CHAPTER 6

Configuring Interfaces

Use the information in this chapter to understand the types of interfaces supported on our routers. Our routers 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 routers support include subinterfaces and IP tunnels.

For hardware technical descriptions and information about installing the router interfaces, refer to the hardware installation and maintenance publication for your product. For command descriptions and usage information, refer to the chapter entitled "Interface Commands" of the *Router Products Command Reference* publication. For a conversion table of the modular products and Cisco 7000 processors, refer to the appendix entitled "Cisco 7000 Processors" in the *Router Products Command Reference* publication.

Interface Configuration Task List

You can perform the tasks in the following sections to configure and maintain the interfaces supported on our routers. The first two sections introduce material that you might need to know in advance of the other tasks.

- Understand Supported Interfaces and Encapsulations
- Understand Fast, Autonomous, and SSE Switching Support
- Configure the Interface Type
- Add a Description for an Interface
- Configure Subinterfaces
- Understand Tunneling
- Configure IP Tunneling
- Configure Synchronous Serial Features
- Select Ethernet Encapsulation
- Configure the Ethernet Network Interface Module on the Cisco 4000
- Extend the 10BaseT Capability on the Cisco 4000
- Configure a Hub
- Configure ATM-DXI
- Convert HSSI to Clock Master
- Configure MOP

- Configure Token Ring Features
- Configure ISDN Features
- Configure Channelized T1
- Configure FDDI Timers and Features
- Configure PPP
- Configure Dial Backup Service
- Configure Loopback Detection
- Control Interface Hold-Queue Limits
- Set Bandwidth
- Set Interface Delay
- Adjust Timers
- Limit Transmit Queue Size
- Adjust Maximum Packet Size or MTU Size
- Monitor and Maintain the Interface

See the end of this chapter for configuration examples.

Understand Supported Interfaces and Encapsulations

The following sections describe the interfaces and encapsulations that our routers support:

- Synchronous Serial
- Asynchronous Serial
- Ethernet
- Token Ring
- ISDN Basic Rate Interface (BRI)
- Fiber Distributed Data Interface (FDDI)
- High-Speed Serial Interface (HSSI)
- Channelized T1

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 high-speed synchronous serial interface on the Cisco 2500 series and Cisco 3000 network servers

- On the Cisco 7000 series, the fast serial interface processor (FSIP) for four or eight channel-independent, synchronous serial ports that support full-duplex operation at DS-1 (1.544 Mbps) and E-1 (2.048 Mbps) speeds. Each port supports any of the available interface types (RS-232, RS-449, V.35, X.21, and RS-530), and each can be configured individually to operate with either internal or external timing signals.

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 the router supports.

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)
- Synchronous Data Link Control (SDLC)
- Switched Multimegabit Data Services (SMDS)
- Cisco Serial Tunnel (STUN)
- X.25-based encapsulations

Encapsulation methods are set according to the type of protocol or application you configure on your router. ATM-DXI is described later in this chapter in the section “Configure ATM-DXI.” 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 PPP.” The remaining methods are defined in their respective chapters describing the protocols or applications. Serial encapsulation methods are also discussed in the *Router Products Command Reference* publication in the chapter entitled “Interface Commands” under the **encapsulation** command.

Synchronous Serial Compression

The synchronous serial interface supports point-to-point compression. Our software implements a predictor compressor (the RAND algorithm). Compression of LAPB data is supported for LAPB.

Asynchronous Serial

All of our router platforms configured with an auxiliary port support the asynchronous serial interface.

Asynchronous Serial Encapsulation Methods

There are two asynchronous serial encapsulation methods:

- SLIP
- Asynchronous PPP

See the section "Establish Asynchronous Connections Using PPP or SLIP" later in this 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 in the modular routers, 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 in the modular routers, which provides two, four, or six high-speed Ethernet connectors compatible with Ethernet Versions 1 and 2 and the IEEE 802.3 protocol.
- An integrated Ethernet controller on the Cisco 2500 series and Cisco 3000 models.
- On the Cisco 7000 series, the high-speed Ethernet interface processor (EIP) for two, four, or six AUI ports. The EIP ports are in compliance with Ethernet versions 1 and 2 and the IEEE 802.3 specifications.

