-hop-nsap> ::= <nsap>

<nsap>           ::= xx.xxxx...xx     (up to 40 hex digits separated by .'s)
                     xxxxxx...xx      (up to 40 hex digits)


<snpa>           ::= xxxx.xxxx.xxxx  12 hex digits for Ethernet
                     ddddddddddddd   14 decimal digits for X.25
```

3.8.6 CLNS Global Configuration Commands

These are the CLNS configuration commands that apply to an entire system.

[no] clns configuration-time *seconds*

The **[no] clns configuration-time** command specifies the number of seconds between sending ESHs or ISHs. The default value for *seconds* is 60.

[no] clns holding-time *seconds*

The **[no] clns holding-time** command specifies the number of seconds of holding time transmitted in ESH and ISH packets. The default value for *seconds* is 300.

[no] clns nsap *nsap*

The **[no] clns nsap** *nsap* command specifies how many or adds an NSAP. A system can have more than one NSAP. All NSAPs will be included in the ESHs sent.

[no] clns packet-lifetime *number*

The **[no] clns packet-lifetime** command specifies the initial lifetime for locally generated packets. The default value for *number* is 64.

[no] clns routing

The **clns routing** command enables CLNS operation for the Network Entity Title (NET) specified by the **clns net** *nsap* command. This command configures the router as an Intermediate System (IS). Only one NET can be specified. If the command is issued twice, the second NET overrides the first. **no clns routing** disables CLNS IS functions on all interfaces.

[no] clns NET *nsap*

The **clns NET** command specifies or changes the NET. Only one NET can be specified. If the command is issued twice, the second NET overrides the first. This command configures the router as an End System (ES).

[no] clns route *nsap-prefix next-hop-nsap*

The **[no] clns route** command enters or deletes a static route. NSAPs that start with *nsap-prefix* are forwarded to *next-hop-nsap.*

[no] clns route *nsap-prefix* **discard**

The **clns route discard** command explicitly tells a CLNS IS node to discard packets with the specified *nsap-prefix.*

[no] clns want-erpdu

The **[no] clns want-erpdu** command specifies whether to request error PDUs on packets sourced by the router. The default is to request error PDUs.

3.8.7 CLNS Interface Configuration Commands

The following are subcommands of the **interface** configuration command. They specify interface-specific CLNS parameters.

[no] clns checksum

The **clns checksum** command enables checksum generation (default). The optional **no** keyword disables checksum generation.

[no] clns congestion-threshold *number*

The **clns congestion-threshold** command causes the system to set the "congestion experienced" bit if the output queue has more than *number* packets on it. A *number* value of zero or the **no** keyword prevents this bit from being set. The default value for *number* is 4.

[no] clns enable

The **clns enable** command enables CLNS processing on the interface. The **no** keyword disables processing (the default).

[no] clns mtu *mtu*

The **clns mtu** command sets the maximum CLNP packet size. This size includes the CLNS header and data portion of the packet. The default MTU is media-dependent. The **no** keyword restores the default.

[no] clns send-rdpdu

The **clns send-rdpdu** command controls the sending of Redirect PDUs. The **no** keyword disables this function.

[no] clns rdpdu-interval *milliseconds*

The **clns rdpdu-interval** command determines the minimum interval time, in milliseconds, between RDPDUs. A *millisecond* value of zero or the **no** keyword turns off the interval rate and effectively sets no limit to the RDPDU rate. The default rate is once every 100 milliseconds.

[no] clns rdpdu-mask

The **clns rdpdu-mask** command sets software to send address masks on RDPDUs. The **no** keyword disables this function. This parameter is enabled by default.

[no] clns send-erpdu

The **[no] clns send-erpdu** command sets software to send Error PDUs. The **no** keyword disables this function. This parameter is enabled by default.

[no] clns erpdu-interval *milliseconds*

The **clns erpdu-interval** command determines the minimum interval time, in milliseconds, between ERPDUs. A *millisecond* value of zero or the **no** keyword turns off the interval rate and effectively sets no limit to the ERDPDU rate. The default rate is once every 10 milliseconds.

[no] clns es-neighbor *nsap snpa*

The **clns es-neighbor** command enters an ES neighbor (static ES-IS entry). The argument *nsap* specifies the CLNS address. The argument *snpa* specifies the Level 2 address. The **no** keyword deletes the ES neighbor.

[no] clns is-neighbor *nsap snpa*

[no] clns is-neighbor *nsap snpa* enters an IS neighbor (static ES-IS entry). The argument *nsap* specifies the CLNS address. The argument *snpa* specifies the Level 2 address. The **no** keyword deletes the IS neighbor.

3.8.8 CLNS EXEC Show Commands

Release 8.0 of the software also includes monitoring commands for the CLNS ISO protocol.

show clns

The **show clns** command displays basic CLNS information such as the NET, if present, the configuration and holding times, and the initial packet lifetime for locally generated packets.

show clns cache

The **show clns cache** command displays the CLNS routing cache derived from the CLNS routing table.

show clns traffic

The **show clns traffic** command reports counters for types of packets received, transmitted, discarded, and so forth.

show clns interface [*interface*]

The **show clns interface** command displays the CLNS parameters of the specified *interface*, or all interfaces if no interface name was specified.

show clns neighbors [es|is]

The **show clns neighbors** command displays neighbors learned from ESHs (**es** keyword) and ISHs (**is** keyword).

show clns redirects

The **show clns redirects** command displays CLNS redirect information. Only ESs maintain redirect information.

show clns route [*nsap*]

The **show clns route** command displays routing information. The argument *nsap* specifies the CLNS address.

3.8.9 CLNS EXEC Debug Commands

The ISO CLNS software includes an extensive list of debugging commands, as described next.

debug clns-packets

The **debug clns-packets** command causes all CLNS activity to be traced, including forwarding of packets.

debug clns-events

The **debug clns-events** command traces "interesting" CLNS events, including packet discards, sending of redirects, and so forth.

debug clns-esis-packets

The **debug clns-esis-packets** command traces ES-IS activity, including sending and receiving of ESHs and ISHs, receiving Redirects (RDs), and aging out of ESH/ISH/RD entries.

debug clns-esis-events

The **debug clns-esis-events** command traces "interesting" ES-IS events, including previously unknown neighbors, neighbors which have aged out, neighbors which have changed roles (ES to IS, etc).

3.8.10 CLNS Ping Command

The **ping** command has been extended in Release 8.0 to work with the CLNS protocol. The **ping** command is used to test whether a specified destination is reachable, and if so, to report the round-trip delay encountered.

Unfortunately, the OSI CLNP Protocol (ISO 8473) does not specify a protocol for this purpose, so **cisco** has invented such a protocol. Non-**cisco** routers may or may not forward these packets (depending on whether they are specific about the packet types they will forward). Other End Systems will not recognize these packets, but will typically generate an error packet (ERPDU) as a response. This ERPDU is useful, as it confirms the reachability of the end system. The packet type sent in these "echo" packets is 7. **cisco** is working with other vendors to standardize the CLNP Echo protocol.

If a non-**cisco** router is in the path of the packet, and it refuses to forward this non-standard packet type, then an option may be used to send a data packet instead of an echo packet. This will not be replied to by **cisco** routers, but may be observed if the **debug clns-packets** command is enabled.

Several of the parameters of the CLNS **ping** command are common to other protocols. These are:

- Repeat count. The number of packets to send. The default is five packets.
- Datagram size. The total size of the CLNP packet to send, exclusive of subnetwork encapsulation bytes. The default is 100 bytes.
- Timeout in seconds. How long to wait for a response to an echo request. The default is two seconds.
- Extended Commands. Provides entry to other options.

If the "Extended commands" option is selected, then these additional options may be selected. Each may be toggled on or off by repeated selection.

- Padding. A small Padding Option is added to the packet. Three bytes total are added to the packet header, including the option id, option length, and one null byte.
- QOS-Global. A Quality-of-Service Option is added to the packet header, specifying Globally-recognized QOS. Global QOS is important because the congestion experienced bit is maintained in this option.
- Verbose. Additional information is reported.
- Data packet. An ISO 8473-compliant DT PDU (data packet) is sent, rather than the **cisco**-invented Echo Request.

The first example of the CLNS **ping** command uses all defaults as a simple test of reachability:

```
test-system#ping
Protocol [ip]: clns
Target CLNS address: 47.0004.0046.0478.aa00.0400.06cc.01
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:

Type escape sequence to abort.
Sending 5, 100-byte CLNS Echos to 47.0004.0046.0478.AA00.0400.06CC.01,
       timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 160/201/212 ms
```

This second example uses the extended commands to specify the Global QOS:

```
test-system#ping
Protocol [ip]: clns
Target CLNS address: 47.0004.0046.0478.aa00.0400.06cc.01
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Padding, QOS-Global, Verbose, Data-packet, or ? : QOS
Padding, QOS-Global, Data-packet, or ? :

Type escape sequence to abort.
Sending 5, 100-byte CLNS Echos to 47.0004.0046.0478.AA00.0400.06CC.01, timeout
is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 160/198/208 ms
```

3.8.11 CLNS on Ethernet, Token Ring and Serial Interfaces

Configuring Ethernets, Token Rings and serial lines using HDLC encapsulation for CLNS can be as simple as just enabling CLNS on the interfaces. This is all that is ever required on serial lines using HDLC encapsulation. If all systems on an Ethernet or Token Ring support ISO 9542 ES-IS, then nothing else is required as well. In this case an Ethernet and a serial line can be configured this way:

```
!
interface ethernet 0
clns enable
!
interface serial 0
clns enable
!
```

If there are systems on the Ethernet which do not use ES-IS, or if X.25 is being used, NSAP/NET (protocol address) to SNPA (media address) mappings must be specified, using the **clns is-neighbor** and/or the **clns es-neighbor** interface-specific commands. Configuring an Ethernet interface with the Ethernet address (MAC address) of systems that do not use ES-IS is as follows:

```
interface ethernet 0
es-neighbor 66.6666.00 0000.00c0.a45b
```

In this case the End Systems with NSAP (or NET) of 66.6666.00 is configured at an Ethernet MAC address of 0000.00c0.a45b.

The following is a more complete example of CLNS routing on a system with two Ethernet interfaces.

```
!
! Define a Network Entity Title and enable CLNS routing
clns routing 22.2222.00
!
! Define an NSAP for this system
clns nsap 22.2222.01
!
! Define a static route
clns route 47.0004.000c  47.0005.0000.0000.0001.0000.00
!
!
! Enable CLNS processing on Ethernet 0 and Ethernet 1
interface Ethernet 0
clns enable
!
interface Ethernet 1
clns enable
!
! Configure a static ISH entry
clns is-neighbor 44.4444.00 1234.1234.1234
!
```

3.8.12 CLNS Over X.25

X.25 is not a broadcast medium, and therefore ES-IS is not used to automatically advertise and record NSAP/NET (protocol address) to SNPA (media address) mappings. Operation of CLNS over X.25 requires that this mapping information be statically entered, by using the **clns is-neighbor** and/or the **clns es-neighbor** interface-specific commands.

Configuring a serial line to use CLNS over X.25 requires configuring the general X.25 information and the CLNS-specific information. The general X.25 information must be configured first. Then, the **clns is-neighbor** and **clns es-neighbor** commands are issued to list all the Intermediate and End Systems that will be used.

The general syntax for these commands is:

[no] clns es-neighbor *nsap snpa* [*X.25-facilities-info*]
[no] clns is-neighbor *nsap snpa* [*X.25-facilities-info*]

In this case, the SNPAs are the X.25 network addresses (X.121 addresses). These are usually assigned by the X.25 network provider. Use the *X.25-facilities-info* to specify non-default packet and window size, reverse charge information, and so on. A typical simple command that maps NSAP 33.3333.01 to X.121 address 310117 follows:

```
clns es-neighbor 33.3333.01 310117
```

Only one **es-neighbor** or one **is-neighbor** entry can be made for each X.121 address.

The X.25 facilities information that can be specified is exactly the same as in the **x25 map** command documented in the X.25 section of the *Gateway System Manual.* You can specify the following information:

- Packet window size (send and receive)
- Packet size (send and receive)
- Reverse charges requested
- Reverse charges accepted
- Closed user group
- Maximum number of virtual circuits (VCs) to open

Following is a more complicated example that specifies non-default packet and window size:

```
clns is-neighbor 47.0004.0021.0001.0000.0000.00 3101 windowsize 7 7
packetsize 512 512
```

Note that all of this configuration command must be given on one line. The above line was broken into two for typographical reasons.

The system does not accept an X.25 call unless the source of the call has been configured with these commands. The system will not accept a call requesting reverse charges unless the keyword **accept-reverse** is used as follows:

