CHAPTER 6

Configuring Interfaces

Use the information in this chapter to understand the types of interfaces supported on our communication servers. Our communication servers support two types of interfaces: physical and virtual interfaces. The physical types of interfaces you have depend on the appliques or interface processors (IPs) you have. The virtual interfaces our communication servers support include subinterfaces and IP tunnels.

For hardware technical descriptions and information about installing the communication server interfaces, refer to the hardware installation and maintenance publication for your particular product. For command descriptions and usage information, refer to the *Communication Server Command Reference* publication.

Interface Configuration Task List

This section lists the tasks you can perform to configure and maintain the interfaces supported on your communication servers. The first section introduces material that you might need to know in advance of the other tasks.

- Understand Supported Interfaces and Encapsulations
- Interface Configuration Process
- Configure the Interface Type
- Add a Description for an Interface
- Configure Subinterfaces
- Understand Tunneling
- Configure IP Tunneling
- Reenable HDLC Serial Encapsulation
- Select the Ethernet Encapsulation
- Enable MOP
- Enable MOP Message Support
- Select the Token Ring Speed
- Enable Early Token Release
- Configure Token Ring Interfaces in Source-Route Bridging Environments
- Configure the Point-to-Point Protocol, including the Challenge Handshake Authentication Protocol (CHAP), Link Quality Monitoring (LQM) and PPP Magic Number Support.

- Configure Dial Backup Service
- Configure Loopback Detection
- Set Transmit Delay (synchronous serial interfaces only)
- Configure DTR Signal Pulsing (synchronous serial interfaces only)
- Configure the Clock Rate on DCE Appliques (synchronous serial interfaces only)
- Control Interface Hold-Queue Limits
- Set Bandwidth
- Set Interface Delay
- Limit Size of the Transmit Queue
- Adjust Maximum Packet Size or MTU Size
- Monitor and Maintain the Interface

Understand Supported Interfaces and Encapsulations

The following sections describe the interfaces and encapsulations supported by our communication servers.

Synchronous Serial

Support for the synchronous serial interface is supplied on the following serial network interface cards or systems:

- The Multiport Communications Interface (CSC-MCI), a single card that provides up to two high-speed synchronous serial port connectors that support RS-232, V.35, RS-449, and X.21 connections
- The Serial Port Communications Interface (CSC-SCI), a single card that provides up to four high-speed serial ports that support RS-232, V.35, RS-449, and X.21 connections

The MCI and SCI cards can query the appliques to determine their types for use in reports displayed by the EXEC **show** commands. However, they do so only at system startup, so the appliques must be attached when the system is started. Use the **show interfaces** and **show controllers mci** EXEC commands to display the serial port numbers. These commands provide a report for each interface supported by the communication server.

Synchronous Serial Encapsulation Methods

By default, synchronous serial lines use the High-level Data Link Control (HDLC) serial encapsulation method, which provides the synchronous framing and error detection functions of HDLC without windowing or retransmission. The synchronous serial interfaces support the following serial encapsulation methods:

- Asynchronous Transfer Mode-Data Exchange Interface (ATM-DXI)
- High-level Data Link Control (HDLC)
- Frame Relay
- Point-to-Point Protocol (PPP)
- Switched Multimegabit Data Services (SMDS)

- X.25-based encapsulations

Encapsulation methods are set according to the type of protocol or application you configure on your communication server. ATM-DXI is described later in this chapter. HDLC is described later in this chapter in the section "Reenable HDLC Serial Encapsulation." PPP is described later in this chapter in the section "Configure the Point-to-Point Protocol." The remaining methods are described in their respective chapters describing the protocols or applications. Serial encapsulation methods are also discussed in the *Communication Server Command Reference* publication, under the **encapsulation** command.

Asynchronous Serial

There are two asynchronous serial encapsulation methods:

- SLIP
- Asynchronous PPP

See the "Configuring SLIP and PPP" chapter for more information about these encapsulation methods.

Ethernet

Support for the Ethernet interface is supplied on one of the following Ethernet network interface cards or systems:

- The Multiport Communications Interface (MCI) card, which provides one Ethernet connector compatible with Ethernet Versions 1 and 2 and the IEEE 802.3 protocol
- The Multiport Ethernet Controller (CSC-MEC) interface card, which provides two, four, or six high-speed Ethernet connectors compatible with Ethernet Versions 1 and 2 and the IEEE 802.3 protocol

Use the **show interfaces** and **show controllers mci** EXEC commands to display the Ethernet port numbers. These commands provide a report for each interface supported by the communication server.

Ethernet Encapsulation Methods

Currently, there are three common Ethernet encapsulation methods:

- The standard Ethernet Version 2.0 encapsulation, which uses a 16-bit protocol type code
- The IEEE 802.3 encapsulation, in which the type code becomes the frame length for the IEEE 802.2 LLC encapsulation (destination and source Service Access Points, and a control byte)
- The SNAP method, as specified in RFC 1042, which allows Ethernet protocols to run on IEEE 802.2 media

The encapsulation method you use depends upon the type of Ethernet media connected to the communication server and the routing application you configure. Further detail is provided in the sections "Select the Ethernet Encapsulation" later in this chapter. See also the chapters describing the protocols or applications.

Token Ring

Support for the Token Ring interface is supplied on one of our Token Ring network interface cards:

- The 4/16-Mbps Token Ring cards, which interconnect network servers to IEEE 802.5 and IBM-compatible Token Ring media at speeds of 4 or 16 Mbps. The 4/16-Mbps cards or CSC-R16M, CSC-1R, and CSC-2R (dual Token Ring card).

  Support for the Token Ring MIB variables is provided as described in RFC 1231, "IEEE 802.5 Token Ring MIB," by K. McCloghrie, R. Fox, and E. Decker, May 1991. The mandatory Interface Table and Statistics Table are implemented, but the optional Timer Table of the Token Ring MIB is not. The Token Ring MIB has been implemented for the TRIP.

Use the **show interfaces** and **show controllers token** EXEC commands to display the Token Ring numbers. These commands provide a report for each ring supported by the communication server.

Note If the system receives an indication of a cabling problem from a Token Ring interface, it puts that interface into a reset state and does not attempt to restart it. It functions this way because periodic attempts to restart the Token Ring interface have a drastic impact on the stability of protocol routing tables. Once you have replugged the cable into the MAU, restart the interface by typing the command **clear interface tokenring** *number,* where *number* is the interface number.

Token Ring Encapsulation Methods

The Token Ring interface by default uses the SNAP encapsulation format defined in RFC 1042. It is not necessary to define an encapsulation method for this interface.

Interface Configuration Process

When you configure the interface, identify the interface type followed by the number of the connector or interface card. These numbers are assigned at the factory at the time of installation, or when added to a system, and can be displayed with the **show interfaces** EXEC command. A report is provided for each interface supported by the communication server, as seen in the following partial sample display:

```
Serial 0 is administratively down, line protocol is down
  Hardware is MCI Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
```

Use the **show hardware** EXEC command to see a list of the system software and hardware.

For example, to begin configuring interface Serial 0, you would add the following line to the configuration file:

```
interface serial 0
```

Note It is not necessary to add a space between the interface type and interface number. For example, you can specify either serial 0 or serial0.

The **interface** global configuration command is then followed by interface configuration commands, which define the protocols and applications that will run on this interface. Remain in interface configuration mode as long as you are entering interface commands for a routing protocol. This mode ends when you enter a command that is not an interface configuration command or when you type the Ctrl-Z sequence to get out of configuration mode and return to privileged EXEC mode.