Use the **show interfaces**, **show controllers mci**, and **show controllers cbus** EXEC commands to display the Ethernet port numbers. These commands provide a report for each interface supported by the router. Use the **show interfaces pcbus** and **show controllers pcbus** commands to display information about the ISA bus interface. The **show interfaces pcbus** and **show controllers pcbus** commands are valid on LanOptics' Branchcard or Stacknet 2000 products only.

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 router and the routing or bridging application you configure. Further detail is provided in the sections "Select Ethernet Encapsulation" and "Example of Enabling Ethernet Encapsulation" later in this chapter. See also the chapters describing specific 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 are the CSC-C2CTR,CSC-R16 (or CSC-R16M), CSC-1R, and CSC-2R (dual Token Ring card).
- On the Cisco 7000 series, the high-speed Token Ring Interface Processor (TRIP) that has two or four DB-9 ports and interconnects network servers to IEEE 802.5 and IBM-compatible Token Ring media.

  The Token Ring interface supports both routing (Layer 3 switching) and source-route bridging (Layer 2 switching). The use of routing and bridging is on a per-protocol basis. For example, IP traffic could be routed while SNA traffic is bridged. The routing support interacts correctly with source-route bridges.

  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**, **show controllers token**, and **show controllers cbus** EXEC commands to display the Token Ring numbers. These commands provide a report for each ring supported by the router. Use the **show interfaces pcbus** and **show controllers pcbus** commands to display information about the ISA bus interface. The **show interfaces pcbus** and **show controllers pcbus** commands are valid on LanOptics' Branchcard or Stacknet 2000 products only.

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.

ISDN Basic Rate Interface (BRI)

The Integrated Services Digital Network (ISDN) Basic Rate Interface (BRI) is currently supported only on the Cisco 2500 series and Cisco 3000 routers. To place calls on the ISDN interface, you must configure it with dial-on-demand routing (DDR). For configuration information about ISDN using dial-on-demand routing (DDR), see the chapter entitled "Configuring DDR" in this document. For command information, refer to the chapter entitled "DDR Commands" in the *Router Products Command Reference* publication.

The BRI interface includes one ISDN Basic Rate connection. The Basic Rate connection consists of a D channel and two B channels, both of which are full-duplex, 64-kbps channels.

The D channel is used for call setup and network connection teardown. Call setup involves the data link and network connection. D-channel communication is from the router to the ISDN switch.

The B channels transmit user data. The B channels are treated as serial lines and support HDLC and PPP encapsulation. The interface configuration is propagated to each of the B channels. Although each channel is treated as a separate line, you cannot configure the channels separately. A single switch type must be configured for the router as a whole, because B channels are in a permanent rotary group.

The ISDN data link layer interface provided by the router conforms to the specification defined by the CCITT recommendation Q.921. The ISDN network layer interface provided by the router conforms to the specification defined by the CCITT recommendation Q.931.

For a list of ISDN switch types that the ISDN interface supports, see the section "Configure an ISDN Basic Rate Interface" later in this chapter.

Use the **show controllers bri** EXEC command to display the B and D channel information.

ISDN Encapsulation Methods

Each of the B channels is treated as a serial line and supports HDLC and PPP encapsulation. The default serial encapsulation is HDLC.

Fiber Distributed Data Interface (FDDI)

The FIber Distributed Data Interface (FDDI) is an ANSI-defined standard for timed 100-Mbps token passing over fiber-optic cable. An FDDI network consists of two counter token-passing fiber-optic rings. On most networks, the primary ring is used for data communication and the secondary ring is used as a hot standby. The FDDI standard sets total fiber lengths of 2 kilometers for multimode fiber and 10 kilometers for single-mode fiber, both of which are supported by our FDDI interface controller. (The maximum circumference of the FDDI network is only half the specified kilometers because of the *wrapping* or looping back of the signal that occurs during fault isolation.)

The FDDI standard allows a maximum of 500 stations with a maximum distance between active stations of two kilometers. The FDDI frame can contain a minimum of 22 bytes and a maximum of 4500 bytes. Our implementation of FDDI complies with Version 6.1 of the X3T9.5 FDDI specification, offering a Class A dual-attach interface that supports the fault-recovery methods of the dual attachment stations (DASs).