```
clns is-neighbor 47.0005.0000.0001.0000.00 310120249 accept-reverse
```

All of the information that is entered using the **clns is-neighbor** and **clns es-neighbor** commands actually goes into two places in the system: the ES-IS table, and the X.25 map table. The ES-IS table stores only the NSAP/NET and X.121 address information.

Three EXEC commands display this information:

show clns es-neighbor
show clns is-neighbor
show clns neighbor

The X.25 map table stores this information but, in addition, stores the facility information. If a virtual circuit (VC) has been established, the logical channel numbers (LCNs) in use are shown.

3.9 Novell IPX Routing

With Release 8.0, the **cisco** routers support the Novell IPX family of protocols. The **cisco** IPX support provides all of the functionality of a Novell "External Bridge" (Novell refers to their router functionality as bridging). As a Novell External Bridge, the **cisco** router connects Ethernets and Token Rings using high-speed serial lines (56 kilobit to T1 rate) or X.25. Novell workstations on any LAN, including those without a file server can connect to Novell file servers on any other LAN. Furthermore, these same interfaces and lines can be used to simultaneously route TCP/IP, DECnet, Xerox XNS, AppleTalk, and ISO CLNP, and bridge any other protocol. There are restrictions on simultaneous use of DECnet, IPX and Token Ring; see section 3.9.5.

Novell IPX is a variation on Xerox Network Services, or XNS. The major difference between Novell IPX and Xerox XNS is that packets on the IEEE 802.3 media are encapsulated in a different manner than Xerox's XNS packets, which are encapsulated on Ethernet. A second important difference is that Novell IPX includes a Service Advertisement Protocol, SAP, which is used to advertise special network services. A file server is one instance of a service which can be advertised.

Novell sells an X.25 and a T1 interface capability. At this time the **cisco** X.25 and T1 support is not compatible with Novell. This means that **cisco** routers must be used on both ends of T1 and X.25 circuits.

3.9.1 Configuring Novell

Novell addresses have a network and a host portion. The network number is written as a hexadecimal number (maximum of eight digits). The host number is formatted as an Ethernet address (a dotted triple of four hexadecimal digits). When written together the network and host portions of the address are separated by a dot. For example, the Novell address 45.0000.0c00.1ac9 indicates host 0000.0c00.1ac9 on network 45. The network number –1 indicates "all networks."

The global configuration command

[no] novell routing [*host-address*]

enables or disables Novell routing and optionally specifies the system-wide host address to use. If you do not specify an address, the MAC address of the first Ethernet interface is used. If there are no Ethernet interfaces present, you must specify the host address. The address must not be multi-cast. The **novell routing** command enables Novell RIP routing and SAP services. Novell network numbers must still be assigned to the appropriate interfaces with the **novell network** command.

If DECnet routing is being performed, you must not specify a Novell address. This restriction exists because the DECnet protocol requires the interface hardware address to correspond to the DECnet node address.

To enable Novell routing on a particular interface the **interface** subcommand is

[no] novell network *number*

where *number* is the number of the Novell network to which that interface is attached. Novell packets received on an interface which do not have a Novell network number are ignored.

Static routes for a Novell network can be specified with the global configuration command

[no] novell route *network network.address*

This causes packets received for the specified network to be forwarded to the specified router, whether or not that router is sending out dynamic routing. For example, if the router that handled traffic for network 33 had the address 45.0000.0c00.1ac9, then the command would be

```
novell route  33  45.0000.0c00.1ac9
```

There are two different data formats used by Novell on Ethernets. The command **novell encapsulation** *keyword* can be used to select which data format or encapsulation is used on an Ethernet interface. The default is **novell**. The keyword **arpa** is used when the Novell systems must communicate with other vendors' systems, such as DEC VAX/VMS.

Note: Some 3COM 3+ Novell hosts do not recognize Token Ring packets with the source-route bridging RIF field set. You can work around this Novell discrepancy by using the **no multiring** interface subcommand on Token Ring interfaces that are used for Novell IPX or 3COM XNS routing. See section 4.2.1 for more information.

3.9.2 Novell Access Lists

Access lists can be used to filter traffic on Novell interfaces. Novell access lists are defined as XNS access lists since Novel IPX is a variant of XNS. See section 3.6.3 in this document for the command format.

The Novell access list is assigned with the **interface** subcommand

[no] novell access-group *number*

where *number* refers to the appropriate XNS access list.

3.9.3 Monitoring Novell

The following EXEC commands display Novell status and configuration information.

show novell route

The **show novell route** command displays the Novell routing table. The leading character "R" indicates routes learned via RIP, "C" indicates connected entries, and "S" indicates statically-defined entries.

show novell servers

The **show novell servers** command lists the servers discovered through SAP advertisements.

show novell traffic

The **show novell traffic** command displays information on the number and type of Novell packets transmitted and received.

show novell interface [*interface*]

The **show novell interface** command shows the Novell parameters that have been configured on the interfaces. An optional interface name can be specified to see information for just that interface.

The following EXEC commands enable some debugging and tracing output that can be useful in resolving Novell-related problems.

debug novell-packet

The **debug novell-packet** command outputs information about packets received, transmitted, and forwarded.

debug novell-routing

The **debug novell-routing** command prints out information on RIP packets. RIP is the name of the routing protocol used by Novell. (It is also the name of several other routing protocols, but all are different.)

debug novell-sap

The **debug novell-sap** command displays additional information about Novell Service Advertisement packets.

3.9.4 Sample Novell Configuration

The following configuration commands enable Novell routing, defaulting the Novell host address to that of the first Ethernet interface. Routing is then enabled on Ethernet 0 and Ethernet 1 for Novell networks 2 and 11, respectively.

```
!
novell routing
!
interface ether 0
novell network 2
!
interface ether 1
novell network 11
!
end
```

3.9.5 Novell, XNS, and DECnet Configuration Restrictions

This section is important only if you are running any of DECnet, XNS, or Novell protocols concurrently in the same router. If this is not the case, skip this section.

When running in a **cisco** router, DECnet, XNS, and Novell protocols all require that interfaces be initialized to the same MAC addresses. This is done when the particular protocol is enabled for routing.

DECnet requires the MAC address to be a particular value. XNS and Novell only require that all interfaces have the same MAC address. When enabling XNS or Novell, specifying the protocol address forces all interfaces to have that same address.

Once a protocol stack is running it is assumed that the interface MAC addresses will not change. Prior to Release 8.0, this restriction was not enforced. Now when one of the three protocols is running, it is not permitted to enable another protocol that would change the interface addresses. Table 9 provides examples of configuration commands that are correct and incorrect.

Incorrect Commands	*Outcome*
`xns routing` `decnet routing 5.1`	XNS routing turns on the XNS protocol stack using the address of the first Ethernet interface that it finds. "decnet" then tries to force all interfaces to the value needed for DECnet to run. If this were allowed it would break XNS.
`xns routing` `novell routing 0230.4444.1234`	Turn on the XNS protocol stack, then turn on the Novell protocol stack, but force the interfaces to a new value. Assuming the first Ethernet interface is not 0230.4444.1234, this would break XNS.

Correct Commands	*Outcome*
`xns routing` `novell routing`	Turn on XNS using first Ethernet address. Turn on Novell using the same address. No collision.
`xns routing 0000.0c00.1014` `novell routing 0000.0c00.1014`	Turn on XNS forcing to the specified address. Turn on Novell to the same.
`decnet routing 5.8` `xns routing`	Turn on DECnet using area 5, node 8 as our DECnet address. This will set any Ethernets to aa00.0400.0814. Turn on XNS using the address of first Ethernet, aa00.0400.0814.

`decnet routing 5.8`
`xns routing`
`novell routing`
	Turn on all three protocols. Default XNS and Novell to the address set by DECnet.

`decnet routing 5.8`
`xns routing aa00.0400.0814`
`novell routing aa00.0400.0814`
	Same as the previous example, but all addresses are explicitly specified. This is what will be seen in non-volatile memory.

Table 9: Sample XNS, DECnet and Novell Commands

In addition to the above restrictions, **if there are both Ethernets and Token Rings in the system, then DECnet cannot be run simultaneously with either Novell or XNS.** This is because the Ethernet interface addresses must be set to aa00.0400.xxxx to run DECnet. This address cannot be used on Token Ring interfaces since it is a multi-cast address on Token Ring media. This arrangement means that all interfaces *cannot* be set to the same MAC address, breaking XNS and Novell.

Currently there is no standard way for running DECnet over Token Ring. **cisco Systems** at this time does not support DECnet on Token Ring. However, it would be possible to run DECnet on the Ethernets and other protocols on the Token Rings if not for the above restriction.

Table 10 indicates which of DECnet, XNS, or Novell can be run concurrently, given a particular interface configuration. 'y' indicates a supported configuration, 'n' an impossible configuration, and '*' is not supported.

Token Ring	*Ethernet*	*DECnet*	*XNS*	*Novell*
n	n	y	y	y
n	y	y	y	y
y	n	*	y	y
y	y	n	y	y
y	y	*	n	n

Table 10: Novell, XNS, and DECnet Combinations

3.10 Apollo Domain Routing

cisco Systems has implemented packet forwarding and routing for the Apollo Domain network protocols in Release 8.0. Domain is the native mode networking protocol for Apollo workstations. This capability is available on Ethernets and serial interfaces using HDLC or X.25 encapsulation. Direct attachment to the 12-megabit Domain Token Ring is not supported.

3.10.1 Configuring Apollo Domain

The global configuration command

[no] apollo routing *address*

enables or disables Domain routing and specifies which system-wide host address to use. *address* is a five digit hexadecimal number. Apollo network numbers must still be assigned to the appropriate interfaces.

This is done with the **interface** subcommand

[no] apollo network *number*

where *number* is an eight digit hexadecimal number.

Static routes for an Apollo Domain network can be specified with the global configuration command

[no] apollo route *network network.address*

This causes packets received for the specified network to be forwarded to the specified router, whether or not that router is sending out dynamic routing. For example, if the router that handled traffic for network 33 had the address 45.91ac6, then the command would be

```
apollo route  33  45.91ac6
```

3.10.2 Restrictions for Using Apollo Domain

If bridging is enabled on an Ethernet for which Apollo Domain routing is also enabled, special care must be taken. An Ethernet type code access list must be specified that filters out datagrams with the Apollo Domain type code (hexadecimal 8019).

The **cisco** Domain support assumes that it can use ARP to locate workstations on the local cable. The versions of the operating system that support Domain ARP are:

- DN3000 and DN3010 nodes need sr9.7.4.1. This version of the OS is available on a patch from local Apollo field offices. It is patch 186.
- DN3500, 4000 and 4500 nodes need 9.7.5.1 which is available on patch tape 185.
- The ARPing version of 9.7 for DN5xx-T nodes need 9.7.4.b101. No patch is available for these machines as this version was never released except on the Technet tape.

Note that sr10.0 does not do ARPing. You must migrate to 10.1 and later versions of the Domain operating system before successfully operating with **cisco** routers. Also, D-ARP support does not currently exist in Apollo's 802.5 implementation.

3.10.3 Monitoring Apollo Domain

The following EXEC commands display Apollo Domain parameters and traffic statistics.

show arp apollo

The **show arp apollo** command displays that portion of the ARP table that pertains to the Apollo Domain Address Resolution Protocol.

show apollo route

The **show apollo route** command displays the Apollo Domain routing table. Entries prefaced by "R" were learned via RIP, "C" indicates connected entries, and "S" indicates static entries defined by the **apollo route** configuration command.

show apollo traffic

The **show apollo traffic** command displays information on the number and type of Apollo Domain packets transmitted and received.

show apollo interface [*interface*]

The **show apollo interface** command shows the Apollo Domain parameters that have been configured on the interfaces. An optional *interface* name can be specified to see information for just that interface.

The following EXEC commands enable debugging and tracing output that can be useful in resolving Domain-related problems.

debug apollo-packet

The **debug apollo-packet** command outputs information about packets received, transmitted, and forwarded.

debug apollo-routing

The **debug apollo-routing** command prints out information on RIP packets. RIP is the name of the routing protocol used by Domain. (It is also the name of several other routing protocols, but all are different.)

3.10.4 Sample Apollo Domain Configuration

The following is a very simple example of setting up Apollo Domain routing on a router with two Ethernet interfaces. First, enable the RIP routing protocol and assign a Domain network address using the **apollo routing** command. Next, assign network numbers to the two interfaces, as seen in the following example:

```
!
apollo routing 23d5a
!
interface ethernet 0
apollo network 5
!
interface ethernet 1
apollo network 4
!
end
```

3.11 PUP Protocol

With Release 8.0, **cisco** routers support routing the Xerox PUP (PARC Universal Protocol) family and provide a minimal set of PUP host functions, primarily the PUP Echo server and client. PUP packets can be routed over all **cisco**-supported media. The **cisco** PUP implementation uses the PUP GWINFO routing protocol to maintain routing information across a PUP network.

Two modes of operation are available. The first mode is based on a one-to-one mapping between PUP and IP networks and addresses. It is identical to the earlier PUP protocol support that was formerly controlled by the **router gwinfo** configuration command. The second mode is a stand alone PUP protocol stack, and runs independently of IP or any other **cisco**-supported protocol.

3.11.1 Configuring PUP Routing

PUP addresses are 16-bit quantities written as two octal numbers separated by a pound sign (for example, 10#371). The most significant eight bits of the address constitute the PUP network number, and the least significant eight bits constitute the PUP host number. In the example case, the network number is 10, and the host number is 371 (both in octal).

PUP routing is enabled and disabled with the global configuration command

[no] pup routing

3.11.2 Stand Alone PUP

In the stand alone mode, addresses must be configured for each interface through which PUP traffic is routed. Any arbitrary collection of interfaces can be used for PUP traffic, unlike the mapped mode.

PUP routing is enabled on a per interface basis by the **interface** configuration command

pup address *address*

where *address* is the desired PUP address of the interface.

PUP routing can be disabled on a per interface basis by using the **interface** configuration command

no pup address

The default state is for PUP routing to be disabled.

3.11.3 PUP Mapped to IP

This is the PUP support in versions of the **cisco** router software prior to Release 8.0. PUP addresses are automatically configured by the system. The PUP addresses are calculated from the subnet and host addresses of the interfaces connected to the IP network onto which the PUP network is mapped. The PUP network number is the least significant eight or fewer bits of the subnet address of an interface on the mapped network, and the PUP host number is the least significant eight or fewer bits of the host address of the interface. As a result of this, all of the PUP networks must be mapped onto one subnetted IP network.

To configure the mapping, use the configuration command **pup map** *address* where *address* is the network number of the major network over which the PUP network is overlaid.

All PUP routes acquired via the GWINFO routing protocol are entered into the IP routing table and labeled with a 'G'. The reverse is **not** true; IP routes are not propagated into the PUP routing table.

3.11.4 PUP Miscellaneous Services

The **cisco** PUP support allows broadcasts to the PUP Miscellaneous Services socket to be forwarded to another network (the "helper address") where the appropriate servers can be found. This function is available in either mode of routing operation. To set the helper address, use the interface subcommand **pup helper-address** *address* where *address* is the PUP address of the desired server in the format described above.

3.11.5 Monitoring PUP Routing

The following EXEC commands display PUP routing entries and statistics about PUP packets.

show pup route

The **show pup route** command displays routing entries obtained from the PUP routing protocol.

show arp pup

The **show arp pup** command displays PUP-specific ARP entries as two 8-bit octal numbers separated by a pound sign.

show pup traffic

The **show pup traffic** command displays statistical summaries of PUP packets when PUP routing is enabled.

debug pup-packet

The **debug pup-packet** command enables logging of PUP routing activity to the console terminal.

debug pup-routing

The **debug pup-routing** command enables logging of PUP GWINFO routing exchanges to the console terminal.

ping

The privileged EXEC **ping** command sends PUP Echo packets. Specify the keyword **pup** when prompted for a protocol. The gateway server also prompts you for a PUP protocol address; enter an address to begin the exchange.

4 Bridging

This chapter discusses the bridging support available in the **cisco Systems** router product line. Knowledge of the routing functions and configuration of the router is assumed.

Today, most large computer networks are constructed using ISO Level 3 router (gateway) technology. The stability and traffic isolation characteristics offered by Level 3 devices are required for reliable operation. There are occasions, however, when protocols must be transported that do not have a well-formed Level 3 part or that are proprietary. These are ideal applications for Level 2 bridges.

There are two general types of bridges: transparent and source-route. Both styles of bridging are supported by the **cisco** gateway server products.

Transparent bridges are used with Ethernet networks. This style of bridge is sometimes called a MAC or Level 2 bridge. The name *transparent* comes from the fact that hosts do not have to modify packets when bridges are present.

Source-route bridges are used in IBM Token Ring (IEEE 802.5) networks. This style of bridging relies on routing information inserted into the MAC header of the packet (the Routing Information Field, or RIF) by the host to specify which rings (network segments) the packet must transit.

For both styles of bridging, the **cisco** software can act as a combination bridge and router. If a protocol is not being routed and bridging of either sort is enabled on an interface, that protocol can be bridged. This allows protocols such as IP or DECnet to be routed while other protocols, such as LAT or SNA, can be bridged. There are numerous filtering controls to specify which protocols are bridged and which are routed.

4.1 Transparent Bridging

Transparent bridges come in three basic varieties:

- Adaptive/learning
- Spanning tree
- Hybrid bridge-router

Adaptive bridges provide some traffic isolation by learning where a particular station is located. They operate by "wire-tapping" all Ethernet traffic and observing the pattern of source addresses.

Spanning tree bridges add the ability to work in configurations where loops may exist in the network.

Hybrid bridge-routers such as the **cisco** gateway server combine the advantages of a spanning tree bridge and a full multi-protocol router.

Network interfaces on the **cisco** Multi-port Communications Interface (MCI card) must be configured to serve as both multi-protocol routers and MAC level bridges, bridging any traffic that cannot otherwise be routed. For example, a gateway server routing the Internet Protocol could also bridge DEC's LAT protocol or Apple's AppleTalk protocol.

The transparent bridging support provided by **cisco** in the gateway server product lines is based upon draft versions of the IEEE Standard 802.1 (D) MAC bridges. For compatibility with the DEC and other LAN bridges, the older spanning tree protocol is also supported.

Both automatic filtering/learning and spanning tree functions are included. The bridge learns of host locations and addresses by watching the Ethernet traffic. The bridging process interacts with other **cisco** bridges to execute a spanning tree algorithm that works to eliminate the use of multiple paths that would otherwise confuse the bridge's learning functionality. The gateway server's ability to load balance and route over redundant paths is thus not limited by the use of bridging.

Remote bridging over synchronous serial lines is also supported. Ethernet datagrams are encapsulated in HDLC frames for transmission between **cisco** bridges/routers. As with datagrams received on Ethernet interfaces, the learning process and any filtering is applied to HDLC-encapsulated Ethernet datagrams.

The following sections replace descriptions and commands in Chapter 7 of the *Gateway System Manual* dated July 1988.

4.1.1 Setting Up Basic Bridging

To set up the bridging, a spanning tree protocol must be defined and network interfaces must be assigned to that spanning tree.

Two spanning tree protocols are supported in Release 8.0, the IEEE 802.1 standard and the earlier spanning tree protocol upon which the IEEE standard is based. To define a spanning tree protocol, give a configuration command of the form

bridge *group* **protocol**

followed by one of the keywords **ieee** or **dec**. The *group* parameter is a number between 1 and 9 chosen by the customer to refer to a particular spanning tree. The software allows for multiple spanning tree protocols in a single bridge/router. Few situations require more than one spanning tree group.

Note that the IEEE has not yet fully ratified the 802.1 bridging standard. It is recommended that the **dec** keyword be used.

The next step towards configuring bridging is to assign network interfaces to a particular spanning tree group. This is done with the **bridge-group** subcommand of the **interface** configuration command. The simplest form of the **bridge-group** command takes just a spanning tree group number as an argument. More complex forms of the **bridge-group** command are discussed below.

There are restrictions as to which interfaces can be configured for bridging. Only MCI Ethernet and serial interfaces are capable of supporting simultaneous bridging and routing. Earlier technology interfaces are not adequate for the task. The MCI Serial interfaces must be running with HDLC encapsulations. Bridging using X.25 or LAPB encapsulation is not possible.

Bridging between MCI interfaces can occur between interfaces on different MCI cards, although the performance is somewhat lower as compared with interfaces on the same MCI card.

All protocols except IP are bridged by a gateway server unless their routing is explicitly enabled. Consult the *Gateway Server Manual* on the procedures for enabling the routing of individual protocols. IP is normally routed by the gateway server. See section 4.1.2 for more details on IP bridging and routing.

Another point to note is that bridging and routing are done on a per system basis. If a protocol is being routed, it must be routed on all interfaces that are handling that protocol. This is similar for bridging. Routing IP on one interface and bridging it on another interface won't work.

The following is an example of a basic bridging configuration. The system has two Ethernets and one serial line on the same MCI card. The Internet Protocol (IP) is being routed and everything else is being bridged. The DEC-compatible bridging algorithm with default parameters is being used.

```
interface ethernet 0
address 192.31.7.26  255.255.255.240
bridge-group  1
interface serial 0
address 192.31.7.34  255.255.255.240
bridge-group  1
interface ethernet 1
address 192.31.7.65  255.255.255.240
bridge-group  1
bridge 1 protocol dec
end
```

4.1.2 Bridging and Routing IP

To bridge IP you must disable IP routing by giving the global configuration command

no ip routing

Assign an IP address (the *same* IP address) to the Ethernet interfaces to manage the system with Telnet, TFTP, SNMP, and so forth. You will not be able to connect or send ICMP Echo messages to serial interfaces.

After bridging is set up, all IP and ARP datagrams not intended for the **cisco** gateway server are handled according to standard bridging and spanning tree rules. Any IP routing processes (e.g., IGRP or RIP) must not be running.

Hewlett-Packard 3000 minicomputers and IBM PCs running OfficeShare generate IP datagrams with a format that requires some additional configuration to work correctly with the bridging support.

When IP routing is running, there is no problem recognizing the IEEE 802.2 format of the IP datagrams directed at the gateway server. The bridging support, however, does not recognize such datagrams as IP datagrams. Hence, when bridging, unless special access controls are in effect, some of the IEEE 802.2 format IP datagrams will be bridged when they should instead be ignored.

This problem of IEEE 802.2 format IP traffic being bridged inappropriately may be solved by a type code access list that explicitly specifies which protocols should be bridged. All other protocols, including IEEE 802.2 format IP, would then be filtered out. Type code access lists are discussed in section 4.1.5.

The same problem can occur with SNAP-encapsulated IP datagrams (see RFC 1042) and the same solution will work.

4.1.3 Adjusting Spanning Tree Parameters

It may be desirable under some circumstances to adjust certain spanning tree parameters. Parameters affecting the entire spanning tree are configured with the **bridge** configuration command, followed by the bridge group number, followed by a keyword and a keyword argument.

The interval between HELLO Bridge Protocol Data Units (BPDUs) is specified with the configuration command

bridge *group* **hello-time** *seconds*

The *seconds* parameter can be any value between 1 and 10 seconds. The default value is 1 second.

The forward delay interval is the amount of time spent listening for topology change information after an interface has been activated for bridging and before forwarding actually begins. The configuration command is

bridge *group* **forward-time** *seconds*

The *seconds* parameter can be any value between 10 and 200 seconds. The default value is 30 seconds.

If a bridge does not hear BPDUs from the root bridge within a certain interval, that bridge assumes that the network has changed and recomputes the spanning tree topology. This interval is 15 seconds by default. It can be changed with the configuration command

bridge *group* **max-age** *seconds*

Note that each bridge in a spanning tree adopts the **hello-time**, **forward-time**, and **max-age** parameters of the root bridge, regardless of what their individual configuration might be.

The priority of an individual bridge, or the likelihood that it will be selected as the root bridge, can be configured with the command

bridge *group* **priority** *number*

The *number* parameter can range from 1 to 65,000. The lower the value of *number*, the more likely the bridge will be chosen as root. The default priority is 128.

In addition to global spanning tree parameters, interface-specific spanning tree parameters can be adjusted. This is done with the **bridge-group** subcommand of the **interface** configuration command.

Each interface is associated with a path cost. The path cost can range from 0 to 255, with higher values indicating higher costs. The default value is 10. The configuration command

bridge-group *group* **path-cost** *cost*

can be used to set a different path cost.

A priority can also be associated with an interface. This priority is used in tie-breaking when computing a network topology. The configuration command

bridge-group *group* **priority** *number*

can be used to set an interface priority. The *number* parameter can range from 0 to 255. The default value is zero.

4.1.4 Administrative Filtering By Address

Administrative filtering can be done by Ethernet address, by Ethernet protocol type code or by vendor code. When setting up administrative filtering, remember that there is virtually no performance penalty in filtering by Ethernet address or vendor code. There can be a significant performance penalty when filtering by Ethernet type code.

To block datagrams with a particular Ethernet station source or destination address, use the configuration command

bridge *group* **address** *ethernet-address* [**forward**|**discard**] [*interface*]

followed by the keywords **forward** or **discard**. The *ethernet-address* parameter is a 48-bit dotted triple address such as is displayed by the **show arp** command (e.g., 0800.cb00.45e9). The *ethernet-address* parameter can be a station address or a broadcast or multi-cast destination address.

If the **forward** keyword is specified, a datagram sent from or destined to the specified address is forwarded as appropriate. If the **discard** keyword is specified, a datagram sent from or destined to the specified address is discarded without further processing.

An optional interface specification such as *Ethernet 0* can be added after the **discard** or **forward** keyword. This interface specification indicates the interface on which that address can be reached.

The command

no bridge *group* **address** *ethernet-address*

removes an address from the forwarding database. Any number of addresses can be configured into the system without a performance penalty.

Normally the system forwards any datagrams for stations that it has learned about dynamically. This default can be changed with the configuration command

no bridge *group* **acquire**

The resulting default is to filter out all datagrams except those sourced by or destined to addresses that have been statically configured into the forwarding cache.

The command

bridge *group* **acquire**

restores the default behavior.

The **bridge** configuration command has been enhanced to enable the forwarding (but not learning) of multi-cast source addresses. The command has the form

[no] bridge *group* **multicast-source**

4.1.5 Administrative Filtering By Type Code

Filtering by Ethernet protocol type code is done using the access list mechanism. The **access-list** command is used to specify an element in an access list. The order in which **access-list** commands are entered into the system affects the order in which the access conditions are checked. Each condition is tested in succession. A matching condition is then used to execute a permit or deny decision. If no conditions match, a deny decision is reached.

Type code access lists have the form

access-list *list keyword type-code wild-mask*

The *list* parameter is a customer-selected number between 200 and 299 that identifies the list. The *keyword* parameter is one of **permit** or **deny**. The *type-code* parameter is a 16-bit hexadecimal number written with a leading "0x", for example, 0x6000. The *wild-mask* is another 16-bit hexadecimal number whose one bits correspond to bits in the *type-code* parameter that should be ignored when making a comparison.

For example, the following access list permits only LAT packets (type 0x6004) and filters out all other packet types.

```
access-list  201  permit  0x6004  0x0000
access-list  201  deny    0x0000  0xFFFF
```

The following access list filters out only type codes assigned to DEC (0x6000 through 0x600F) and lets all other types pass.

```
access-list  202  deny    0x6000  0x000F
access-list  202  permit  0x0000  0xFFFF
```

It is always a good idea to use the last item of an access list to specify a default action, for example, permit everything else or deny everything else. If nothing else in the access matches, the default action is normally to deny access, that is, filter out all other type codes.

Access lists can have an impact on system performance, therefore, **cisco** advises keeping the lists as short as possible and using wildcard bit masks whenever possible.

Packets can be filtered by type on input. The access list specifying the type codes to be filtered is given by the **interface** subcommand **bridge-group** *group* **input-type-list** *list*. This access list is then applied to all packets received on that interface prior to the bridge learning process.

Packets can be filtered by type on output. The access list specifying the type codes to be filtered is given by the **interface** subcommand

bridge-group *group* **output-type-list** *list*

This access list is then applied just before sending out a packet to an interface.

For performance reasons, it is not a good idea to have both input and output type code filtering on the same interface.

Access lists for Ethernet type codes affect only bridging functions. It is not possible to use such access lists to interfere with protocols that are being routed.

The following example is of a system with two Ethernet and one serial interfaces on the same MCI. The system is routing both IP and DECnet. Each interface has an access list that allows only the LAT protocol to be bridged. The bridging software has also been instructed to discard datagrams sent to or from the address AB00.0C00.AE35.