Configure the Interface Type

Begin interface configuration in global configuration mode. To configure an interface, follow these steps:

- Enter the **configure** EXEC command at the privileged EXEC prompt.
- Once in the global configuration mode, start configuring the interface by entering the **interface** command. Follow each **interface** command with the interface commands your particular interface requires. The commands are collected and applied to the **interface** command until you enter another **interface** command or a command that is not an interface configuration command.
- Once an interface is configured, you can check its status by entering the EXEC **show** commands described after the task tables that follow.

The following sections show each interface type as a separate task. For examples of interface configurations, see the section "Example of Enabling Interface Configuration" at the end of this chapter.

Configure an Asynchronous Serial Interface

To configure an asynchronous interface, see the "Configuring SLIP and PPP" chapter in this manual. To use SLIP or PPP to make a connection, see the *Communication Server and Protocol Translator Connection Guide*.

Configure a Dialer Interface

To specify a dialer rotary group leader, perform the following task:

Task	Command
Begin dialer rotary group interface configuration.	**interface dialer** *interface-number*

Configure an Ethernet Interface

To configure an Ethernet interface, perform the following task

Task	Command
Begin Ethernet interface configuration.	**interface ethernet** *interface-number*

Configure a Loopback Interface

You can specify a software-only interface called a loopback interface that emulates an interface that is always up. It is supported on all platforms. A loopback interface is a virtual interface that allows BGP and RSRB sessions to stay up even if the outbound interface is down.

You can use the loopback interface as the termination address for BGP sessions or for establishing a Telnet session from the communication server's console to its auxiliary port when all other interfaces are down. In applications where other communication servers will attempt to reach this loopback interface, you should configure a routing protocol to distribute the subnet assigned to the loopback address.

Packets routed to the loopback interface are rerouted back to the box and processed locally. IP packets routed out the loopback interface but not destined to the loopback interface are dropped. This means the loopback interface does double duty as the Null0 interface.

To configure a loopback interface, perform the following task:

Task	Command
Begin loopback interface configuration.	**interface loopback** *interface-number*

Configure a Null Interface

The communication server supports a "null" interface. This pseudo-interface functions similarly to the null devices available on most operating systems. This interface is always up and can never forward or receive traffic; encapsulation always fails. The only interface configuration command that you can specify for the null interface is **no ip redirects**.

The null interface provides an alternative method of filtering traffic. The overhead involved with using access lists can be avoided by directing undesired network traffic to the null interface.

To specify the null interface, perform the following task in global configuration mode:

Task	Command
Begin null interface configuration.	**interface null 0**

Specify null 0 (or null0) as the interface name and unit. The null interface can be used in any command that has an interface type as an argument. The following example configures a null interface for IP route 127.0.0.0:

```
ip route 127.0.0.0 255.0.0.0 null 0
```

Configure a Synchronous Serial Interface

To configure a synchronous serial interface, perform the following task:

Task	Command
Begin synchronous serial interface configuration	**interface serial** *interface-number*

Configure a Token Ring Interface

To configure a Token Ring interface, perform the following task:

Task	Command
Begin Token Ring interface configuration	**interface tokenring** *interface-number*

Configure a Tunnel Interface

To configure a tunnel interface, you will use the **interface tunnel** command. Before you configure a tunnel interface, see the sections in this chapter entitled "Understand Tunneling" and "Configure IP Tunneling."

Task	Command
Begin tunnel interface configuration.	**interface tunnel** *interface-number*

Add a Description for an Interface

You can add a description about an interface to help you remember what is attached to it. This entry is meant solely as a comment to help identify what the interface is being used for. The description will appear in the output of the following commands: **show configuration**, **write terminal**, and **show interfaces**. To add the description, complete the following task in interface configuration mode:

Task	Command
Add a description for an interface.	**description** *string*

For examples of this command, see the examples at the end of this chapter.

Configure Subinterfaces

Configuring multiple virtual interfaces, or subinterfaces, on a single physical interface allows greater flexibility and connectivity on the network. With subinterfaces, you can provide full connectivity on partially meshed Frame Relay networks. You can also configure subinterfaces on LANs to support multiple IPX encapsulations on the same physical interface.

You can perform the following subinterface configuration tasks:

- Understand which interface types and encapsulations the subinterfaces support
- Configure subinterfaces on serial interfaces running Frame Relay encapsulation
- Configure subinterfaces on Token Ring and Ethernet interfaces to support multiple IPX encapsulations

Understand Supported Interfaces and Encapsulations

Subinterfaces can be used in the following situations:

- To support partially meshed multiprotocol frame relay networks over a serial interface
- To support multiple simultaneous IPX encapsulations on Token Ring and Ethernet interfaces

Table 6-1 lists the commands that are supported on subinterfaces.

Table 6-1 Subinterface Configuration Commands

Command	Interface Type Supported
bandwidth	Serial, Ethernet, Token Ring
delay	Serial, Ethernet, Token Ring
description	Serial, Ethernet, Token Ring

Command	Interface Type Supported
exit	Serial, Ethernet, Token Ring
frame-relay	Serial only
ip	Serial only
ipx	Serial, Ethernet, Token Ring
ntp	Serial only
shutdown	Serial, Ethernet, Token Ring

See the system management chapter for more information about NTP.

The commands listed in Table 6-2 support subinterfaces as parameters.

Table 6-2 Commands That Allow Subinterfaces as Parameters

Command	Command Type
ip unnumbered	Interface configuration
tunnel source	Interface configuration
interface	Global configuration
ip route	Global configuration
routemap match interface	Route-map configuration
distribute-list	Router configuration
neighbor address update-source	Router configuration
passive-interface	Router configuration

Example

In the following example, the route to IP network 10.0.0.0 is configured to exit the communication server via subinterface serial 0.1:

```
ip route 10.0.0.0 255.0.0.0 serial 0.1
```

The **show** commands listed in Table 6-3 support subinterfaces as parameters.

Table 6-3 Show Commands that Allow Subinterfaces as Parameters

Command	Command Type
show buffers	EXEC
show interfaces	EXEC
show ip igrp2 neighbors	EXEC
show ip ospf neighbor	EXEC
show ip ospf interface	EXEC
show ip irdp	EXEC
show ip interface	EXEC
show novell	EXEC
show protocols	EXEC

Configure Subinterfaces on Serial Interfaces Running Frame Relay Encapsulation

Frame Relay networks provide multiple point-to-point links, or PVCs (permanent virtual circuits), through the same physical serial interface. Subinterfaces allow blocks of one or more Frame Relay PVCs to be treated as separate subnetworks. Protocols such as IP, IPX, and bridging view each subinterface as a separate interface with its own address and protocol assignments.

A subinterface with a single Frame Relay PVC is modeled as a point-to-point link. A subinterface with multiple Frame Relay PVCs is modeled as a LAN.

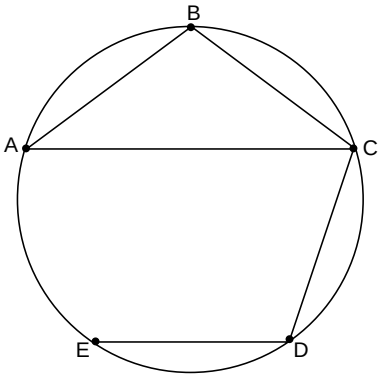
Subinterfaces provide a mechanism for supporting partially meshed Frame Relay networks. In the past, a single network number (such as an IP subnet or an IPX network number) was assigned to an entire Frame Relay network. Most protocols assume transitivity on a logical network; that is, if station A can talk to station B, and station B can talk to station C, then station A should be able to talk to station C directly. This is true on LANs, but is not true on Frame Relay networks unless they are fully meshed.

Subinterfaces address these limitations by providing a way to subdivide a partially meshed Frame Relay network into a number of smaller, fully meshed (or point-to-point) subnetworks. Each subnetwork is assigned its own network number and appears to the protocols as if it is reachable through a separate interface. (Note that point-to-point subinterfaces may be unnumbered for use with IP, reducing the addressing burden that might otherwise result.)