Support for FDDI is supplied on one of our FDDI interface cards, as follows:

- The CSC-FCI interface card, which operates with the standard modular router controller complex
- The CSC-C2/FCIT interface card, which operates with the ciscoBus II controller complex
- On the Cisco 7000 series, the high-speed multimode-to-multimode, single mode-to-single mode, multimode-to-single mode, or single mode-to-multimode FDDI interface processor (FIP).

We also provide support for some of the FDDI MIB variables as described in RFC 1285, "FDDI Management Information Base," published in January 1992 by Jeffrey D. Case of the University of Tennessee and SNMP Research, Inc. One such variable that we support is *snmpFddiSMTCFState*.

FDDI Encapsulation Methods

Our FDDI by default uses the SNAP encapsulation format defined in RFC 1042. It is not necessary to define an encapsulation method for this interface when using the CSC-FCI interface card or FIP.

The CSC-C2/FCIT interface card and FIP fully support transparent and translational bridging for the following configurations:

- FDDI to FDDI
- FDDI to Ethernet
- FDDI to Token Ring

When using the CSC-C2/FCIT interface card or FIP, you can specify the encapsulation method. See the section "Enable FDDI Bridging Encapsulation" later in this chapter.

Using Connection Management (CMT) Information

Connection Management (CMT) is an FDDI process that handles the transition of the ring through its various states (off, on, active, connect, and so on) as defined by the X3T9.5 specification. The FIP provides CMT functions in microcode.

A partial sample output of the **show interfaces fddi** command follows, along with an explanation of how to interpret the CMT information in the output.

```
Phy-A state is active, neighbor is B, cmt signal bits 08/20C, status ALS
Phy-B state is active, neighbor is A, cmt signal bits 20C/08, status ILS
CFM is thru A, token rotation 5000 usec, ring operational 0:01:42
Upstream neighbor 0800.2008.C52E, downstream neighbor 0800.2008.C52E
```

The **show interfaces fddi** example shows that Physical A (Phy-A) completed CMT with its neighbor. The state is active and the display indicates a Physical B-type neighbor.

The sample output indicates cmt signal bits 08/20C for Phy-A. The transmit signal bits are 08. Looking at the PCM state machine, 08 indicates that the port type is A, the port compatibility is set, and the LCT duration requested is short. The receive signal bits are 20C, which indicate the neighbor type is B, port compatibility is set, there is a MAC on the port output, and so on.

The neighbor is determined from the received signal bits, as follows:

Bit Positions	9	8	7	6	5	4	3	2	1	0
Value Received	1	0	0	0	0	0	1	1	0	0

Interpreting the bits in the diagram above, the received value equals 0x20C. Bit positions 1 and 2 (0 1) indicate a Physical B-type connection.

The transition states displayed indicate that the CMT process is running and actively trying to establish a connection to the remote physical connection. The CMT process requires state transition with different signals being transmitted and received before moving on to the state ahead as indicated in the PCM state machine. The ten bits of CMT information are transmitted and received in the Signal State. The NEXT state is used to separate the signaling performed in the Signal State. Therefore, in the preceding sample output, the NEXT state was entered 11 times.

Note The display line showing transition states is not generated if the FDDI interface has been shut down, or if the **cmt disconnect** command has been issued, or if the **fddi if-cmt** command has been issued. (The **fddi if-cmt** command applies to the AGS+ and Cisco 7000 only.)

The CFM state is thru A in the sample output, which means this interface's Phy-A has successfully completed CMT with the Phy-B of the neighbor and Phy-B of this interface has successfully completed CMT with the Phy-A of the neighbor.

The display (or nondisplay) of the upstream and downstream neighbor does not affect the ability to route data. Since the upstream neighbor is also its downstream neighbor in the sample, there are only two stations in the ring; the network server and the router at address 0800.2008.C52E.

High-Speed Serial Interface (HSSI)

The High-Speed Serial Interface (HSSI) consists of the following components:

- The CSC-HSCI controller card, which is ciscoBus-resident.
- The CSC-HSA, which is a back-panel applique.