```
!
decnet address 34.88
!
interface ethernet 0
address 192.31.7.26  255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
interface serial 0
address 192.31.7.34  255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
interface ethernet 1
address 192.31.7.65  255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
bridge 1 protocol dec
bridge 1 address AB00.0C00.AE35 discard
!
access-list 201 permit 0x6004  0x0000
access-list 201 deny   0x0000  0xFFFF
!
end
```

4.1.6 Administrative Filtering By Vendor Code

Ethernet address access lists have also been added. These lists support filtering groups of Ethernet addresses, including those with particular vendor codes. There is no noticeable performance loss in using these access lists. The lists can be of indefinite length.

The **access-list** command for Ethernet address access lists has the following form

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

where *list* is an integer from 700 to 799 and *address* and *mask* are 48-bit Ethernet addresses written in dotted triple form. The ones bits in *mask* are the bits to be ignored in *address.*

The **interface** subcommand

[no] bridge-group *group* **input-address-list** *list*

assigns to a particular interface an access list for filtering Ethernet source addresses.

Likewise, the **interface** subcommand

[no] bridge-group *group* **output-address-list** *list*

assigns to a particular interface an access list for filtering Ethernet destination addresses.

As an example, suppose you wish to disallow the bridging of Ethernet packets of all SUN workstations on Ethernet 1. Software assumes that all such hosts have Ethernet addresses with the vendor code 0800.2000.0000. The first line of the access list denies access to all SUN workstations while the second line permits everything else. You then assign the access list to the input side of Ethernet 1.

```
access-list 700 deny 0800.2000.0000  0000.00FF.FFFF
access-list 700 permit 0000.0000.0000  FFFF.FFFF.FFFF
interface ethernet 1
bridge-group 1 input-address-list 700
```

4.1.7 Bridge Monitoring and Troubleshooting

The EXEC commands for monitoring and troubleshooting bridging are as follows.

debug span

The **debug span** command is used to track changes in the spanning tree topology. This command is useful for verifying correct operation of the spanning tree protocol.

clear bridge *group*

The **clear bridge** *group* command removes any learned entries from the forwarding database and zeros the transmit and receive counts for any statically configured forwarding entries.

The EXEC **show bridge** command allows viewing of classes of entries in the bridge forwarding database. The new form of the command is

show bridge [*bridge-group*] [*interface*]
show bridge [*bridge-group*] [*address* [*mask*]]

The *bridge-group* number is optional. You can specify an interface (e.g., Ethernet 0) or a 48-bit Ethernet address with an optional mask of bits to be ignored in the address. In the example below, the first command would display all entries for hosts reachable via interface Ethernet 0, the second command would display all entries with the vendor code of 0000.0c00.0000, and the third command displays the entry for address 0000.0c00.0e1a. In the fourth command, all entries in the forwarding database would be displayed. In all four examples, the bridge group number has been omitted.

```
show bridge ethernet 0
show bridge 0000.0c00.0000  0000.0cFF.FFFF
show bridge 0000.0c00.0e1a
show bridge
```

Sample output of the **show bridge** command follows:

```
Total of 300 station blocks, 295 free

BG Hash    Address        Action   Interface Age  RX count  TX count
 1 00/0    FFFF.FFFF.FFFF discard      -      P         0         0
 1 09/0    0000.0C00.0009 forward  Ethernet0  0         2         0
 1 49/0    0000.0C00.4009 forward  Ethernet0  0         1         0
 1 CA/0    AA00.0400.06CC forward  Ethernet0  0        25         0
```

The first line of the **show bridge** output lists the total number of forwarding database elements in the system and the number in the free list. The total number of forwarding elements is expanded dynamically, as needed. The first column of the display indicates the bridging group to which the address belongs. The second column is the address, the third column is the action to be taken when that address is looked up, and the fourth column is the interface, if any, on which that address was seen. The *Age* column is the number of minutes since a datagram was received from or sent to that address. The letter "P" indicates a permanent entry. The *RX count* column counts the number of datagrams received from that address. The *TX count* column counts the number of datagrams sent to that address.

show span

The **show span** command displays the spanning tree topology known to the system. The first part of the display lists global spanning tree parameters. Port-specific parameters are listed thereafter.

Following is a sample output of the **show span** command.

```
Bridge Group 1 is executing the DEC compatible spanning tree protocol
  Bridge Identifier has priority 129, address 0000.0C00.0005
  Configured hello time 1, max age 15, forward delay 30
  We are the root of the spanning tree
  Acquisition of new addresses is enabled
  Topology change flag not set, detected flag not set
  Times:  filter 1, topology change 30, notification 30
          hello 1, max age 15, forward delay 30
  Timers: hello 1, topology change 0, notification 0
Port 1 (Ethernet0) of bridge group 1 is forwarding.  Path cost 10
   Designated root has priority 129, address 0000.0C00.0005
   Designated bridge has priority 129, address 0000.0C00.0005
   Designated port is 1, path cost 0
   Timers: message age 0, forward delay 0, filter 1
   Access list for input type filter is not set
```

```
   Access list for output type filter is not set
Port 2 (Serial0) of bridge group 1 is forwarding.  Path cost 10
   Designated root has priority 129, address 0000.0C00.0005
   Designated bridge has priority 129, address 0000.0C00.0005
   Designated port is 2, path cost 0
   Timers: message age 0, forward delay 0, filter 1
   Access list for input type filter is not set
   Access list for output type filter is not set
```

4.1.8 Bridging Configuration Command Summary

This section summarizes bridging-specific configuration commands. The general use of configuration commands is discussed in the *Gateway System Manual* dated July 1988.

The following are top level configuration commands that set up a spanning tree protocol prior to bridging actually taking place.

bridge *group* **protocol dec|ieee**
bridge *group* **hello-time** *seconds*
bridge *group* **forward-time** *seconds*
bridge *group* **max-age** *seconds*
bridge *group* **priority** *number*
bridge *group* **acquire**
bridge *group* **address** *ethernet-address* **forward|discard** [*interface*]
bridge *group* **multicast-source**

The **bridge-group** configuration command is a subcommand of the **interface** configuration command. It is used to set interface-specific bridging parameters.

bridge-group *group*
bridge-group *group* **path-cost** *cost*
bridge-group *group* **priority** *number*
bridge-group *group* **output-type-list** *list*
bridge-group *group* **input-type-list** *list*

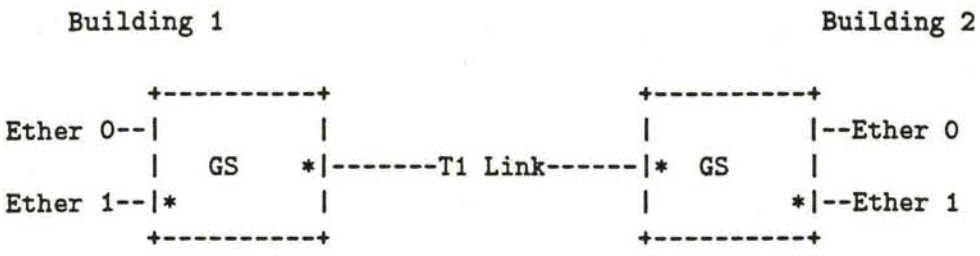
The **access-list** command has a special format for administrative filtering based on Ethernet type codes:

access-list *list* **permit|deny** *type-code wild-mask*

4.1.9 A Sample Configuration

In this example, two buildings have networks that must be connected via a T1 link. For the most part, the systems in each building use either IP or DECnet and hence should be routed. There are some systems in each building that must communicate, but they can use only a proprietary protocol.

This example places two Ethernets in each building. One of the Ethernets will be used to attach to the rest of the building network that speaks IP and DECnet. The other Ethernet will be attached to the hosts that use a proprietary protocol. This Ethernet will be enabled for bridging to the serial line and hence to the other building. The configuration looks like this:



The interfaces marked with "*" are to be configured as part of spanning tree 1. The routers will be configured to route IP and DECnet. This configuration permits hosts on any Ethernet to communicate with hosts on any other Ethernet using IP or DECnet. In addition, hosts on Ethernet 1 in either building can communicate using protocols not supported for routing.

The configuration file for the gateway server in Building 1 would be as follows. (Note that no bridging takes place over Ethernet 0. Both IP and DECnet routing are enabled on all interfaces.)

```
decnet address 3.34
interface ethernet 0
address 128.88.1.6  255.255.255.0
decnet cost 10
!
interface serial 0
address 128.88.2.1  255.255.255.0
bridge-group  1
decnet cost 10
!
interface ethernet 1
address 128.88.3.1  255.255.255.0
bridge-group  1
```

```
decnet cost 10
!
bridge 1 protocol dec
end
```

The configuration file for the gateway server in Building 2 is very similar.

```
!
decnet address 3.56
!
interface ethernet
address 128.88.11.9  255.255.255.0
decnet cost 10
!
interface serial 0
address 128.88.2.2  255.255.255.0
bridge-group  1
decnet cost 10
!
interface ethernet 1
address 128.88.16.8  255.255.255.0
bridge-group  1
decnet cost 10
!
bridge 1 protocol dec
!
end
```

4.2 Token Ring Multi-ring Support

8.0 is a major release for new Token Ring functionality. Included is support for multi-ring communications and source-route bridging. This section focuses on the multi-ring support necessary for source-route bridging.

Multi-ring support allows a **cisco** router to communicate to hosts across Token Ring source-route bridges. Prior to Release 8.0, **cisco** routers could only communicate to hosts connected to the same ring.

Source-route bridging allows a **cisco** router to act as a bridge between two Token Rings. The **cisco** router forwards traffic from one Token Ring to another for protocols that are not being routed. This allows a **cisco** router to handle protocols that it otherwise does not know about. This occurs on the same interfaces that carry routed traffic.

The source-route bridge uses the Routing Information Field (RIF) contained in the MAC header of a Token Ring packet. The RIF is inserted into the MAC header by the originating or source host, hence the name "source-routing." Thus a source bridge is not transparent in the sense of an Ethernet transparent bridge where hosts do not participate in the bridging process.

To support source-route bridging, the Token Ring interface firmware was extended. Release 8.0 requires Token Ring Monitor Version 2.0 or higher. An appropriate error message is displayed if an incompatible combination of system software and Token Ring interface firmware is detected.

In brief, the new features of the Release 8.0 Token Ring support are:

- Full multi-ring support
- Multi-ring debugging tools
- Static RIF cache entries
- User-settable RIF cache aging
- Two-port source bridge
- Simultaneous source bridging and routing
- Source bridge debugging tools
- Support for 3COM 3+ Share on Token Ring
- Token Ring firmware Revision 2.0

A **cisco** router with Token Ring interfaces is by default a multi-ring host. It keeps track of the location of hosts with which it is communicating by using the RIF (Routing Information Field) seen in incoming packets. When generating packets, the destination MAC address is looked up in this RIF cache and, if not found, an explorer packet will be generated. If the destination host supports source-routing, it will respond. The **cisco** router inserts the resultant RIF into its RIF cache. Broadcast packets and various types of multi-cast packets are always sent out as explorer packets. This includes ARP broadcast requests. This allows a **cisco** router to communicate with other TCP/IP hosts in a multi-ring environment.

The system administrator has the option of turning off the multi-ring behavior on a per interface basis. This may be useful in a number of circumstances. For example, it may be that all traffic is routed and the extra traffic generated by explorer packets is not desired. There may also be hosts on the local ring which cannot understand packets containing RIFs, including some Novell hosts.

The system administrator also has the ability of making static entries in the RIF cache. This can be used in cases where the target host does not implement multi-ring protocols.

4.2.1 Multi-ring Configuration Commands

There is one new **interface** subcommand to set up multi-ring configurations.

[no] multiring

The **multiring** configuration command is used to enable or disable the specified interface's ability to use multi-ring protocols. Multi-ring functionality is off by default. Typically, it would be turned on when all hosts on that ring understand RIFs.

There are two new global configuration commands:

[no] rif timeout *minutes*

This command sets the period of inactivity after which unused RIF cache entries are removed. The form **no rif timeout** resets the RIF timeout period to its default of 15 minutes.

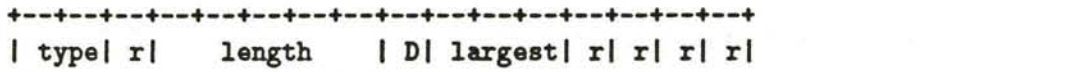
[no] rif *MAC-address [RIF-string]*

The MAC address is a 12-digit hexadecimal string written as a dotted triple, (e.g., 0010.0a00.20a6). The RIF string consists of a series of 16-bit hexadecimal numbers separated by a dot (.). An exact representation of the RIF is inserted when a packet is sent to the destination with the specified MAC address. **no rif** *MAC-address* removes an entry from the cache. The format of a RIF string is illustrated in Figures 3, 4, and 5.

The command **rif** *MAC-address* puts an entry into the RIF cache indicating that packets for this MAC address do not have a RIF. This is very different from an entry not being in the cache.

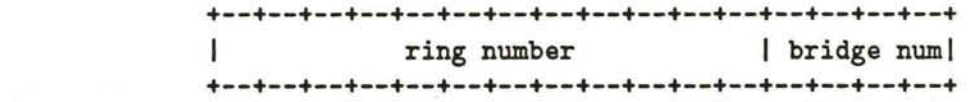
```
+-------+-------+-------+---
|  RC   |  RD   |  RD   | ...
+-------+-------+-------+---

RC = Routing Control field
RD = Routing Descriptor
each block is 16 bits wide.
```

Figure 3: Basic RIF Format




```
r         reserved
type      RIF type, 00: specific route
                    10: all rings, all routes
                    11: all rings, spanning routes (limited broadcast)
length    total length in bytes of the RIF.
D         direction 0: interpret route left to right (forward)
                    1: interpret route right to left (reverse)
largest   Largest frame that can be handled by this route.
                  000:   516 bytes
                  001:  1470 bytes
                  010:  2052 bytes
                  011:  4472 bytes (current cisco maximum)
                  100:  8144 bytes
                  101: 11454 bytes
                  110: 17800 bytes
                  111: initial value (not ever used).
```

Figure 4: RIF Routing Control Format




```
ring number: transit ring number.  Unique within the bridged network.
bridge num:  transit bridge.  Unique between any two parallel rings.
```

Figure 5: Routing Descriptor Format

As an example, the path between two rings 8 and 9 connected via a source-route bridge 1 is described by the route descriptor 0081.0090 (in hexadecimal). A full RIF, including the route control field, would be 0630.0081.0090.

To describe a path from a router to a host two hops away, one might use the RIF, 0830.0155.100a.5550. A packet with this RIF would leave the **cisco** router on ring 21 (0x15), traverse bridge 5 to ring 256 (0x100), and then traverse bridge 10 (0xa) to ring 1365 (0x555) for delivery to the destination host on that ring.

It is not recommended that you configure static RIFs with any of the "all rings" type codes. Doing so causes traffic for the configured host to appear on more than one ring causing unnecessary congestion.

4.2.2 Monitoring RIF Activity

The following EXEC commands display RIF activity and statistics.

debug rif

Use the EXEC command **debug rif** to watch RIF cache activity. This command shows insertions, changes, and flushed entries as the cache changes.

An example of these displays is shown below.