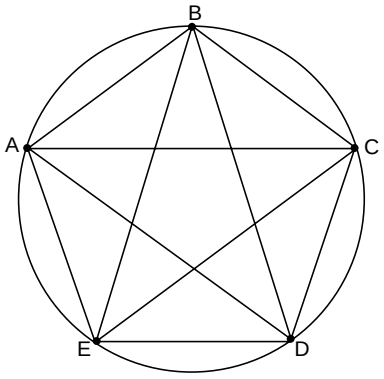
For example, suppose you have a five-node Frame Relay network (see the first part of Figure 1-1) that is partially meshed. If the entire network is viewed as a single subnetwork (with a single network number assigned), most protocols assume that node A can transmit a packet directly to node E, when in fact it must be relayed through nodes C and D. This can be made to work with certain protocols (for example, IP) but will not work at all with other protocols (for example, AppleTalk) because nodes C and D will not relay the packet out the same interface on which it was received. The only way to make this work fully is to create a fully meshed network (see the second part of Figure 1-1), but that requires a large number of PVCs, which may not be economically feasible.

Using subinterfaces, the Frame Relay network can be subdivided into three smaller networks (see the third part of Figure 1-1) with separate network numbers. Nodes A, B, and C are connected to a fully meshed network, and nodes C and D, as well as nodes D and E are connected via point-to-point networks. In this configuration, nodes C and D would see two subinterfaces, allowing them to forward packets without violating split horizon rules.

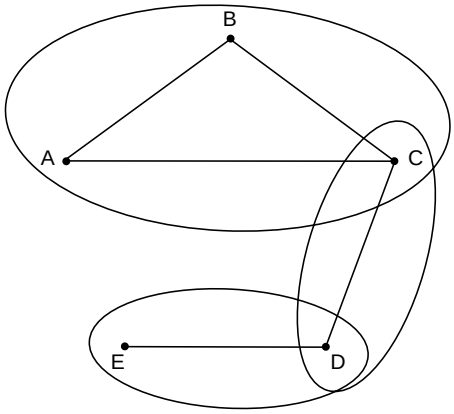
Figure 1-1 Using Subinterfaces to Provide Full Connectivity on a Partially Meshed Frame Relay Network



Network A: Partially Meshed Frame Relay
Network without Full Connectivity



Network B: Fully Meshed Frame Relay
Network with Full Connectivity



Network C: Partially Meshed Frame Relay Network with Full Connectivity
(configuring subinterfaces)

S1528a

To configure a subinterface, perform the following tasks:

Task		Task
Step 1	Configure a serial interface	**interface serial** *interface-number*
Step 2	Configure Frame Relay encapsulation on the serial interface.	**encapsulation frame-relay**
Step 3	Configure a subinterface.	**interface serial** *interface-numbe.subinterface-number* [**multipoint** | **point-to-point**]
Step 4	Configure the feature you want the subinterface to support.	See the examples following this table.

Example

In the following example, subinterface 1 models a point-to-point subnet and subinterface 2 models a broadcast subnet:

```
interface serial 0
encapsulation frame-relay
interface serial 0.1 point-to-point
ip address 10.0.1.1 255.255.255.0
frame-relay interface-dlci 42

interface serial 0.2 multipoint
ip address 10.0.2.1 255.255.255.0
frame-relay map 10.0.2.1 255.255.255.0 17 broadcast
frame-relay map 10.0.2.2 255.255.255.0 18
```

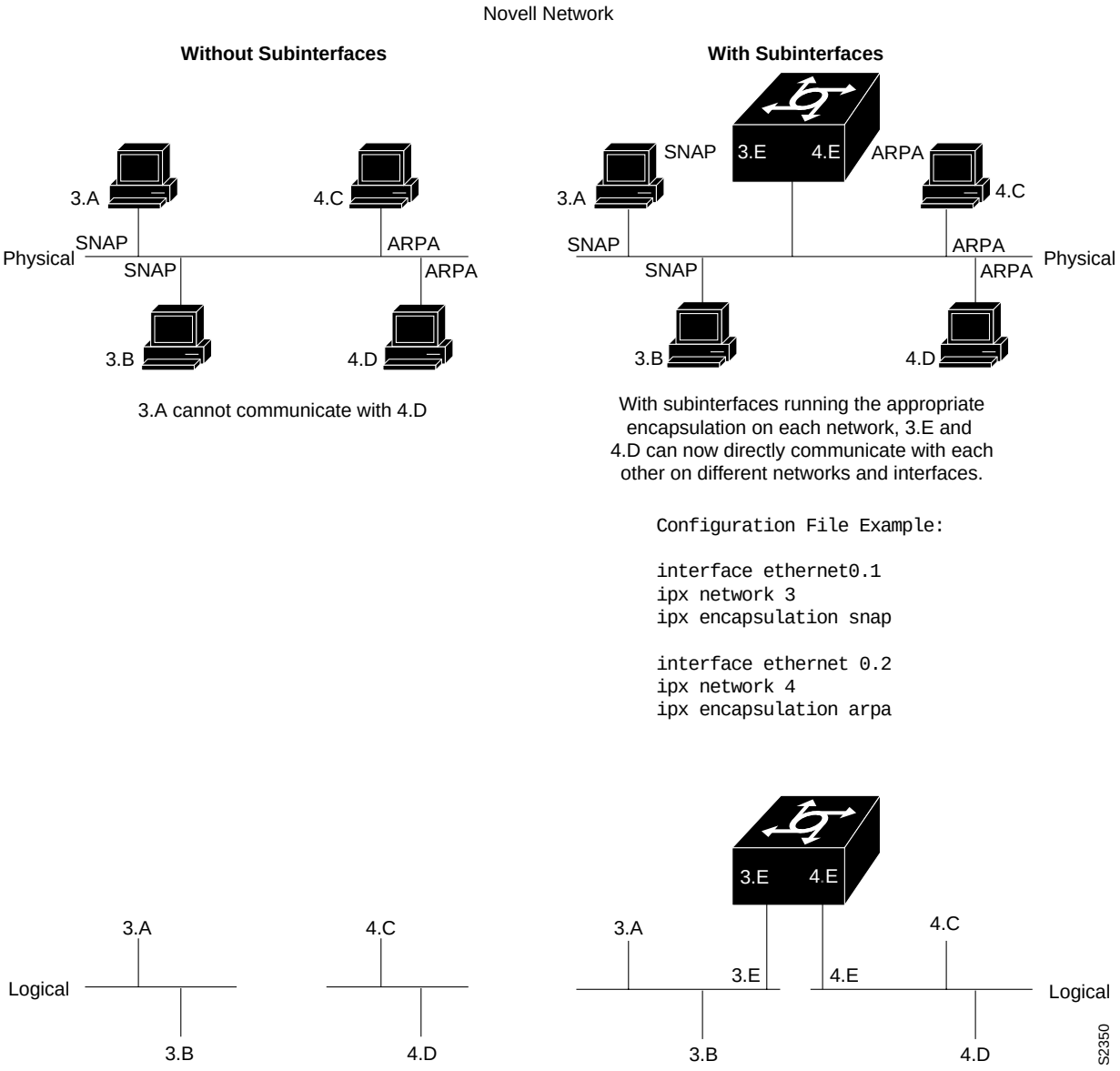
Configure Subinterfaces on Token Ring and Ethernet Interfaces Running Multiple IPX Encapsulations

Another use for subinterfaces is to support the multiple IPX encapsulations on LAN media. There are a number of ways to encapsulate IPX packets on each LAN medium, and hosts that use different encapsulation techniques cannot communicate with each other. Logically, they are on different networks and separate IPX network numbers are assigned accordingly. The left side of Figure 1-2 illustrates this situation.