  The controller card provides a single, full-duplex, synchronous serial interface capable of transmitting and receiving data at up to 52 megabits per second (Mbps). The HSSI is an approved standard (ANSI/EIA RS-613) providing connectivity to T3 (DS-3), E3, SMDS (at a DS-3 route), and other high-speed wide-area services through a DSU or Line Termination Unit.

  — The high-speed, full-duplex, synchronous serial interface is supported only on our modular network server products.

  — This ciscoBus card can query the appliques to determine their types. However, it does so only at system startup, so the appliques must be attached when the system is started. Issue a **show controllers cbus** command to determine how the HSSI card has identified them. The command also will show the capabilities of the card and report controller-related failures.
- On the Cisco 7000 series, the HSSI interface processor (HIP), which provides a single HSSI network interface for the Cisco 7000. The network interface resides on a modular interface processor (IP) that provides a direct connection between the high-speed Cisco Extended Bus (CxBus) and an external network.

HSSI Encapsulation Methods

The HSSI supports the serial encapsulation methods, except for X.25-based encapsulations, listed in the section "Synchronous Serial Encapsulation Methods" earlier in this chapter.

Channelized T1

Support for channelized T1 (also referred to as *fractional* T1) is provided only on the Cisco 7000 series by means of a MultiChannel Interface Processor (MIP) and a CxBus channelized T1 adapter (CxCT1). Each CxCT1 can support a maximum of 24 DS0 channel groups. Each channel group is presented to the system as a serial interface that can be configured individually. The MIP can support one or two CxCT1 adapters, providing a maximum of 48 DS0 channel groups per MIP. In effect, up to 24 DS0 circuits are multiplexed to a single hardware port on the CxCT1 adapter.

Use the **show controllers t1** EXEC command to display current T1 status. This command provides a report for each physical interface configured to support channelized T1.

Channelized T1 supports the following WAN protocols:

- X.25
- LAPB

- Frame Relay
- Dial-on-Demand

Online Insertion and Removal (OIR)

The Cisco 7000's Online Insertion and Removal (OIR) feature allows you to remove and replace CxBus interface processors (IPs) while the system is on line. You can shut down the IP before removal and restart it after insertion without causing other software or interfaces to shut down.

Note Do not remove or install more that one interface processor at one time. After a removal or installation, observe the LEDs before continuing.

You do not need to notify the software that you are going to remove or install an interface processor. When the route processor is notified by the system that an interface processor has been removed or installed, it stops routing and scans the system for a configuration change. All interface processors are initialized, and each interface type is verified against the system configuration; then the system runs diagnostics on the new interface.

Note Only the Cisco 7000 series supports OIR.

Only an interface of a type that has been configured previously will be brought on line; others require configuration. If a newly installed interface processor does not match the system configuration, the interface is left in an administratively down state until the system operator configures the system with the new interfaces.

Hardware (MAC-level) addresses for all interfaces on the Cisco 7000 are stored on an electronically erasable programmable read-only memory (EEPROM) component in the route processor (RP) instead of on the individual interface boards. An address allocator in the EEPROM contains a sequential block of 40 addresses (5 interface slots times a maximum of 8 possible ports per slot). Each address is assigned to a specific slot and port address in the chassis, regardless of how the interfaces are configured. This allows interfaces to be replaced online without requiring the system to update routing tables and data structures. Regardless of the types of interfaces installed, the hardware addresses do not change unless you replace the system RP. If you do replace the RP, the hardware addresses of *all* ports change to those specified in the address allocator on the new RP.

Understand Fast, Autonomous, and SSE Switching Support

This section discusses fast switching, autonomous switching, and silicon switching engine (SSE) switching.

Fast Switching Support

Fast switching allows higher throughput by switching a packet using a cache created by previous packets. Fast switching is enabled by default on all interfaces that support fast switching. The router supports fast switching of the following protocols:

- AppleTalk
- DECnet
- IP
- IPX
- ISO CLNS
- Source-route bridging
- VINES
- XNS

See the appropriate protocol-specific chapters for more information about fast switching.

Autonomous Switching Support

This section discusses the router's autonomous switching support. Autonomous switching is a feature that provides faster packet switching by allowing the CiscoBus controller to switch packets independently without having to interrupt the system processor. It is available only in Cisco 7000 systems and in AGS+ systems with high-speed network controller cards. Autonomous switching is disabled by default on all interfaces.