```
RIF: flushed 0000.2856.4800
RIF: flushed 4007.0600.03ED
RIF: U chk 8000.2856.4800 [C630.130F.1290] type 4
RIF: U add 0000.2856.4800 [C630.130F.1290] type 4
RIF: L checking 0000.2856.4800
RIF: U chk 8000.2856.4800 [0630.130F.1290] type 4
TR0: sent XID response to 8000.2856.4800
RIF: expired 4007.0600.03F5
RIF: L checking 4007.0600.03F5
RIF: L Sending XID for 4007.0600.03F5
RIF: U chk C007.0600.03F5 [06B0.130F.1290] type 4
RIF: U add 4007.0600.03F5 [06B0.130F.1290] type 4
RIF: rcvd XID from C007.0600.03F5.
        ...
```

show rif

The EXEC command **show rif** is used to view the current contents of the RIF cache.

A sample display of this command follows:

```
wilma>show rif

rif timeout 20 minutes.

Hardware Addr   Idle (min)    Routing Information Field
0000.284E.4800          *     -
0000.2856.4800          4     06B0.130F.1290
4007.0600.03F5          *     -
0000.0000.4444          -     -
5000.5a1a.21a4          -     0230
```

Entries marked with "*" denote **cisco's** interface addresses. Entries marked with "–" denote static entries. Entries with a number denote cached entries. If the RIF timeout is set to something other than the default of 15 minutes, the timeout is displayed at the top of the display.

4.3 Source-Route Bridging

In Release 8.0, the **cisco** Token Ring implementation allows a **cisco** router to be configured as a combination router and source-route bridge. Similar to a **cisco** Ethernet bridge, a source-route bridge allows the passing of packets between connected rings for protocols that cannot be routed or that are currently configured to bridge. A source bridge uses only the information contained in a packet's RIF in deciding whether to bridge the packet. It looks at the information contained in the Route Descriptors to see if one of them indicates that the packet should traverse this bridge. In addition to propagating specific routes, a source bridge also participates in route exploration. An explorer packet eligible for forwarding will be modified to indicate that it has traversed this bridge. There are two types of explorer packets, *all rings* and *spanning*.

Spanning is also called *limited routing* in IBM literature. An all rings explorer will be forwarded if it has not been on the target ring yet. A spanning explorer will be forwarded if there has not been a spanning tree violation (e.g., a loop) and both the input and output port are in a forwarding state.

Spanning or limited route explorers are used to curtail the explosive propagation of all rings explorers. The spanning tree is currently established by manually configuring ports to be in the forwarding state. This is equivalent to how IBM source bridges are configured. Interfaces default to being in a blocking state. An IEEE subcommittee is working on an algorithm to establish the spanning tree automatically.

4.3.1 Configuring the Source-Route Bridge

There are several new **interface** subcommands that control source-route bridging behavior:

source-bridge *local-ring bridge-number target-ring*

This form of the **source-bridge** command is used to enable source bridging on an interface. *local-ring* is the ring number for this interface's ring. It must be unique within the bridged Token Ring network. *bridge-number* is the number of this source bridge. It must be unique for any bridges connecting the same two rings. *target-ring* is the ring number that is the other half of this bridge. It must also be unique within the bridged Token Ring network.

The current implementation only supports dual-port source bridges, bridges that connect two rings. Since the AGS chassis can support up to four Token Rings, two dual-port bridges can be configured in a single **cisco** chassis. Remember that these are logically separate bridges and must each meet the above configuration restrictions.

In this implementation, a bridge is configured by specified commands on the two associated interfaces. The bridge number used must be the same for both interfaces, and the local ring and target ring parameters must be consistent. See the examples below for more details.

no source-bridge

This command is used to disable source bridging on a particular interface. Note that source bridging increases the system loading when enabled and thus should be disabled unless you need source-route bridging functionality.

[no] source-bridge span

This command is used to manually change the forwarding state with respect to spanning explorer packets. **source-bridge span** enables the forwarding of these kinds of packets. Due to the manual nature of this spanning tree it is possible through misconfiguration to create bridging loops. When a **cisco** source bridge detects one of these loops, an appropriate message is displayed.

[no] source-bridge max-rd *count*

This command allows the system administrator to limit the maximum size of the RIF that this bridge will deal with. This effectively limits the maximum diameter of the bridged network. *count* determines the number of route descriptors that may appear in the RIF. It is one more than the number of bridges a packet may traverse. The command **no source-bridge max-rd** resets the count back to the maximum value.

4.3.2 Monitoring Source Bridging

To monitor the source bridge use the **debug source-bridge** command. Messages displayed include the following:

```
SRB0: forwarding from ring 122 to 232
SRB3: forwarding from ring 4002 to 1 (spanning)
SRB0: max rif length exceeded - 0x14.
SRB2: spanning violation 422 to 256 [c830001410051010]
SRB0: dropping spanning - not forwarding
SRB2: forwarded specific from 101 to 102
SRB1: enqueued for bridge input (from 202 to 203)
```

The **show interface** command display for Token Rings has been enhanced to provide information about the state of source bridging for this interface. An example is provided below.

```
TokenRing 1 is up, line protocol is up
  Hardware type is Token Ring, hardware address is 0000.1403.4800
  Internet address is 131.108.130.2, subnet mask is 255.255.255.0
  MTU 4464 bytes, BW 4000 Kbit, DLY 2500 usec, rely 255/255, load 1/255
  Encapsulation is SNAP, loopback is not set, keepalive is not set (10 sec.)
  ARP type: SNAP
  Source Bridge: local ring 304 (0x130), target ring 297 (0x129)
      Bridge num 15, Max route desc. 8, span explorers enabled
      Bridge (cnt:bytes) rx: 10:1248, tx: 15:1524, drops: 0
  Multiring node
  Last Ring Status 0:20:35 <Soft Error> (0x2000)
  Last input 0:02:26, output 0:00:47, 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
     13 packets input, 724 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     9 packets output, 540 bytes
     0 output errors, 0 collisions, 2 interface resets, 0 restarts
     5 transitions
```

The fields relating to source-route bridging are illustrated in this next example:

```
  Source Bridge: local ring 304 (0x130), target ring 297 (0x129)
      Bridge num 15, Max route desc. 8, span explorers enabled
      Bridge (cnt:bytes) rx: 10:1248, tx: 15:1524, drops: 0
  Multiring node
```

Most of the fields are self explanatory. The **(cnt:bytes)** fields are respectively the number of packets and number of bytes received or transmitted. The **drops** report indicate the number of packets that were discarded for various reasons. **Multiring node** indicates that this interface is enabled for using the multi-ring Token Ring protocols. If disabled this line will read **Single ring node.**

4.3.3 Source Bridge Commands Summary

The **interface** subcommands for source-route bridging follow:

[no] multiring
source-bridge *local-ring bridge-number remote-ring*
no source-bridge
[no] source-bridge span
[no] source-bridge max-rd *count*

The global configuration commands are:

[no] rif timeout *minutes*
[no] rif *MAC-address [RIF-string]*

The EXEC commands include:

debug rif
debug source-bridge
show rif

4.3.4 Configuration Notes and Examples

The Token Ring system software is written such that a minimum of configuration is the normal case. A Token Ring equipped router is by default a multi-ring host. Source bridging is off by default.

Basic Token Ring Configuration

This example configures a simple router.

```
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
!
interface Ethernet 0
ip address 131.108.2.68 255.255.255.0
!
```

Basic Token Ring Source Bridge

In this partial configuration the source bridge is turned on, IP is routed, and all other protocols are bridged.

```
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
!
interface Ethernet 0
ip address 131.108.2.68 255.255.255.0
!
```

Source Bridge Only

In this partial configuration all protocols are bridged including IP. Note that because IP is being bridged, the system has only one IP address.

```
!
no ip routing
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
!
interface TokenRing 1
ip address 131.108.129.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
!
interface Ethernet 0
ip address 131.108.129.2 255.255.255.0
!
```

Dual Bridges in a Single Router

In this example, two source-route bridges are contained in the same router. IP is being routed while all other protocols are being bridged. While it is possible to configure both bridges to connect the same two rings (the bridge numbers would have to different), this is not a recommended configuration since the router would be a single point of failure. Redundancy would be better served by two different routers.

```
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
!
interface TokenRing 2
ip address 131.108.131.2 255.255.255.0
source-bridge 131 2 132
source-bridge spanning
!
interface TokenRing 3
ip address 131.108.132.2 255.255.255.0
source-bridge 132 2 131
source-bridge spanning
!
```

Other Protocols Routed at the Same Time

In this configuration IP, XNS, and Novell are all being routed while all others will be bridged between rings. While not strictly necessary, the Novell and XNS network numbers are set consistently with the IP subnetwork numbers. This makes the network easier to maintain.

```
!
xns routing 0000.0C00.02C3
!
novell routing 0000.0C00.02C3
!
```

```
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
xns network 129
novell network 129
source-bridge 129 1 130
source-bridge spanning
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
xns network 130
novell network 130
source-bridge 130 1 129
source-bridge spanning
!
interface Ethernet 0
ip address 131.108.2.68 255.255.255.0
xns network 2
novell network 2
!
```

Communicating With an IBM RT

IBM RTs running AIX 2.1.1 (and possibly other versions of the RT operating system) have a known problem that precludes communication with non-IBM hosts on the same ring. **cisco** provides two workarounds. The first involves installing a static RIF entry for every RT with which the router communicates. If there are many RTs on the ring this can be impractical. The second workaround involves setting the MAC address of the **cisco** Token Ring to a value that the RT recognizes. In presenting the configurations only the additional lines are given.

Method 1: Static RIF

```
rif 5000.5a1a.2a1f 0230
rif 5000.5a1a.2a20 0230
rif 5000.5a1a.3601 0230
```

In this example, there are three RTs directly connected to the same ring as the **cisco** router. The MAC address of each RT's interfaces is the first parameter to the **rif** configuration command. This address must be determined either using tools provided on the RT or **debug arp** on the **cisco** router.

Method 2: Modified **cisco** MAC address

If a large number of RTs must be configured, it will be easier to change **cisco's** MAC address to allow compatibility with the RTs. This incompatibility is due

to a problem with the RT implementation. This method uses the **xns routing** command to change all interfaces in the router, specifically any Token Rings, to the same address. XNS is not actually turned on for the interfaces involved. If XNS is desired, then the **xns network interface** subcommand should be used to actually enable this traffic.

The configuration command is:

```
xns routing 5000.5axx.xxxx
```

where **xx.xxxx** is an appropriate second half of the MAC address to use. This number should be chosen so that it does not duplicate any addresses on any of the router's interfaces or any addresses currently in use on the Token Ring.

Configuring in the Presence of Hosts Without XID

This example configures the system to communicate with hosts that do not implement the IEEE 802.2 XID command required by the standard. XID is the mechanism used to obtain the RIF information. In the case of IP, since the ARP and RIF caches are generally aged at different rates, the XID mechanism is used to update the RIF cache when there is still a valid ARP cache entry.

The preferred solution involves making a static RIF entry in the cache. To use this method, the RIF for the faulty host and its MAC address must be known. Assuming that for our example this RIF is across ring 144, bridge 5, ring 206, bridge 3, ring 1022, bridge 10, to ring 566 the static entry would be configured as below. The RIF data itself is entered in hexadecimal. This method works for any network level protocol.