However, a communication server can be placed on the LAN, and a separate subinterface can be created for each encapsulation type (with distinct IPX network numbers). The communication server will then switch packets between the two logical networks, converting the encapsulations as necessary. Since separate subinterfaces are used, split horizon rules are not violated.

The right side of Figure 1-2 illustrates the use of subinterfaces on LAN interfaces that use multiple IPX encapsulation methods.

Figure 1-2 Novell Network with and without Subinterfaces



Understand Tunneling

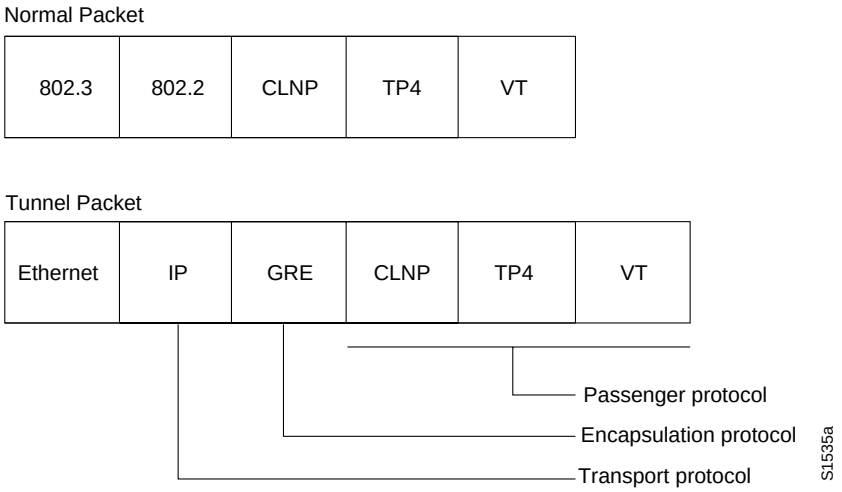
Tunneling provides a way to encapsulate arbitrary packets inside of a transport protocol. This feature is implemented as a virtual interface to provide a simple interface for configuration. The tunnel interface is not tied to specific “passenger” or “transport” protocols, but rather, it is an architecture that is designed to provide the services necessary to implement any standard point-to-point encapsulation scheme.

Tunneling has three primary components:

- Passenger protocol, which is the protocol you are encapsulating (IPX or IP)
- Carrier protocol, which is one of the following encapsulation protocols:
  - Generic Router Encapsulation (GRE), Cisco's multiprotocol carrier protocol
  - Cayman, a proprietary protocol for AppleTalk over IP
  - NOS, IP over IP compatible with the popular KA9Q program
- Transport protocol, which is the protocol used to carry the encapsulated protocol (IP only)

Figure 1-3 illustrates IP tunneling terminology and concepts.

Figure 1-3 IP Tunneling Terminology and Concepts

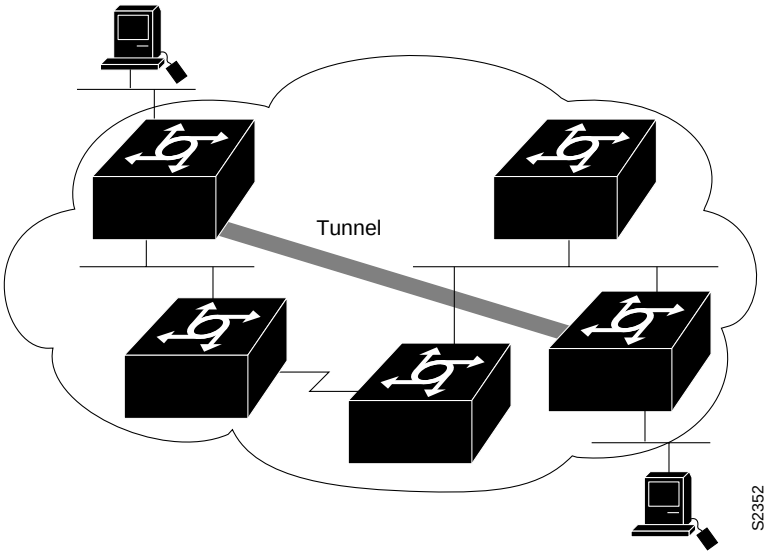


Advantages of Tunneling

There are several situations where encapsulating traffic in another protocol is useful:

- To provide multiprotocol local networks over a single-protocol backbone
- To provide workarounds for networks containing protocols that have limited hop counts; for example, AppleTalk (see Figure 1-4)
- To connect discontinuous subnetworks
- To allow virtual private networks across Wide-Area Networks (WANs)

Figure 1-4 Providing Workarounds for Networks with Limited Hop Counts



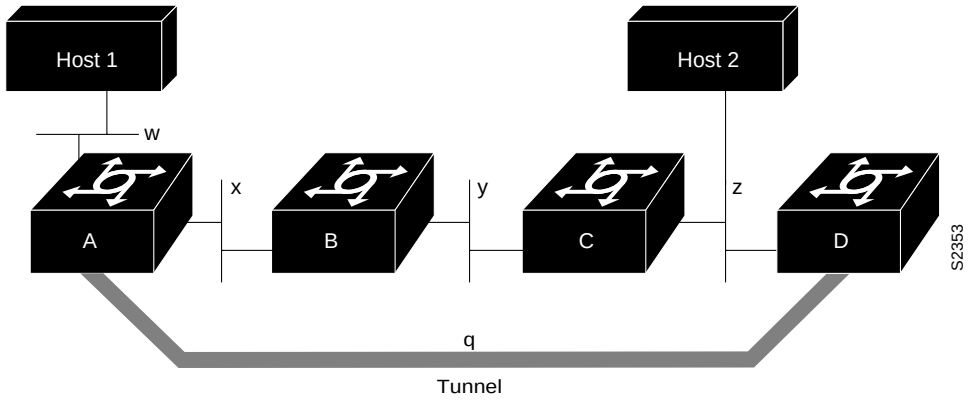
If the path between two computers has more than 15 hops, they cannot talk to each other, but it is possible to hide some of the hops inside the network with a tunnel.

Special Considerations

The following are considerations and precautions to observe when configuring tunneling:

- Encapsulation and decapsulation at the tunnel endpoints are slow operations; currently only processor switching is supported.
- Be cautious in your configuration and take into account security and topology issues. Be careful not to violate access control lists. You can configure a tunnel with a source and destination that is not restricted by firewall access communication servers.
- Tunneling may create problems with transport protocols with limited timers (for example, DECnet) due to increased latency.
- Multiple point-to-point tunnels can saturate the physical link with routing information.
- Routing protocols that make their decisions based solely on hop count will often prefer a tunnel over a multipoint real link. A tunnel might appear to be a one-hop, point-to-point link and have the lowest-cost path, but may actually cost more. For example, in the topology shown in Figure 1-5, packets from Host 1 will travel across networks w, q, and z to get to Host 2 instead of taking the path w, x, y, z because it “appears” shorter.

Figure 1-5 Tunnel Precautions: Hop Counts



- An even worse problem will occur if routing information from the tunneled network mixes with the transport networks' information. In this case, the best path to the "tunnel destination" is via the tunnel itself. This is called a recursive route and will cause the tunnel interface to temporarily shut down. To avoid recursive routing problems, keep passenger and transport network routing information disjointed:
  - Use a different AS number or tag
  - Use a different routing protocol
  - Use static routes to override the first hop (but watch for routing loops)
- If you see line protocol down, as in the following example, it may be because of a recursive route:

```
%TUN-RECURDOWN Interface Tunnel 0
temporarily disabled due to recursive routing
```

Configure IP Tunneling

If you want to configure IP tunneling, you must perform at least the first three tasks in the list that follows. The remaining tunnel configuration tasks are optional.