The router supports autonomous switching of the following protocols:

- IP
- IPX (except on cbus I)
- Source-route bridging
- Transparent bridging (except on cbus I)

See the separate protocol-specific chapters for more information about autonomous switching.

Transparent Bridging

The router supports the following types of transparent bridging:

- Ethernet (MEC) to Ethernet (MEC)
- Ethernet (MEC) to FDDI (FCIT)
- FDDI (FCIT) to Ethernet (MEC)
- FDDI (FCIT) to FDDI (FCIT)

IP over PPP and IETF Frame Relay

The router supports autonomous switching of IP over PPP and IETF Frame Relay-encapsulated traffic.

IPX

You can autonomously switch to and from any of the encapsulations listed below:

- Ethernet ARPA (Novell Ethernet_II)
- Ethernet NOVELL-ETHER (Novell Ethernet_802.3)
- Ethernet SAP (Novell IEEE 802.2)
- Ethernet SNAP (Novell IEEE 802.2/SNAP)
- Token Ring SAP (Novell IEEE 802.2)
- Token Ring SNAP (Novell IEEE 802.2/SNAP)
- FDDI SAP (Novell IEEE 802.2)
- FDDI SNAP (Novell IEEE 802.2/SNAP)
- Serial HDLC

See the **ipx route-cache cbus** command in the chapter entitled "Novell IPX Commands" in the *Router Products Command Reference* publication for more information.

SSE Switching

The silicon switching engine (SSE) is on the Silicon Switch Processor (SSP) board in a Cisco 7000. SSE switching contributes to very fast packet processing by allowing the SSE to perform switching independently of the system processor. SSE switching give a router even faster packet processing by allowing the SSE to switch packets without interrupting the system processor. It works only in Cisco 7000 systems with the SSP board.

Configure the Interface Type

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

Step 1 Enter the **configure** EXEC command at the privileged EXEC prompt to enter global configuration mode.

Step 2 Once in the global configuration mode, start configuring the interface by entering the **interface** command. 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 the router supports, 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, in the preceding line you can specify either serial 0 or serial0.

Step 3 Follow each **interface** command with the interface configuration commands your particular interface requires. These command define the protocols and applications that will run on this interface. The commands are collected and applied to the **interface** command until you enter another **interface** command, a command that is not an interface configuration command, or you type the Ctrl-Z sequence to get out of configuration mode and return to privileged EXEC mode.

Step 4 Once an interface is configured, you can check its status by entering the EXEC **show** commands described after the task tables that follow.

Note When you configure channelized T1, you must first define the channels and the timeslots that comprise the channels by using the **controller t1** and the **channel-group** controller configuration commands. Then configure the virtual serial interfaces using the **interface serial** global configuration commands. See the section "Configure Channelized T1" later in this chapter for T1 configuration tasks.

The following sections show how to begin to configure each interface type as a separate task. Follow this command with the routing or bridging interface configuration commands for your particular protocol or application, as described in subsequent chapters.

See the section "Examples of Enabling Interface Configuration" at the end of this chapter.

Configure an ATM Interface

To configure an ATM interface on the Cisco 7000 series, perform the following task in global configuration mode:

Task	Command
Begin interface configuration.	**interface atm** *slot*/**0**

See the "Configuring ATM" chapter for more details on how to configure ATM.

Configure an ISDN Basic Rate Interface

To configure an ISDN Basic Rate Interface (BRI), perform the following task in global configuration mode:

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

You must also specify an ISDN switch type. See the sections "Select the ISDN BRI Switch Type" and "Define ISDN Service Profile Identifiers (SPIDs)" later in this chapter for additional information.