```
rif 0001.3000.0200 0a30.0905.0ce3.3fea.2360
```

Another potential solution, although not recommended, would involve increasing the RIF timeout period. Modifying the RIF timeout affects how the system recovers from host movement and changing topology conditions. Modifying the timeout would introduce potential network problems in an attempt to compensate for the 802.2 implementation.

The best solution is to get a complete version of the software which implements XID for the host. The above patch is shown as a stopgap way to keep the network running until a complete software implementation is installed.

5 The X.25 Protocol

This section describes the enhancements to commands supporting the X.25 Protocol in software Release 8.0.

5.1 X.25 Enhancements

With Release 8.0, X.25 can be used to transport Novell, Apollo and CLNS datagrams, as well as the IP, DECnet, AppleTalk and XNS datagrams previously supported. Two new protocol keywords, **novell** and **apollo** have been added to the **x25 map** subcommand. CLNS invokes X.25 mapping actions through the **clns is-neighbor** or **clns es-neighbor** command and never uses the **x25 map** subcommand.

Table 11 lists the hexadecimal values of the initial byte of Call User Data used by the **cisco** software to distinguish which high-level protocol will be carried by a particular virtual circuit. The use of 0x81 for ISO 8473 (CLNS) is an ISO standard. The use of 0xCC for DOD IP is defined by RFC 877. The other values are meaningful only to **cisco** software. All the values are padded with three bytes of 0x00, except for the BFE X.25 encapsulation.

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

Table 11: Protocols and Initial Byte of Call User Data

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 updated by using the **x25 use-source-address** subcommand.

The **x25 map** command now supports the **nvc** *count* argument to set 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.

The various EXEC **show** commands relating to X.25 have been consolidated as follows:

show x25 map
show x25 route
show x25 switch
show x25 vc *circuit*
show x25 pad *line*

The EXEC command **debug x25-vc** *lcn* takes an integer argument which specifies the logical channel number of a virtual circuit (switched or permanent) to watch with **debug x25** or **debug x25-events**.

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. The new **x25** configuration subcommand is

x25 hold-vc-timer *minutes*

where *minutes* is the number of minutes to leave the virtual circuit idle (and ignore packets for a given destination) in this circumstance.

The **x25** configuration command

[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** gateways open one virtual circuit for each type of service. An associated problem is that some hosts send non-standard data in the TOS field, thus causing multiple, wasteful virtual circuits to be created.

The Defense Data Network Blacker Front-End (an encryption device) is supported. A new encapsulation keyword, **bfex25**, provides a mapping from Class A IP addresses to the type of X.121 addresses expected by the BFE. Contact **cisco** for further details.

The **lapb th** configuration parameter has been removed.

In Release 7.1 the ability to route XNS over X.25 circuits was added; see section 3.6 for more information.

5.2 X.25 Switching

Generalized X.25 switching is now supported as part of the X.25 software.

Following is a list of the facilities and parameters that the X.25 switching subsystem supports:

- The D-bit is ignored but passed through transparently.
- The modulo sequence is set to 8 at the packet level.
- Support for variable length Interrupt data.
- Support for Flow Control Parameter Negotiation:
  - Window size up to 7.
  - Packet size up to 1024.
- Support for the basic closed user group.
- Support for throughput class negotiation.
- Support for 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.

Previous versions of the **cisco** X.25 implementation would only originate or terminate virtual circuits. The software also expected the contents of the virtual circuits to contain a higher level protocol such as IP or DECnet. In Release 8.0, switched virtual circuits can 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 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 it 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 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, and T1 serial. Thus X.25 data can be forwarded over these media to another **cisco** router where it can be output to an X.25 interface.

5.2.1 Configuring X.25 Switching

To establish X.25 switching, first give the global configuration command **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.

Next an X.25 routing table must be constructed. This table is consulted when an incoming call is received which 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 following global configuration command:

[no] x25 route [*#position*] *x121-pattern* [**cud** *pattern*] **interface** *interface-name*
[no] x25 route [*#position*] *x121-pattern* [**cud** *pattern*] **ip** *ip-address*

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 (a # 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 *x121-pattern* parameter is the X.121 address of the called destination 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.

As mentioned earlier, the X.121 address and Call User Data are used to find a matching routing table entry. The list is scanned from the beginning to the end and each entry is pattern-matched with the incoming X.121 address and Call User Data to the X.121 and Call User Data in the routing table entry. If the pattern match for both entries succeeds, then that route is used. If the incoming call does not have any Call User Data, then only the X.121 address pattern match need succeed with an entry which only contains an X.121 pattern. If Call User Data is present, and while scanning a route is found which matches the X.121 address, but the route doesn't have a Call User Data pattern, then that route is used when an exact match cannot be found.

Regular expressions are used to allow pattern matching operations on the X.121 addresses and Call User Data. The most common operation is to do prefix matching on the X.121 DNIC field and route accordingly. For example, the pattern ^3306 will match all X.121 addresses with a DNIC of 3306. The caret (^) is a special regular expression character that says to anchor the match at the beginning of the pattern. See Section 5.2.2 for a complete list of regular expression characters and their behavior.

If a matching route is found, the incoming call is forwarded to the "next hop" depending on the routing entry. If no match is found, the call is cleared. If the route specifies a serial interface running X.25, the call will attempt to be forwarded over that interface. If the interface is not operational or out of available virtual circuits, the call will be cleared. Otherwise, a Clear Request or Call Accepted message will be expected and forwarded back towards the originator.

If the matching route specifies an IP address, a TCP connection will be established to port 1998 at the specified IP address which must be another **cisco** router. The Call Request packet will be forwarded to the remote router where it will be processed in a similar fashion. If a routing table entry isn't present or the serial interface is down or out of virtual circuits, a Clear Request will be sent back and the TCP connection will be closed. Otherwise, the call will be forwarded over the serial interface and the Clear Request of Call Accepted will be returned. Incoming calls received via TCP connections that match a routing entry specifying an IP address will be cleared. This restriction prevents **cisco** routers from establishing a TCP connection to another router which would establish yet another TCP connection. A router must always connect to the remote router with the destination DTE directly attached.

5.2.2 Regular Expressions

Regular expressions provide a way to specify wide ranges of X.121 addresses and Call User Data fields by using just a few keystrokes. If you are familiar with regular expressions from UNIX programs such as *regexp*, you are already familiar with much of **cisco System's** regular expression implementation.

Writing regular expressions is simple once you see and try a few examples. A regular expression is a formula for generating a set of strings. If a particular string can be generated by a given regular expression, then that string and regular expression *match*. In many ways, a regular expression is a program, and the regular expression matches the strings it generates.

A regular expression is built up of different components, each of which is used to build the regular expression string-generating program. The simplest usable component is the *atom*, but first, you need to understand *ranges*, as atoms are built of these.

Ranges: A range is a sequence of characters contained within "[" and "]" (left and right square brackets). A character matches a range if that character is contained within the range, for example,

[aqcsbvd]

forms the range consisting of the characters "a," "q," "c," "s," "b," "v," and "d." The order of characters is usually not important; however, there are exceptions and these will be noted.

You can specify an ASCII sequence of characters by specifying the first and last characters in that sequence, and separating them with a "-" (hyphen). The above example could also be written as:

[a-dqsv]

To specify "]" (right square brackets) as a character in a range, enter the bracket as the *first* character after the initial left square bracket that starts the range. This example

[]d]

matches "]" (right bracket) and the letter "d."

To include a "-" (hyphen), enter it as either the first or the last character of the range.

You can reverse the matching of the range by including a "^" (caret) at the start of the range. For example,

[^a-dqsv]

matches any letter *except* the ones listed. When using the "^" (caret) with the special rules for including a bracket or hyphen, make the caret the very first character. As an example,

[^]d]

matches anything except a "]" (right square bracket) or the letter "d."

Atoms: Atoms are the most primitive usable part of regular expressions. An atom can be as simple as a single character. The letter "a" is an atom, for example. It is also a very simple regular expression, that is, a program that generates only one string, which is the single-letter string made up of the letter "a." While this may seem trivial, it is important to understand the set of strings that your regular expression program generates. As will be seen in upcoming explanations and examples, much larger sets of strings can be generated from more complex regular expressions.

Certain characters have a special meaning when used as atoms. These are

.	matches any single character
^	matches the null string at the beginning of the input string
$	matches the null string at the end of the input string
character	matches *character*, independent

(Note the new use for the ^symbol).

As an example, the regular expression:

```
^abcd
```

matches "abcd" if, and only if, "abcd" *starts* the full string to be matched.

Whereas

```
[^abcd]
```

is an atom that is a range that matches any single letter, as long as it is *not* the letters "a," "b," "c," or "d."

It was previously stated that a single character string such as the letter "a" is an atom. To remove the special meaning of a character, precede it with a "\" (backslash). For example, this atom

```
$
```

matches the end of a line, but this atom

```
\$
```

matches a $ (dollar sign).

Any character can be preceded with the backslash character with no adverse affect. For example, the string

```
\a
```

matches just the letter "a."

Atoms are also full regular expressions surrounded by parentheses. For example, both "a" and "(a)" are atoms matching the letter "a." This will be important later, as we see patterns to manipulate entire regular expressions.

Pieces: A piece is an atom followed optionally by one of the following symbols:

*	Matches 0 or more sequences of the atom
+	Matches 1 or more sequences of the atom
?	Matches the atom or the null string

Some examples:

a*

matches any number of occurrences of the letter "a," including *none.*

This string

a+

requires there to be at least one letter "a" in the string to be matched.

The string

a?

means that the letter "a" can be there *once,* but it doesn't have to be.

The string

**

matches any number of "*" (asterisks).

Here is an example using parentheses. The string:

(ab)*

matches any number of the two-atom string "ab." As a more complex example, the string:

([A-Za-z][0-9])+

matches one or more instances of letter-digit pairs (but not none, that is, an *empty string* is not a match.)

The order for matches using the optional *, +, or ? symbols is longest construct first. Nested constructs are matched from outside to inside. Concatenated constructs are matched beginning at the left side of the construct.

Branch: A branch is simply a set of zero or more concatenated pieces. The previous letter-digit example was an example of a branch as concatenated pieces. Branches are matched in the order normally read – from left to right. For example, in the previous example, the regular expression matches "A9b3," but not "9Ab3" because the alphabet is given first in the two-atom branch of [A-Za-z][0-9].

Regular Expressions: A regular expression is a branch, or any number of branches separated by a "|" (vertical bar). A string is said to match the regular expression if it is generated by the "program" specified in any of the branches. Of course, a string can be generated by more than one branch. For example, "abc" is generated by all branches in the regular expression

abc|a*(bc)+|(ab)?c|.*

Also remember that if a regular expression can match two different parts of an input string, it will match the earliest part first.

The regular expression support and the technical information for this portion of the documentation is based on Henry Spencer's public domain *regexp(3)* library package. See the example in section 5.2.3. for illustrations of the use of regular expression matching.

5.2.3 Sample X.25 Switching Configuration

The following sample configuration shows how to enable X.25 switching as well as 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 interface serial 1
x25 route 100 cud ^pad$ interface serial 2
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 serial 0 interface. 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 which 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.

6 Addenda, Hints

This chapter contains useful miscellaneous information about using the **cisco** gateway servers which is not necessarily directly related to the 7.0, 7.1, or 8.0 software releases.

6.1 Interface Statistics

The following are explanations of some of the fields in the display of the EXEC **show interface** command.

bandwidth The **cisco** unit's notion of the bandwidth of the line. This number is arbitrarily chosen for DTE serial interfaces. On slow speed lines under any release, and on any high speed line in releases prior to 7.1, the default bandwidth for DTE serial interfaces is is 56KBits/second. For high speed serial lines release 7.1 and above, the default bandwidth is 1.544MBits/second. (Note that this field is used only for the IGRP. Bandwidth is not sensed from the serial line clocks.)

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.

last output The number of hours, minutes, and seconds since the last packet was successfully transmitted by an interface.

last output hang The number of hours, minutes, and seconds 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, the number of days and hours is printed. When that field overflows, asterisks are printed.

output queue length The number of packets currently in the output queue. The maximum number that can be in the queue at one time is shown in parenthesis beside the current queue length.

packets input The total number of error-free packets received by the system.

bytes input The total number of bytes, including data and MAC encapsulation, in the error-free packets received by the system.

no input buffers The number of received packets discarded because there was no buffer space in the main system. Compare with **ignored count**. Broadcast storms on Ethernets and bursts of noise on serial lines are often responsible for **no input buffer** events.

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 1518 bytes is considered a **giant.**

input errors For the Ethernet, includes **runts, giants, no input buffers, crc, frame, overrun,** and **ignored** counts. For a serial interface, includes **no input buffers, runts, giants, crc, frame, overrun, ignored,** and **abort** counts. Other input-related errors can also cause the **input error** count to be incremented, and some datagrams may have more than one error, therefore this sum may not balance with the sum of the 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. 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 non-integer number of octets. On a LAN this is usually the result of collisions or a malfunctioning Ethernet device. On a serial line this is usually the result of noise or other transmission problems.

overrun The number of times the Ethernet or serial receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeds 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 **no input buffer** description. Broadcast storms and bursts of noise can cause the **ignored** count to be incremented.

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. **abort** counts are not applicable to the Ethernet interface.

output packets Total number of messages transmitted by the system.

output bytes Total number of bytes, including data and MAC encapsulation, transmitted by the system.

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.

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 transmit clock, or could be 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 shutdown.

restarts The number of times a Type 2 Ethernet controller was restarted because of errors.

carrier transitions The number of times the carrier detect signal of a serial interface changed state. You are having modem problems if the carrier detect line is changing state often.

6.2 CSC/2 Watchdog Timer Mechanism Problems

cisco Systems has identified a problem in the watchdog timer mechanism on certain CSC/2 Processor cards. The symptoms of the problem are only seen when software Version 7.0 (or later) is running on a system with a suspect processor interface. The symptoms are:

- Rebooting at random intervals of time.
- Rebooting continuously during the boot process. The system clock runs in one-quarter real time, and a reboot will likely occur during a **show configuration** command when non-volatile memory is present.

The customer may visually inspect for the problem by locating the timer chip soldered at card position U97, near the console cable attachment (see Figure 6).

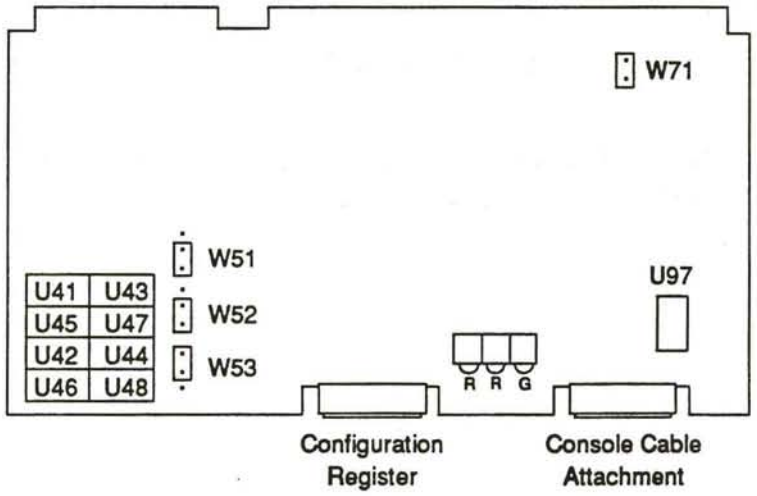


Figure 6: CSC/2 Watchdog Timer Mechanism

If there is an Intel 8254 in that position, then that card will *not* have a problem with the watchdog timer logic. If the chip is an AMD 82C54, the card *will* exhibit the problem. System software Version 7.0 (or later) should not be run on a suspect unit.

A processor with the above problem will be replaced. Contact the **cisco Systems** Customer Engineering group. Have the serial number of the unit available for support personnel.

Note: This problem does *not* appear when running Version 6.2 (or earlier) of the system software.

6.3 Attachment to a Verilink Connect1 DSU/CSU

This section explains to how to attach **cisco Systems** products to a T1 network using the **Verilink** ConnecT1 DSU/CSU.

6.3.1 Description of Configuration

The connection of **cisco** products to leased digital telephone circuits requires a special conversion unit. In North America, this device is commonly referred to as a DSU/CSU (Data Service Unit / Customer Service Unit), which is a "clear-channel T1" circuit. The **Verilink** ConnecT1 is an example of such a unit, and is designed for use with DS1 service provided by most long distance carriers and local phone companies. This amounts to digital service at rates of 1.544 megabits per second.

The ConnecT1 provides a V.35 or RS-449 interface which connects to the phone circuit by way of an RJ-45 connector.

6.3.2 Configuring the cisco unit

The ConnecT1 operates with any **cisco** product equipped with a high-speed serial interface, including either an appropriately configured MCI controller card or a CSC-T card.

cisco Systems offers a variety of possible configurations for its products. Since the ConnecT1 works with either V.35 interface or RS-449 interface, the customer's preference determines which interface will be used on the **cisco** product. Currently, if not specified, **cisco** sends a V.35 interface with all T1-speed-capable-interfaces.

Configuration of the **cisco** product is documented in the *Gateway System Manual.* No switch or jumper changes are required. A male-to-male cable is necessary to join the two units. For optimal results in the connection of cables, both Signal ground (RS-449 pin 19, V.35 pin B) and Chassis ground (RS-449 pin 1, V.35 pin A) should be connected from the ConnecT1 to the **cisco** unit.

6.3.3 Configuring the Verilink ConnecT1

The **Verilink** ConnecT1 offers a variety of possible switch configurations. The arrangement depends more on the T1 circuit itself than on the **cisco** equipment. In most cases, the carrier provides what is referred to as a clear channel T1. This means that the only electronic equipment used on the circuit is repeaters, freeing the user to transmit the data timed to his own clock. The only requirement is that the data conform to a certain ones density. The advantage of the clear channel T1 is that the user can obtain the maximum bandwidth from the circuit.

6.3.4 Connection to Clear Channel T1

cisco provides HDLC framing of all data. This ensures that the data contains a 1 in at least every 8 bits, providing a suitable level of density for use on a clear T1 circuit. The following configuration settings assume this type of circuit. **ON** is the **DOWN** position.

- **S1-4, S1-3, S1-2, S1-1:** ON, ON, ON, ON — Port 0 Data Rate
- **S1-5:** OFF — DTE is in HDLC format
- **S1-6:** OFF — Network is ESF framed
- **S1-7, S1-8:** OFF, OFF — Use Terminal Timing, Port 0

- **S2-4, S2-3, S2-2, S2-1:** ON, ON, ON, ON — Port 1 Data Rate
- **S2-5:** OFF — Alarm Indication Latch
- **S2-6:** OFF — All Alarms reported
- **S2-7, S2-8:** ON, ON — Not Used

- **S3-1:** OFF — DSU detects network ESF yellow alarm
- **S3-2:** OFF — DSU scrambles/descrambles data
- **S3-3:** ON
- **S3-4:** ON — Use 62411 rule for density enforcement
- **S3-5:** ON — Network is AMI
- **S3-6, S3-7, S3-8:** ON, ON, ON — Use Internal Crystal Oscillator

- **S4-1:** ON — Do not assert channel one's DM lead
- **S4-2:** ON — Do not assert channel one's CS lead
- **S4-3:** ON — Do not assert channel one's RR lead
- **S4-4:** ON — Do not assert channel one's RS lead
- **S4-5:** ON — Do not assert channel one's TR lead
- **S4-6, S4-7:** — Not Used
- **S4-8:** — Channel one is V.35

6.3.5 Connection to Network Timed T1

In many cases, the T1 timing is controlled by multiplexing gear of some sort. This is usually the case for circuits which traverse long distances. Typically the device which controls this action is a DACCs or multiplexor panel. Whenever this type of equipment is used, it expects to see certain framing bits in the data pattern. The ConnecT1 must be configured to insert these bits. This also requires the speed to be reduced. As it is frequently difficult to determine the appropriate speed, the most conservative setting is recommended: D4 framing. Assume the data is NOT in HDLC format, and operate at 1.344 megabits per second.

The following are the appropriate switch settings for this mode:

- **S1-4, S1-3, S1-2, S1-1**: OFF, OFF, ON, OFF — Port 0 Data Rate
- **S1-5**: ON — DTE is in NOT HDLC format
- **S1-6**: ON — Network is D4 framed
- **S1-7, S1-8**: OFF, OFF — Use Terminal Timing, Port 0
- **S2-4, S2-3, S2-2, S2-1**: ON, ON, ON, ON — Port 1 Data Rate
- **S2-5**: OFF — Alarm Indication Latch
- **S2-6**: OFF — All Alarms reported
- **S2-7, S2-8**: ON, ON — Not Used
- **S3-1**: ON — DSU does not detect network ESF yellow alarm
- **S3-2**: OFF — DSU does scramble/descramble data
- **S3-3**: ON
- **S3-4**: ON — Use 62411 rule for density enforcement
- **S3-5**: ON — Network is AMI
- **S3-6, S3-7, S3-8**: ON, OFF, ON — Use recovered network clock
- **S4-1**: ON — Do not assert channel one's DM lead
- **S4-2**: ON — Do not assert channel one's CS lead
- **S4-3**: ON — Do not assert channel one's RR lead
- **S4-4**: ON — Do not assert channel one's RS lead
- **S4-5**: ON — Do not assert channel one's TR lead
- **S4-6, S4-7**: — Not Used
- **S4-8**: — Channel one is V.35

6.3.6 cisco Operation

Please consult the *Gateway System Manual* when operating **cisco** equipment. When the T1 is connected, several factors will indicate that the telecommunication gear is working properly. The **show interface** command to an EXEC on the **cisco** equipment shows the state of each interface on the box. Below is typical output from this command:

```
Gateway>show interface serial 0

Serial 0 is up, line protocol is up
  Hardware type is MCI Serial
  Internet address is 1.1.1.1, subnet mask is 255.0.0.0
  MTU 1500 bytes, bandwidth 56 Kbit, delay 20000 usec, rely. 255/255
  Encapsulation is HDLC, loopback is not set, keepalive is set
  Last input 0:00:05, output 0:00:05, output hang never
  Output queue length is 0 (maximum 100)
  Five minute input rate 47 bits/sec, 0 packets/sec
  Five minute output rate 47 bits/sec, 0 packets/sec
     28 packets input, 2278 bytes, 0 no input buffers
     Received 28 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     28 packets output, 2278 bytes, 0 congestion drops
     0 output errors, 0 collisions, 1 interface resets, 0 restarts
     1 carrier transitions
Gateway>
```

The portion *Serial 0 up* indicated that the DSU/CSU is connected and is providing a "carrier up" signal. The *line protocol up* is the result of a keepalive beacon relayed between the two **cisco Systems** boxes. That beacon indicates that the circuit is alive and working.

As further indication that the link is working, a user can use ICMP echoes (pings) generated by one box to ping the other unit. Another useful tool involves placing the DSU/CSU in loopback mode and attempting to ping one's own IP address. This verifies that the **cisco** unit is working correctly with the local DSU/CSU.

Below is the output from two pings from one unit to another using 100 byte packets and 1500 byte packets. This test of multiple-sized frames is useful for distinguishing problems related to sending lots of zeros in the data field.

```
Gateway#ping
Protocol [ip]:
Target IP address: 1.1.1.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 4/4/4 ms
Gateway#p
Protocol [ip]:
Target IP address: 1.1.1.2
Repeat count [5]:
Datagram size [100]: 1500
Timeout in seconds [2]:
Extended commands [n]:
Type escape sequence to abort.
Sending 5, 1500-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 16/21/28 ms


Gateway#
```

6.3.7 Verilink Operation

Please consult the **Verilink** manual for operation of the ConnecT1. The front panel controls offer help in the diagnosis of problems with either the T1 link or connection to the **cisco** unit. By placing the ConnecT1 in DTE loopback, one can verify (through the pinging process described above) that the link to the DSU is working. By using the QRSS test, one can verify that the T1 is working to a first degree. Using the remote inband loop control, one can force the remote ConnecT1 in loopback to allow the testing of pings through the T1 without involving the remote **cisco** unit. The **loopback** configuration command on the **cisco** unit can also be employed to force the local ConnecT1 into DTE loopback by activating the LT signal on the V.35 interface. The latter method is useful when the DSU/CSU is not easily accessible by the operator.

6.4 Attachment to a Cylink 4201 DSU/CSU

This section explains to how to attach **cisco Systems** products to a T1 network using the **Cylink** 4201 DSU/CSU.

6.4.1 Description of Configuration

The connection of **cisco** products to leased digital telephone circuits requires a special conversion unit. In North America, this device is commonly referred to as a DSU/CSU (Data Service Unit / Customer Service Unit), which is a "clear-channel T1" circuit. The **Cylink** 4201 is an example of such a unit, and is designed for use with DS1 service provided by most long-distance carriers and local phone companies. This amounts to digital service at a rate of 1.544 megabits per second.

The 4201 provides a V.35 or RS-449 interface which connects to the **cisco** router. It then attaches to the phone circuit via an RJ-45 or DB-15 connector, via two pairs of wires.

The 4201 operates with any **cisco** product equipped with a high-speed serial interface, including either an appropriately configured MCI/SCI controller card or a CSC-T card.

cisco Systems offers a variety of possible configurations for its products. Since the 4201 works with either V.35 interface or RS-449 interface, the customer's preference determines which interface will be used on the **cisco** product. Currently, if not specified, **cisco** sends a V.35 interface with all T1-speed-capable-interfaces.

Configuration of the **cisco** product is documented in the *Gateway System Manual.* No switch or jumper setting changes are required. A male-to-male cable is necessary to join the two units. For optimal results in the connection of cables, both Signal ground (RS-449 pin 19, V.35 pin B) and Chassis ground (RS-449 pin 1, V.35 pin A) should be connected from the 4201 to the **cisco** unit.

6.4.2 Configuring the Cylink 4201

The **Cylink** 4201 offers a variety of possible configurations. The arrangement depends more on the T1 circuit itself than on the **cisco** equipment. In most cases, the carrier provides what is referred to as a clear channel T1. This means that the only electronic equipment used on the circuit is repeaters, freeing the user to transmit the data timed to his own clock. The only requirement is that the data conform to a certain ones density. The advantage of the clear channel T1 is that the user can obtain the maximum bandwidth from the circuit.

6.4.3 Connection to Clear Channel T1

The following configuration settings assume this type of circuit. All configuration is done via menus on the front panel.

- **Network Framing:** ESF
- **Network DS-1 Mode:** AMI
- **Ones Density Responsibility:** ACSU
- **Ones Density Control:** ESF DL
- **Transmit Clock Source:** INT
- **DIU Frequency:** 1536
- **DIU Transmit Timing:** ST

6.4.4 Connection to Network Timed T1

In many cases, the T1 timing is controlled by multiplexing gear of some sort. This is usually the case for circuits which traverse long distances. Typically the device which controls this action is a DACCs or multiplexor panel. Whenever this type of equipment is used, it expects to see certain framing bits in the data pattern. The 4201 must be configured to insert these bits. This also requires the speed to be reduced. As it is frequently difficult to determine the appropriate speed, the most conservative setting is recommended: D4 framing. Assume the data is NOT in HDLC format, and operate at 1.344 megabits per second. Please note that this scenario has not actually been tested by **cisco.**

The following are the appropriate switch settings for this mode:

- **Network Framing:** D4
- **Network DS-1 Mode:** AMI
- **Ones Density Responsibility:** ACSU
- **Ones Density Control:** ESF DL
- **Transmit Clock Source:** NET
- **DIU Frequency:** 1344
- **DIU Transmit Timing:** ST

6.4.5 cisco Operation

equipment. When the T1 line is connected, several factors will indicate that the telecommunications gear is working properly. The **show interface** command to an EXEC on the **cisco** equipment shows the state of each interface on the router. Below is typical output from this command:

```
Gateway>show interface serial 0

Serial 0 is up, line protocol is up
  Hardware type is MCI Serial
  Internet address is 1.1.1.1, subnet mask is 255.0.0.0
  MTU 1500 bytes, bandwidth 56 Kbit, delay 20000 usec, rely. 255/255
  Encapsulation is HDLC, loopback is not set, keepalive is set
  Last input 0:00:05, output 0:00:05, output hang never
  Output queue length is 0 (maximum 100)
  Five minute input rate 47 bits/sec, 0 packets/sec
  Five minute output rate 47 bits/sec, 0 packets/sec
     28 packets input, 2278 bytes, 0 no input buffers
     Received 28 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     28 packets output, 2278 bytes, 0 congestion drops
     0 output errors, 0 collisions, 1 interface resets, 0 restarts
     1 carrier transitions
Gateway>
```

The portion `Serial 0 is up` indicated that the DSU/CSU is connected and is providing a "carrier up" signal. The `line protocol is up` is the result of a keepalive beacon relayed between the two **cisco Systems'** routers. That beacon indicates that the circuit is alive and working.

As further indication that the link is working, a user can use ICMP echoes (pings) generated by one router to ping the other unit. Another useful tool involves placing the DSU/CSU in loopback mode and attempting to ping one's own IP address. This verifies that the **cisco** unit is working correctly with the local DSU/CSU.

Below is the output from two pings from one unit to another using 100 byte packets and 1500 byte packets. This test of multiple-sized frames is useful for distinguishing problems related to sending lots of zeros in the data field.