- Configure the tunnel interface (required)
- Configure the tunnel source (required)
- Configure the tunnel destination (required)
- Configure the tunnel mode
- Configure end-to-end checksums
- Configure a tunnel identification key
- Configure a tunnel interface to drop out-of-order datagrams
- Monitor IP tunnels

For examples using these commands, see the tunnel examples at the end of this chapter.

Configure the Tunnel Interface

In order to configure tunneling, you must configure the tunnel interface by performing the following task in global configuration mode:

Task	Command
Configure the tunnel interface.	**interface tunnel** *interface-number*

Configure the Tunnel Source

In order to configure tunneling, you must specify the tunnel interface's source address by performing the following task in interface configuration mode:

Task	Command
Configure the tunnel source.	**tunnel source** {*ip-address* | *interface-type interface-number*}

Note You cannot have two tunnels using the same encapsulation mode with exactly the same source and destination address. The workaround is to create a loopback interface and source packets off of the loopback interface.

Configure the Tunnel Destination

In order to configure tunneling, you must specify the tunnel interface's destination by performing the following task in interface configuration mode:

Task	Command
Configure the tunnel destination.	**tunnel destination** {*hostname* | *ip-address*}

Configure the Tunnel Mode

The encapsulation mode for the tunnel interface defaults to generic route encapsulation (**gre**), so this task is considered optional. However, if you want a mode other than **gre**, you must configure it by performing the following task in interface configuration mode:

Task	Command
Configure the tunnel mode.	**tunnel mode** {**cayman** | **eon** | **gre** | **nos**}

Configure End-to-End Checksums

Some passenger protocols rely on media checksums to provide data integrity. By default, the tunnel does not guarantee packet integrity. By enabling end-to-end checksums, the communication servers will drop corrupted packets. To enable such checksums on a tunnel interface, perform the following task in interface configuration mode:

Task	Command
Configure end-to-end checksumming.	**tunnel checksum**

Configure a Tunnel Identification Key

You can optionally enable an ID key for a tunnel interface. This key must be set to the same value on the tunnel endpoints. Tunnel ID keys can be used as a form of *weak* security to prevent misconfiguration or injection of packets from a foreign source.

The tunnel ID key is available with generic communication server encapsulation (GRE) only.

Note When using GRE, the ID key is carried in each packet. We do *not* recommend relying on this key for security purposes.

To configure a tunnel ID key, perform the following task in interface configuration mode:

Task	Command
Configure a tunnel identification key.	**tunnel key** *key-number*

Configure a Tunnel Interface to Drop Out-of-Order Datagrams

You can optionally configure a tunnel interface to drop datagrams that arrive out of order. This is useful when carrying passenger protocols that behave poorly when they receive packets out of order (for example, LLC2-based protocols). This option is available with generic communication server encapsulation (GRE) only.

To do so, perform the following task in interface configuration mode:

Task	Command
Configure a tunnel interface to drop out-of-order datagrams.	**tunnel sequence-datagrams**

Monitor IP Tunnels

Complete any of the following tasks in EXEC mode to monitor the IP tunnels you have configured:

Task	Command
List tunnel interface information.	**show interface tunnel** *unit* [**accounting**]
List the routes that go through the tunnel.	**show** *protocol* **route**
List the route to the tunnel destination.	**show ip route**

Reenable HDLC Serial Encapsulation

The communication server provides High-level Data Link Control (HDLC) encapsulation for serial lines by default. This encapsulation method provides the synchronous framing and error detection functions of HDLC without windowing or retransmission. Although it is the default, it can be reenabled as the encapsulation method, if necessary, by performing the following task in interface configuration mode:

Task	Command
Reenable HDLC encapsulation.	**encapsulation hdlc**

Select the Ethernet Encapsulation

Ethernet interfaces on the communication server support several encapsulation methods, depending upon the application type code and media type, as follows:

- Standard ARPA Ethernet Version 2.0 encapsulation (default)
- SAP IEEE 802.3 encapsulation
- The SNAP method, as specified in RFC 1042

Establish Ethernet encapsulation by selecting one of the Ethernet encapsulation methods, using the appropriate command in interface configuration mode, as follows:

Task	Command
Select ARPA Ethernet encapsulation.	**encapsulation arpa**
Select SAP Ethernet encapsulation.	**encapsulation sap**
Select SNAP Ethernet encapsulation.	**encapsulation snap**

Enable MOP

You can enable the Maintenance Operation Protocol (MOP) on an interface by performing the following task in interface configuration mode:

Task	Command
Enable MOP.	**mop enabled**

Enable MOP Message Support

You can enable an interface to send out periodic MOP system identification messages on an interface by performing the following task in interface configuration mode:

Task	Command
Enable MOP message support.	**mop sysid**

Select the Token Ring Speed

The Token Ring interface on the CSC-1R and CSC-2R can run at either 4 or 16 Mbps. These Token Ring interfaces do not default to any particular ring speed; you must select the speed the first time you use them.

Caution Configuring a ring speed that is wrong or incompatible with the connected Token Ring causes the ring to beacon, which effectively takes the ring down and makes it nonoperational.

Configure the ring speed on the CSC-1R or CSC-2R Token Ring interfaces by performing the following task in interface configuration mode:

Task	Command
Select the ring speed.	**ring-speed** *speed*

This command is not necessary for the CSC-R16 token ring interface card.

Enable Early Token Release

Our Token Ring interfaces support early token release, a method whereby the interface releases the token back onto the ring immediately after transmitting rather than waiting for the frame to return. This feature can help to increase the total bandwidth of the Token Ring. To configure the interface for early token release, perform the following task in interface configuration mode:

Task	Command
Enable early token release.	**early-token-release**

Configure Token Ring Interfaces in Source-Route Bridging Environments

Communication servers on a Token Ring network in a source-route bridging environment must support the collection and use of routing information field (RIF) information, to provide necessary path information to the host. The following tasks are necessary to enable the RIF:

- Enable the RIF and specify the protocol being used.
- Set the RIF timeout period.
- Set a static RIF table when the token ring host does not support use of IEEE 802.2 TEST or XID datagrams.

The following sections describe these tasks. For further information about the Token Ring interface and routing information fields, see the *Internetworking Technology Overview.*

Enable RIFs

Communication servers on a Token Ring network in a source-route bridging environment must support the collection and use of RIF information. To enable support for RIF information, perform the following task:

Task	Command
Enable collection and use of RIF information.	**multiring ip**

Level 3 routers that use protocol-specific information rather than MAC information to route datagrams must be able to collect and use RIF information to ensure that the Level 3 routers can transmit datagrams across a source-route bridge. The software default is to not collect and use RIF information for routed protocols. This allows operation with software that does not understand or properly use RIF information.

When it is enabled, the router will source packets that include information used by source-route bridges. This allows a communication server with the Token Ring interface to connect to a source-bridged Token Ring network.

Set the RIF Time-out Interval

RIF information is maintained in a cache whose entries are aged. You can set the number of minutes an inactive RIF entry is kept.

Task	Command
Set the RIF timeout interval.	**rif timeout** *minutes*

Configure a Static RIF Entry

If a Token Ring host does not support the use of IEEE 802.2 TEST or XID datagrams as explorer packets, you may need to add static information to the RIF cache of the router/bridge.

Task	Command
Enter static source-route information into the RIF cache.	**rif** *MAC-address* [*RIF-string*] [*interface-name*]

Configure the Point-to-Point Protocol

The Point-to-Point Protocol (PPP), described in RFCs 1331 and 1332, is a method of encapsulating Network Layer protocol information over point-to-point links. The document "Point-to-Point Initial Configuration Options" defines the set of options that are negotiated during startup.