Configure a Dialer Interface

To begin to configure a dialer interface, perform the following task in global configuration mode:

Task	Command
Begin interface configuration.	**interface dialer** *interface-number*
For the Cisco 7000 series.	**interface dialer** *slot/port*

Configure an Ethernet Interface

To begin to configure an Ethernet interface, perform the following task in global configuration mode:

Task	Command
Begin interface configuration.	**interface ethernet** *interface-number*
For the Cisco 7000 series.	**interface ethernet** *slot/port*

Configure an FDDI

To begin to configure an FDDI interface, perform the following task in global configuration mode:

Task	Command
Begin interface configuration	**interface fddi** *interface-number*
For the Cisco 7000 series.	**interface fddi** *slot/port*

Configure an HSSI

To begin to configure an HSSI, perform the following task in global configuration mode:

Task	Command
Begin interface configuration.	**interface hssi** *interface-number*
For the Cisco 7000 series.	**interface hssi** *slot/port*

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 is always up and 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, for RSRB connections, or for establishing a Telnet session from the router's console to its auxiliary port when all other interfaces are down. In applications where other routers 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 router and processed locally. IP packets routed out the loopback interface but not destined to the loopback interface are dropped. This means that the loopback interface does double duty as the Null0 interface.

Note Loopback does not work on an X.21 DTE because the X.21 interface definition does not include a loopback definition.

To configure a loopback interface, perform the following task in global configuration mode:

Task	Command
Begin interface configuration.	**interface loopback** *interface-number*
For the Cisco 7000 series.	**interface loopback** *slot/port*

See also the section "Run Interface Loopback Diagnostics" later in this chapter.

Configure a Null Interface

The router 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 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 begin to configure a synchronous serial interface, perform the following task in global configuration mode:

Task	Command
Begin interface configuration.	**interface serial** *interface-number*
For the Cisco 7000 series.	**interface serial** *slot/port*
Begin interface configuration for a channelized T1 interface.	**interface serial** *slot/port***:***channel-group*

On the Cisco 7000 series, specify the serial interface slot number with the argument *slot.* The *slot* argument is the backplane slot number and can be in the range 0 through 4 on a Cisco 7000 or 0 through 2 on a Cisco 7010. The *port* argument is the port number of the fast serial interface processor (FSIP) and can be in the range 0 through 7.

For synchronous serial features, see the section "Configure Synchronous Serial Features" later in this chapter.

Configure a Token Ring Interface

To begin to configure a Token Ring interface, perform the following task in global configuration mode:

Task	Command
Begin interface configuration.	**interface tokenring** *interface-number*
For the Cisco 7000 series.	**interface tokenring** *slot/port*

Configure a Tunnel Interface

To configure a tunnel interface, perform the following task in global configuration mode. Before you configure a tunnel interface, see the sections in this chapter entitled "Understand Tunneling" and "Configure IP Tunneling."

Task	Command
Begin interface configuration.	**interface tunnel** *interface-number*
For the Cisco 7000 series.	**interface tunnel** *slot/port*

Configure an Asynchronous Serial Interface

Only the auxiliary port on a router can be configured as an asynchronous serial interface. To configure an asynchronous serial interface on the router, you must establish asynchronous serial line connections using PPP or SLIP as described in the next section.

Establish Asynchronous Connections Using PPP or SLIP

PPP and SLIP define methods of sending Internet packets over a standard RS-232 asynchronous serial line. PPP also defines methods for sending IPX packets. The router contains one such port, the "auxiliary" port. To use the asynchronous device as a network interface via PPP or SLIP, complete the following tasks:

- Configure asynchronous interface 1.
- Configure PPP or SLIP encapsulation on the asynchronous interface.
- Configure the addressing method.
- Configure dedicated or interactive mode.
- Enable asynchronous routing.
- Connect to remote routers via PPP or SLIP.

Note You can also configure support for SLIP and PPP using extended BootP requests. See the chapter entitled "Loading System Images, Microcode Images, and Configuration Files."

Configure Asynchronous Interface 1

The auxiliary port's absolute line number is 1. When you configure an asynchronous interface with the **interface async 1** command, you enable asynchronous routing over the auxiliary port to support PPP and SLIP connections to remote routers. The interface number is the same as the absolute line number.

The router automatically associates the interface number 1 with the absolute line number 1 of the auxiliary port, and treats the interface as an asynchronous line. However, to configure the auxiliary port as an asynchronous interface, you must also configure it as an auxiliary line with the **line aux 1** command as described in the chapter entitled "Configuring Terminal Lines and Modem Support." Follow the **line** command with the appropriate line configuration commands for modem control, such as speed. Perform the following task in global configuration mode to specify the auxiliary port line as an asynchronous interface:

Task	Command
Specify an asynchronous interface.	**interface async 1**

Only IP packets can be sent across lines configured for SLIP. PPP supports transmission of both IP and IPX packets.