```
Gateway#ping
```

```
Protocol [ip]:
Target IP address: 1.1.1.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 4/4/4 ms
Gateway#p
Protocol [ip]:
Target IP address: 1.1.1.2
Repeat count [5]:
Datagram size [100]: 1500
Timeout in seconds [2]:
Extended commands [n]:
Type escape sequence to abort.
Sending 5, 1500-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 16/21/28 ms


Gateway#
```

6.4.6 Cylink Operation

Please consult the **Cylink** manual for operation of the 4201. The front panel controls offer help in the diagnosis of problems with either the T1 link or connection to the **cisco** unit. By placing the 4201 in local loopback, one can verify (through the pinging process described above) that the link to the DSU is working. By using the QRSS test, one can verify that the T1 is working to a first degree. Using the remote inband loop control, one can force the remote 4201 in loopback to allow the testing of pings through the T1 without touching the remote **cisco** unit.

6.5 Attachment to a Cylink CIDEC-HS Encrypter

This section explains to how to attach **cisco Systems** products to a **Cylink** CIDEC-HS Encryption Unit.

6.5.1 Description of Configuration

The bulk encryption of data across a T1 line is very common in the government sector, and becoming more prevalent in the commercial sector. Encryption requires

a pair of encryption devices, one at each end of the link.

The **Cylink** CIDEC-HS is an example of a DES-based encryption unit. It can be configured to perform key management in one of two different styles, government or commercial mode.

The HS provides a V.35 or RS-449 interface which connects to the **cisco** router or gateway. It then attaches to a T1 CSU/DSU via another V.35 or RS-449 connector.

6.5.2 Configuring the cisco unit

The HS operates with any **cisco** product equipped with a high-speed serial interface, including either an appropriately configured MCI/SCI controller card or a CSC-T card.

cisco Systems offers a variety of possible configurations for its products. Since the HS works with either V.35 interface or RS-449 interface, the customer's preference determines which interface will be used on the **cisco** product. Currently, if not specified, **cisco** sends a V.35 interface with all T1-speed-capable-interfaces.

Configuration of the **cisco** product is documented in the *Gateway System Manual.* No switch or jumper setting changes are required. A male-to-male cable is necessary to join the two units. For optimal results in the connection of cables, both Signal ground (RS-449 pin 19, V.35 pin B) and Chassis ground (RS-449 pin 1, V.35 pin A) should be connected from the HS to the **cisco** unit.

6.5.3 Configuring the Cylink CIDEC-HS

The **Cylink** CIDEC-HS offers a variety of possible configurations. The configuration depends more on the encryption mode used and the circuit connecting the two encrypters, than on the **cisco** equipment.

6.5.4 Connection to Equipment

The following configuration settings should be used for all connections to **cisco Systems** equipment. All other parameters should be set for the encryption mode, and the characteristics of the T1 circuit. All configuration is done via menus on the front panel.

- **Interface Attributes: TX Clock Source**: DCE Timed
- **Interface Attributes: TX Clock Polarity**: Normal
- **Interface Attributes: RX Clock Polarity**: Normal

On a clear-channel T1, using the SEEK encryption mode, the following settings were used in the in-house testing.

- **Auto DEK: Period of Change**: Daily
- **Auto DEK: Starting Hour**: 0
- **Auto DEK: Starting Minute**: 0
- **Auto DEK: Initial Delay**: Disabled
- **Interface Attributes: End to End Delay**: 0.4 sec
- **Interface Attributes: Resync Holdoff**: 0.9 sec
- **Miscellaneous: Master/Slave**: Yes
- **Miscellaneous: Master/Slave Unit**: Master (Slave for 2nd unit)

6.5.5 cisco Operation

Please consult the *Gateway System Manual* when operating **cisco** equipment. When the HS encrypter is connected, several factors will indicate that the telecommunications gear is working properly. The **show interface** command to an EXEC on the **cisco** equipment shows the state of each interface on the router. Below is typical output from this command:

```
Gateway>show interface serial 0

Serial 0 is up, line protocol is up
  Hardware type is MCI Serial
  Internet address is 1.1.1.1, subnet mask is 255.0.0.0
  MTU 1500 bytes, bandwidth 56 Kbit, delay 20000 usec, rely. 255/255
  Encapsulation is HDLC, loopback is not set, keepalive is set
  Last input 0:00:05, output 0:00:05, output hang never
  Output queue length is 0 (maximum 100)
  Five minute input rate 47 bits/sec, 0 packets/sec
  Five minute output rate 47 bits/sec, 0 packets/sec
     28 packets input, 2278 bytes, 0 no input buffers
     Received 28 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     28 packets output, 2278 bytes, 0 congestion drops
     0 output errors, 0 collisions, 1 interface resets, 0 restarts
     1 carrier transitions
Gateway>
```

The portion `Serial 0 is up` indicated that the HS is connected and is providing a "carrier up" signal. The `line protocol is up` is the result of a keepalive beacon relayed between the two **cisco** routers. That beacon indicates that the circuit is alive and working.

As further indication that the link is working, a user can use ICMP echoes (pings) generated by one router to ping the other unit. Below is the output from two pings from one unit to another using 100 byte packets and 1500 byte packets. This test of multiple-sized frames is useful for distinguishing problems related to sending lots of zeros in the data field.

```
Gateway#ping
Protocol [ip]:
Target IP address: 1.1.1.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 4/4/4 ms
Gateway#p
Protocol [ip]:
Target IP address: 1.1.1.2
Repeat count [5]:
Datagram size [100]: 1500
Timeout in seconds [2]:
Extended commands [n]:
Type escape sequence to abort.
Sending 5, 1500-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 16/21/28 ms


Gateway#
```

6.5.6 Cylink Operation

Please consult the **Cylink** manual for operation of the CIDEC-HS. The front panel controls offer help in the diagnosis of problems with either the T1 link or connection to the **cisco** unit.

To begin the transmission of data by the CIDEC-HS, it must be put into **secure** or **bypass** mode. In order to enter secure mode, the HS must first be told to generate keys. Pressing the *CKEK* button and following the prompts generates an initial set of keys. The *secure* button then begins the transmission of encrypted data. Bypass (unencrypted) mode is simply entered by pressing the *bypass* key on the Master unit.

Addendum to Gateway Server 8.0 Release Note

December 18, 1989

Information in this addendum augments information in Version C of the *Gateway Server 8.0 Release Note* dated December 11, 1989. This addendum describes a problem found in the 8.0(9) software release, and introduces the 8.0(10) software release.

Problem: A problem in the bridging software was discovered after the 8.0(9) software release.

Solution: In observing the bridging software it was found that bridging fails if the port being used for bridging is disconnected or shut down. The 8.0(10) software release contains software that corrects this problem.

Description of Part

Figure 1 illustrates the placement of the EPROMs on the CSC/1 Processor card.

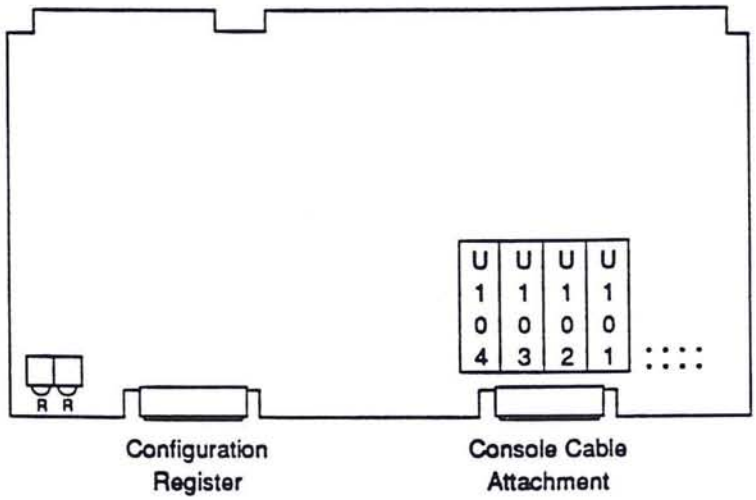


Figure 1: CSC/1 EPROM Placement

Installation

This document describes EPROM replacement procedures for the CSC/1 Processor card.

The software comes in a set of four EPROMs. The EPROMs will be inserted into sockets U101, U102, U103, and U104, which are located near the console cable connector.

The EPROMs are labeled with the appropriate socket U-number. The sockets have labels, although obscure, on the silk screen portion of the processor card.

The correct placement of the EPROMs is crucial. If improperly positioned, it may be damaged when the system is powered on. Each EPROM has a notch cut in one side to indicate its proper orientation. Each EPROM must be placed so that its notch faces the console cable attachment as shown in Figure 1. Note that all the other ICs in the card have notches and that the notches all face the same direction. Do *not* rely on the orientation of the EPROM labels.

Note 1: To prevent damage to the EPROMs from electro-static discharge, observe proper grounding techniques when handling the cards and their components.

Note 2: Be sure to turn off power to the system before gaining access to and removing any cards.

Follow these steps to upgrade the EPROMs in the CSC/1 Processor card.

- Step 1. Turn off power to the server, then gain access to the card cage.

  Cards in the A-chassis and M-chassis can be accessed by removing front or side panels. Access to the C-chassis requires removing the top cover.

- Step 2. Remove the processor card from the card cage. All cards have ejectors that allow them to be easily extracted from their slots. Use your thumbs tc pull the ejectors out and away from the card. Open both ejectors at the same time. Use care not to strain any flat cables still attached to the card as it is being removed.

- Step 3. Remove the EPROM from its socket.

- Step 4. Insert the new EPROM into the socket.

  When inserting an EPROM into a socket, be very careful not to bend or crush any of its pins. If this happens, use a needle nose pliers to straighten them out.

- Step 5. Repeat steps 3 and 4 for each EPROM.

- Step 6. When all EPROMS have been replaced, turn on power and test the installation.

  If you power up a system when one or more of the EPROMs is incorrectly inserted, the system may either halt and light its red halt light or may print a message on the console port complaining of a checksum error. When this happens, locate the offending EPROM pin, straighten it, then reinsert the EPROM and try again.

- Step 7. When the CSC/1 Processor card tests successfully, replace the card in the card cage, then replace the panel or top cover.

  To insert the card, push the card firmly in the card cage until it snaps in place and is firmly seated in the slot. The ejectors close automatically when the card is properly seated.

This completes the EPROM replacement procedure.

Description of Part

Figure 1 illustrates the placement of the EPROMs on the CSC/2 Processor card.

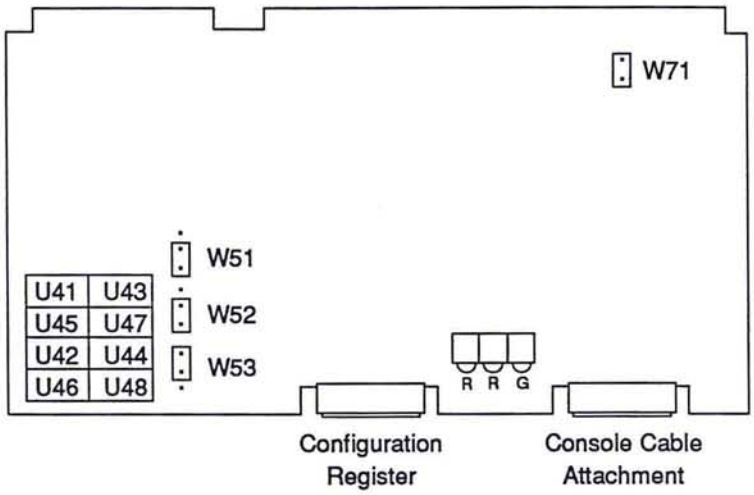


Figure 1: CSC/2 EPROM Placement

Installation

This document describes EPROM replacement procedures for the CSC/2 Processor card.

If the software comes in a set of four EPROMs, the EPROMs are inserted into sockets U42, U44, U46, and U48 only. If the software comes in a set of eight EPROMs, the EPROMs will be inserted into all eight sockets.

The EPROMs are labeled with the appropriate socket U-number. The sockets have labels, although obscure, on the silk screen portion of the processor card.

The correct placement of the EPROMs is crucial. If improperly positioned, it may be damaged when the system is powered on. Each EPROM has a notch cut in one side to indicate its proper orientation. Each EPROM must be placed so that its notch faces the left side of the card as shown in Figure 1. Note that all the other ICs in the card have notches and that the notches all face the same direction. Do *not* rely on the orientation of the EPROM labels.

Note 1: To prevent damage to the EPROMs from electro-static discharge, observe proper grounding techniques when handling the cards and their components.

Note 2: Be sure to turn off power to the system before gaining access to and removing any cards.

Follow these steps to upgrade the EPROMs in the CSC/2 Processor card.

- **Step 1.** Turn off power to the server, then gain access to the card cage.

  Cards in the A-chassis and M-chassis can be accessed by removing front or side panels. Access to the C-chassis requires removing the top cover.

- **Step 2.** Remove the processor card from the card cage. All cards have ejectors that allow them to be easily extracted from their slots. Use your thumbs to pull the ejectors out and away from the card. Open both ejectors at the same time. Use care not to strain any flat cables still attached to the card as it is being removed.

- **Step 3.** Remove the EPROM from its socket.

- **Step 4.** Insert the new EPROM into the socket.

  When inserting an EPROM into a socket, be very careful not to bend or crush any of its pins. If this happens, use a needle nose pliers to straighten them out.

- **Step 5.** Repeat steps 3 and 4 for each EPROM.

- **Step 6.** When all EPROMS have been replaced, replace the card in the card cage, turn on power and test the installation.

  If you power up a system when one or more of the EPROMs is incorrectly inserted, the system may either halt and light its red halt light or may print a message on the console port complaining of a checksum error. When this happens, locate the offending EPROM pin, straighten it, then reinsert the EPROM and try again.

- **Step 7.** When the CSC/2 Processor card tests successfully, replace the panel or top cover.

  To insert the card, push the card firmly in the card cage until it snaps in place and is firmly seated in the slot. The ejectors close automatically when the card is properly seated.

This completes the EPROM replacement procedure.