The current implementation of PPP supports option 4, Link Quality Monitoring and option 5, Magic Number configuration options. The software always sends option 5 and will negotiate for option 4 if so configured. All other options are rejected.

IP and IPX upper-layer protocols are supported.

The software provides PPP as an encapsulation method. It also provides the Challenge Handshake Authentication Protocol (CHAP) on serial interfaces running PPP encapsulation. The following sections describe the tasks to configure these features.

Magic Number support is available on all serial interfaces. When using PPP, PPP will always attempt to negotiate for Magic Numbers, which are used to detect looped-back nets. The link might or might not be taken down upon looped-back detection, depending on the **down-when-looped** command.

For more information about PPP and CHAP, see the chapter "Configuring SLIP and PPP."

The following PPP configuration tasks are documented inthis seciton:

- Enable PPP encapsulation
- Enable CHAP
- Enable Link Quality Monitoring (LOM)

Enable PPP Encapsulation

You can enable the Point-to-Point Protocol on serial lines to encapsulate IP and serial IP (SLIP) datagrams. To do so, perform the following task in interface configuration mode:

Task	Command
Enable the PPP encapsulation.	**encapsulation ppp**

PPP echo requests are used as keepalives, to minimize disruptions to the end users of your network. The **no keepalive command** can be used to disable echo requests.

Enable Challenge Handshake Authentication Protocol (CHAP)

Access control using Challenge Handshake Authentication Protocol (CHAP) is available on all serial interfaces. The authentication feature will reduce the risk of security violations on your communication server.

Note To use CHAP, you must be running PPP encapsulation.

When CHAP is enabled, a remote device (a PC, workstation, or server) attempting to connect to the local communication server is requested, or *challenged*, to respond. The required response is an encrypted version of a secret password, or *secret*, plus a random value and the name of the remote device.

By transmitting this response, the secret is never transmitted, preventing other devices from stealing it and gaining illegal access to the system. Without the proper response, the remote device cannot connect to the local communication server.

CHAP transactions occur only at the time a link is established. The local communication server does not request a password during the rest of the call. (The local communication server can, however, respond to such requests from other devices during a call.)

To use CHAP, you must perform the following tasks:

- Enable CHAP on the interface. Once you have enabled CHAP, the local communication server requires a password from remote devices. If the remote device does not support CHAP, no traffic will be passed to that device.
- Configure server host name authentication. Configure the secret or password for each remote system with which authentication is required.

Perform the follwign tasks:

Task		Command
Step 1	Enable CHAP.	**ppp authentication chap**
Step 2	Configure host authentication.	**username** *name* **password** *secret*

For an example of CHAP, see the section " CHAP with an Encrypted Password Example" at the end of this chapter. CHAP is specified in the IETF RFC 1334 "The PPP Authentication Protocols" by Brian Lloyd of Lloyd and Associates and William A. Simpson of Computer Systems Consulting Services. CHAP is specified as an additional authentication phase of the PPP Link Control Protocol.

Enable Link Quality Monitoring (LQM)

Link Quality Monitoring (LQM) is available on all serial interfaces running PPP. LQM monitors the link quality, and if the quality drops below a configured percentage, the link is taken down. The percentages are calculated for both the incoming and outgoing directions. The outgoing quality is

calculated by comparing the total number of packets and bytes sent with the total number of packets and bytes received by the peer. The incoming quality is calculated by comparing the total number of packets and bytes received with the total number of packets and bytes sent by the peer.

When LQM is enabled, Link Quality Reports (LQRs) are sent every keepalive period. LQRs are sent in place of keepalives. All incoming keepalives are responded to properly. If LQM is not configured, keepalives are sent every keepalive period and all incoming LQRs are responded to with an LQR.

LQR is specified in the IETF RFC-1333, "PPP Link Quality Monitoring" by William A. Simpson of Computer Systems Consulting Services. The latest version is dated May 1992.

To enable LQM on the interface, perform the following task in interface configuration mode:

Task	Command
Enable LQM on the interface.	**ppp quality** *percentage*

The *percentage* argument specifies the link quality threshold. That percentage must be maintained, or the link is deemed to be of poor quality and taken down.

Configure Dial Backup Service

The dial backup service provides protection against WAN downtime by allowing you to configure a backup serial line via a circuit-switched connection.

To configure dial backup, associate a secondary serial interface as a backup to a primary serial interface. This feature requires that an external mode or CSU/DSU device attached to a circuit-switched service be connected on the secondary serial interface. The external device must be capable of responding to a DTR signal (DTR active) by automatically dialing a connection to a preconfigured remote site.

The dial backup software keeps the secondary line inactive (DTR inactive) until one of the following conditions is met:

- The primary line goes down.
- The transmitted traffic load on the primary line exceeds a defined limit.

These conditions are defined using the interface configuration commands described later in this section.

When the software detects a lost Carrier Detect signal from the primary line device or finds that the line protocol is down, it activates DTR on the secondary line. At that time, the modem or CSU/DSU device must be set to dial the remote site. When that connection is made, the routing protocol defined for the serial line will continue the job of transmitting traffic over the dialup line.

You also can configure the dial backup feature to activate the secondary line based upon traffic load on the primary line.

The software monitors the traffic load and computes a five-minute moving average. If this average exceeds the value you set for the line, the secondary line is activated, and depending upon how the line is configured, some or all of the traffic will flow onto the secondary dialup line.

Perform the following tasks in interface configuration mode:

Task		Command
Step 1	Select a serial interface as a backup line.	**backup interface** *interface-name*
Step 2	Enter the load as a percentage of the primary line's available bandwidth.	**backup load** {*enable-threshold* | **never**} {*disable-load* | **never**}
Step 3	Set the line delay for the backup line.	**backup delay** {*enable-delay* | **never**} {*disable-delay* | **never**}

See the examples of these commands at the end of this chapter.

For more information about dial backup, see the dial-on-demand routing (DDR) chapter of this manual.

Configure Loopback Detection

When an interface has a backup interface configured, it is often desirable that the backup interface be enabled when the primary interface is either down or in loopback. By default, the backup is only enabled if the primary interface is down. By using the **down-when-looped** command, the backup interface will also be enabled if the primary interface is in loopback. To achieve this condition, perform the following task in interface configuration mode:

Task	Command
Configure an interface to tell the system it is down when loopback is detected.	**down-when-looped**

If testing an interface with the loopback command, **down-when-looped** should not be configured, or packets will not be transmitted out the interface that is being tested.

Set Transmit Delay

It is possible to send back-to-back data packets over serial interfaces faster than some hosts can receive them. You can specify a minimum dead time after transmitting a packet to alleviate this condition. This setting is available for serial interfaces on the MCI and SCI interface cards. Perform the following task, as appropriate for your system, in interface configuration mode:

Task	Command
Set the transmit delay on the MCI and SCI synchronous serial interfaces.	**transmitter-delay** *microseconds*

Configure DTR Signal Pulsing

You can configure pulsing DTR signals on all serial interfaces. When the serial line protocol goes down (for example, because of loss of synchronization) the interface hardware is reset and the DTR signal is held inactive for at least the specified interval. This function is useful for handling encrypting or other similar devices that use the toggling of the DTR signal to resynchronize. To configure DTR signal pulsing, perform the following task in interface configuration mode:

Task	Command
Configure DTR signal pulsing.	**pulse-time** *seconds*

Configure the Clock Rate on DCE Appliques

You can configure the clock rate for appliques (connector hardware) on the serial interface of the MCI and SCI cards to an acceptable bit rate. To do so, perform the following task in interface configuration mode:

Task	Command
Configure the clock rate on serial interfaces.	**clockrate** *bps*

Control Interface Hold-Queue Limits

Each interface 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 one or more packets from the hold queue, the interface can accept new packets again. You can specify the hold-queue limit of an interface in interface configuration mode as follows:

Task	Command
Specify the maximum number of packets allowed in the hold queue.	**hold-queue** *length* {**in** | **out**}

Set Bandwidth

Higher-level protocols use bandwidth information to make operating decisions. For example, IGRP uses the minimum path bandwidth to determine a routing metric. The TCP protocol adjusts initial retransmission parameters based on the apparent bandwidth of the outgoing interface. Perform the following task in interface configuration mode to set a bandwidth value for an interface:

Task	Command
Set a bandwidth value.	**bandwidth** *kilobits*

The bandwidth setting is a routing parameter only; it does not affect the physical interface.