Configure PPP or SLIP Encapsulation

SLIP and PPP are methods of encapsulating datagrams and other network-layer protocol information over point-to-point links. SLIP is the default method. Use the following command in interface configuration mode to configure PPP or SLIP encapsulation on the asynchronous interface:

Task	Command
Configure PPP or SLIP encapsulation on an asynchronous line.	**encapsulation** {**ppp** | **slip**}

The configured SLIP or PPP encapsulation method applies to an interface configured for *dedicated* asynchronous mode or dial-on-demand routing (DDR). On an asynchronous interface configured for *interactive* mode, the encapsulation type is specified by the user with the **slip** or **ppp** EXEC command. See "Configure Dedicated or Interactive Mode" later in this section.

In order to use SLIP or PPP, the router must be configured with an IP routing protocol or with the **ip host-routing** command. This configuration is done automatically if you are using old-style **slip address** commands. However, you must configure it manually if you configure SLIP or PPP via the **interface async** command.

Configure the Addressing Method

The **async default** and **async dynamic** commands control whether you can specify an address at the EXEC level using the **slip** and **ppp** commands or whether an address is forced by the system.

It is common to configure an asynchronous interface both to have a default address and to allow dynamic addressing. In this case, you must choose between the default address or dynamic addressing when you enter the {**slip** | **ppp**} **default** or {**slip** | **ppp**} *client* [**@***tacacs-server*] EXEC command.

This section describes how to do the following:

- Assign a default asynchronous address.
- Allow an asynchronous address to be assigned dynamically.

You can assign a permanent default asynchronous address to a line by performing the following task in interface configuration mode:

Task	Command
Assign a default IP address to the asynchronous interface.	**async default ip address** *ip-address*

Use the **no** form of this command to disable the default address. If the server has been configured to authenticate asynchronous connections, you are prompted for a password after entering the **slip** *client* [**@***tacacs-server*] or **ppp** *client* [**@***tacacs-server*] EXEC command before the line is placed into asynchronous mode.

The assigned default address is used when the user enters the **slip default** or **ppp default** EXEC command. The TACACS server validates the transaction (when enabled), and the line is put into network mode using the address that is in the configuration file. This feature is useful when the user is not required to know the IP address to gain access to a system; for example, users of a server that is available to many students on a campus. Instead of requiring each user to know an IP address, they need only enter the **slip default** or **ppp default** EXEC command and let the server select the address to use.

When a line is configured for dynamic assignment of asynchronous addresses, the user enters the **slip** or **ppp** EXEC command and is prompted for an address or logical host name. The TACACS validates the address, when enabled, and the line is assigned the given address and put into asynchronous mode. Assigning asynchronous addresses dynamically is also useful when you want to assign set addresses to users. For example, an application on a personal computer that automatically dials in using SLIP and polls for electronic mail messages can be set up to dial in periodically and enter the required IP address and password.

To configure asynchronous dynamic addressing, perform the following task in interface configuration mode:

Task	Command
Allow the IP address to be assigned at login.	**async dynamic address**

The dynamic addressing features of the internetwork allow packets to get to their destinations and back regardless of the router or network they are sent from. For example, if a host such as a laptop computer moves from place to place, it can keep the same address no matter where it is dialing in from. For an example of configuring asynchronous dynamic addressing, see the section “Example of Asynchronous Routing and Dynamic Addressing” at the end of this chapter.

Configure Dedicated or Interactive Mode

You can configure the asynchronous interface to be in dedicated network or interactive mode.

In dedicated mode, there is no user prompt or EXEC level, so no end-user commands are required to place the line into interface mode. When the interface is configured for dedicated mode, the user cannot change the encapsulation method, address, or other parameters.

To configure an asynchronous interface to be in dedicated network mode, perform the following task in interface configuration mode:

Task	Command
Place the asynchronous line into dedicated network mode.	**async mode dedicated**

For an example of placing an asynchronous interface into dedicated network mode, see the section "Example of a Dedicated Asynchronous Interface" at the end of this chapter.