Set Interface Delay

Higher-level protocols might 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, perform the following task in interface configuration mode:

Task	Command
Set a delay value for interfaces.	**delay** *tens-of-microseconds*

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

Limit Size of the Transmit Queue

You can control the size of the transmit queue available to a specified interface on the MCI and SCI cards. To limit the size, perform the following task in interface configuration mode:

Task	Command
Limit the size of the transmit queue.	**tx-queue-limit** *number*

Adjust Maximum Packet Size or MTU Size

Each interface has a default maximum packet size or maximum transmission unit (MTU) size. This number generally defaults to 1500 bytes. On serial interfaces, the MTU size varies, but cannot be set smaller than 64 bytes. To adjust the maximum packet size, perform the following task in interface configuration mode:

Task	Command
Adjust the maximum packet size or MTU size.	**mtu** *bytes*

Monitor and Maintain the Interface

You can perform the following tasks to monitor and maintain the interfaces:

- Monitor interface status
- Clear and reset the interface
- Shutdown and restart the interface
- Enable interface loopback testing

The following sections describe the administrative tasks you perform to keep the interfaces up and running.

Monitor Interface Status

The software contains commands that you can enter at the EXEC prompt to display information about the interface including the version of the software and the hardware, the controller status, and statistics about the interfaces. The following table lists some of the interface monitoring tasks. (The full list of **show** commands can be displayed by entering the **show ?** command at the EXEC prompt.) These commands are fully described in the *Communication Server Command Reference* publication.

Perform the following commands in EXEC mode:

Task	Command
Display current internal status information for the interface controller cards.	**show controllers** {**mci** | **serial** | **token**}
Display the number of packets of each protocol type that have been sent through the interface.	**show interfaces** [*type unit*] [**accounting**]
Display the number of packets of each protocol type that have been sent through the asynchronous serial line.	**show interfaces async** [*unit*] [**accounting**]
Display the current contents of the Token Ring routing information field (RIF) cache.	**show rif**
Display the hardware configuration, software version, the names and sources of configuration files, and the boot images.	**show version**

Clear and Reset the Interface

To clear the interface counters shown with the **show interfaces** command, enter the following command at the EXEC prompt:

Task	Command
Clear the interface counters.	**clear counters** [*type number*]

Note This command will not clear counters retrieved using SNMP, but only those seen with the EXEC **show interfaces** command.

Complete the following tasks in EXEC mode to clear and reset interfaces. Under normal circumstances, you do not need to clear the hardware logic on interfaces.

Task	Command
Reset the hardware logic on an interface.	**clear interface** *type number*
Reset the hardware logic on an asynchronous serial line.	**clear line** [*number*]
Clear the entire Token Ring RIF cache.	**clear rif-cache**

Shut Down and Restart an Interface

You can disable an interface. Doing so disables all functions on the specified interface and marks the interface as unavailable on all monitoring command displays. This information is communicated to other network servers through all dynamic routing protocols. The interface will not be mentioned in any routing updates. On serial interfaces, this command causes the DTR signal to be dropped. On Token Ring interfaces, this command causes the interface to deinsert from the ring.

To shut down an interface and then restart the disabled interface, perform the following tasks in interface configuration mode:

Command	Task
Shut down an interface.	**shutdown**
Reenable an interface.	**no shutdown**

To check whether an interface is disabled, use the EXEC command **show interfaces**. An interface that has been shut down is shown as administratively down in the **show interfaces** command display. See examples in the section "Interface Shutdown Examples" at the end of this chapter.

Run Interface Loopback Diagnostics

You can use a loopback test on lines to detect and distinguish equipment malfunctions between line and modem or CSU/DSU (Channel Service Unit/Digital Service Unit) problems on the network server. If correct data transmission is not possible when an interface is in loopback mode, the interface is the source of the problem. The DSU might have similar loopback functions you can use to isolate the problem if the interface loopback test passes. If the device does not support local loopback, this function will have no effect.

You can specify hardware loopback tests on the Ethernet and synchronous serial interfaces, and all Token Ring interfaces (except the CSC-R 4-megabit card) that are attached to CSU/DSUs, and that support the local loopback signal. The CSU/DSU acts as a Data Communications Equipment (DCE) device; the communication server acts as a Data Terminal Equipment (DTE) device. The local loopback test generates a CSU loop—a signal that goes through the CSU/DSU to the line, then back through the CSU/DSU to the communication server. The **ping** command can also be useful during loopback operation.

The loopback tests are available on the following interfaces:

- Cisco Multiport Communications Interface (MCI) and Cisco Serial Communication Interface (SCI) synchronous serial interfaces
- MCI and Cisco Multiport Ethernet Controller (MEC) Ethernet interfaces; an Ethernet loopback server is also provided on the Ethernet interfaces
- Token Ring interfaces

The following sections describe each test.

Note Loopback does not work on an X.21 DTE because the X.21 interface definition does not include a loopback definition. The loopback tests do not work on the Ethernet interface of the IGS communication server product.

Enable Loopback on MCI and SCI Serial Cards

The MCI and SCI serial interface cards support the loopback function when a CSU/DSU or equivalent device is attached to the communication server. Perform the folowing task in interface configuration mode:

Task	Command
Enable loopback through a CSU/DSU to configure a CSU loop on the MCI and SCI synchronous serial interfaces.	**loopback**

Enable Loopback on MCI and MEC Ethernet Cards

The Ethernet interfaces on the MCI and MEC cards support loopback mode. To enable loopback mode on them, perform the following task in interface configuration mode:

Task	Command
Enable loopback to verify that the interface receives back every packet it sends.	**loopback**

Configure the Ethernet Loopback Server

The communication server software provides an Ethernet loopback server that supports Digital, Intel, and Xerox systems specified by the "blue book," a joint specification written by Digital Equipment Corporation (Digital), Intel, and Xerox that defines the Ethernet protocol. The loopback server responds to forward data loopback messages sent either to the server's MAC address or to the broadcast address. Currently, the Ethernet loopback server does not respond to the loopback assistance multicast address.

Use the Ethernet loopback server to test communications between your internetworking products and Digital systems that do not support the IP **ping** command, such as DECnet-only VMS systems.

To originate a loop test on your VMS system with a Cisco server, use the Digital Network Control Program (NCP) command **Loop Circuit**. For more information about the **Loop Circuit** command, consult the DECnet VAX documentation. Cisco network servers support all options that can be specified by the VMS hosts.

Enable Loopback on Token Ring Cards

You can place all of the Token Ring interface cards except the 4-MB CSC-R card into loopback mode by performing the following task in interface configuration mode:

Task	Command
Enable loopback to verify that the Token Ring interface receives back every packet it sends.	**loopback**

Interface Configuration Examples

Use the configuration examples in this section to understand some aspects of interface configuration.

Example of Enabling Interface Configuration

The following example illustrates how to begin interface configuration. It assigns Point-to-Point (PPP) encapsulation to interface serial 0.

```
interface serial 0
encapsulation ppp
```

Example of Restricting Access on the Asynchronous Interface

The following example assumes that users are restricted to certain servers designated as asynchronous servers, but that normal terminal users can access anything on the local network:

```
! access list for normal connections
access-list 1 permit 131.108.0.0 0.0.255.255
!
access-list 2 permit 131.108.42.55
access-list 2 permit 131.108.111.1
access-list 2 permit 131.108.55.99
!
interface async 1
async dynamic address
ip access-group 1 out
ip access-group 2 in
```

Interface Description Examples

The following example illustrates how to add a description about an interface that will appear in configuration files and monitoring command displays:

```
interface ethernet 0
description First Ethernet in network 1
ip address 101.13.15.78 255.255.255.0
```

Interface Shutdown Examples

The following example turns off the Ethernet interface in slot 2 at port 4:

```
interface ethernet 2/4
shutdown
```

The following example turns the interface back on:

```
interface ethernet 2/4
no shutdown
```

The following example illustrates how to shut down a Token Ring interface:

```
interface tokenring 0
shutdown
```

IP Tunneling Examples

The following example shows a configuration that will route a private IP network and a Novell network across a Public Service Provider.

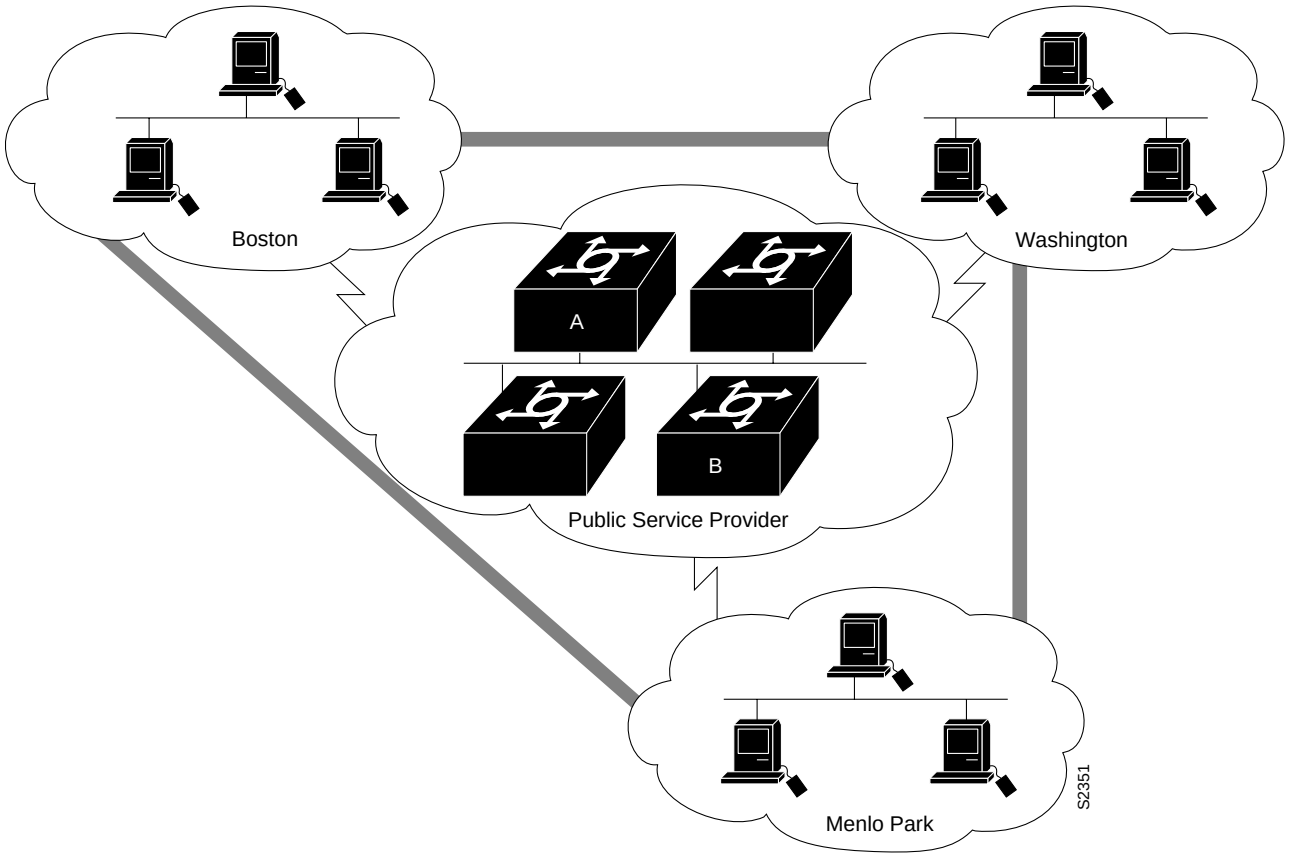
In Figure 1-6, Communication Server A is configured as follows:

```
interface ethernet 0
description boston office
ip address 10.1.1.1 255.255.255.0
novell network 1e

interface serial 0
description connection to NEARnet
ip address 192.13.2.1 255.255.255.0

interface tunnel 0
tunnel source serial 0
tunnel destination 131.108.5.2
ip address 10.1.2.1 255.255.255.0
novell network 1f
```

Figure 1-6 Creating Virtual Private Networks across WANs



The following commands illustrate Router B's configuration:

```
interface ethernet 0
description menlo park office
ip address 10.1.3.1 255.255.255.0
novell network 31

interface serial 4
description connection to BARRnet
ip address 131.108.5.2 255.255.255.0

interface tunnel 0
tunnel source serial 4
tunnel destination 192.13.2.1
ip address 10.1.2.2 255.255.255.0
novell network 1f
```

CHAP with an Encrypted Password Example

The following configuration examples enable CHAP on interface serial 0 of three communication servers.

Communication Server yyy

```
hostname yyy
interface serial 0
encapsulation ppp
ppp authentication chap
username xxx password secretxy
username zzz password secretzy
```

Communication Server xxx

```
hostname xxx
interface serial 0
encapsulation ppp
ppp authentication chap
username yyy password secretxy
username zzz password secretxz
```

Communication Server zzz

```
hostname zzz
interface serial 0
encapsulation ppp
ppp authentication chap
username xxx password secretxz
username yyy password secretzy
```

When you look at the configuration file, the passwords will be encrypted and the display will look similar to the following:

```
hostname xxx
interface serial 0
encapsulation ppp
ppp authentication chap
username yyy password 7 121F0A18
username zzz password 7 1329A055
```

Example of Dial Backup Service When the Primary Line Goes Down

The following example configures serial 1 as a secondary line that activates only when the primary line (serial 0) goes down. The secondary line will not be activated due to load of the primary.

```
interface serial 0
backup interface serial 1
backup delay 30 60
```

The secondary line is configured to activate 30 seconds after the primary line goes down and to remain on for 60 seconds after the primary line is reactivated.

Examples of Dial Backup Service When the Primary Line Reaches Threshold

The following example configures the secondary line (serial 1) to be activated only when the load of the primary line reaches a certain threshold:

```
interface serial 0
backup interface serial 1
backup load 75 5
```

In this case, the secondary line will not be activated when the primary goes down. The secondary line will be activated when the load on the primary line is greater than 75 percent of the primary's bandwidth. The secondary line will then be brought down when the aggregate load between the primary and secondary lines fits within 5 percent of the primary bandwidth.

Examples of Dial Backup Service When the Primary Line Exceeds Threshold

This example configures the secondary line to activate once the traffic threshold on the primary line exceeds 25 percent:

```
interface serial 0
backup interface serial 1
backup load 25 5
backup delay 10 60
```

Once the aggregate load of the primary and the secondary lines return to within 5 percent of the primary bandwidth, the secondary line is deactivated. The secondary line waits 10 seconds after the primary goes down before activating, and remains active for 60 seconds after the primary returns and becomes active again.