Alternatively, you can configure an asynchronous line for interactive mode. In interactive mode, the line can be used to make any type of connection, depending on the EXEC command entered by the user. For example, depending on its configuration, the line could be used for Telnet connections, or SLIP or PPP encapsulation. Perform the following task in interface configuration mode to configure an asynchronous line for interactive mode:

Task	Command
Place the asynchronous line in interactive mode.	**async mode interactive**

Enable Asynchronous Routing

You can enable use of dynamic routing protocols on the asynchronous interface by performing the following task in interface configuration mode:

Task	Command
Configure an asynchronous interface for routing.	**async dynamic routing**

Connect to Remote Routers via PPP or SLIP

You can use an asynchronous device as a network interface connection to a remote router via the auxiliary port using the PPP or SLIP protocols. To do so, perform the following task in EXEC mode:

Task	Command
Use the asynchronous device as a network interface from the auxiliary port using the PPP or SLIP protocol.	{**ppp** | **slip**} [**default** | *client* [**@***tacacs-server*]] [**/routing**] [**/compressed**]

To use the asynchronous device as a network interface using PPP or SLIP, the auxiliary port on the remote router must be configured to act as an asynchronous interface. This is done on the remote router with the **interface async 1** command. You must also use the **async mode interactive** command.

With the **default** keyword, you can specify a default IP address for asynchronous routing. For this option to be effective, the asynchronous interface on the remote router must have a default IP address specified with the **async default ip address** command. Or, you can enter an address if the asynchronous interface on the remote router is configured with the **async dynamic address** command.

The client and TACACs server can be specified either by IP address or by name. Any names used must be in the domain name service (DNS) and must be resolvable to an IP address. The address specified for *tacacs-server* must be an address of a TACACS server configured with the **tacacs-server host** command as described in the chapter entitled "Managing the System."

The **/routing** keyword enables dynamic routing. To use it, the asynchronous interface on the remote router must be configured for asynchronous routing with the **async dynamic routing** command.

Using the **/compressed** keyword, you can set the asynchronous interface to compress packets for more efficient use of the line. To use it, the asynchronous interface on the remote router must have TCP/IP parameters set with the **ip tcp header-compression** command.

For examples of the **ppp** and **slip** commands, see the examples at the end of this chapter.

You can also use the **ip access-group** command to apply access restrictions to the asynchronous interface and the **ip unnumbered** command. See the chapter entitled "Configuring IP."

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 adding interface descriptions, see the section "Examples of Interface Descriptions" 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.

The following sections describe subinterface configuration tasks:

- Understand Supported Interfaces and Encapsulations
- Configure Subinterfaces on Serial Interfaces Running Frame Relay Encapsulation

Understand Supported Interfaces and Encapsulations

Subinterfaces can be used to support partially meshed multiprotocol Frame Relay networks over a serial interface.

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

Table 6-1 Subinterface Configuration Commands

Command	Interface Type Supported
apollo	Serial only
appletalk	Serial only
bandwidth	Serial, Ethernet, FDDI, Token Ring
bridge-group	Serial only
clns	Serial only
decnet	Serial only
delay	Serial, Ethernet, FDDI, Token Ring

Command	Interface Type Supported
description	Serial, Ethernet, FDDI, Token Ring
exit	Serial, Ethernet, FDDI, Token Ring
frame-relay	Serial only
ip	Serial only
ipx	Serial, Ethernet, FDDI, Token Ring
isis	Serial only
iso-igrp	Serial only
ntp	Serial only
shutdown	Serial, Ethernet, FDDI, Token Ring

In the following example, interface serial 0.1 is configured to support bridging on Frame Relay DLCI 42:

```
interface serial 0
encapsulation frame-relay
interface serial 0.1
bridge-group 1
frame-relay interface-dlci 42 broadcast
```

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

Table 6-2 Commands That Allow Subinterfaces as Parameters

Command	Command Type
bridge group address	Global configuration
clns route	Global configuration
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

In the following example, the route to IP network 10.0.0.0 is configured to exit the router 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 apple	EXEC
show buffers	EXEC
show clns	EXEC
show decnet	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.

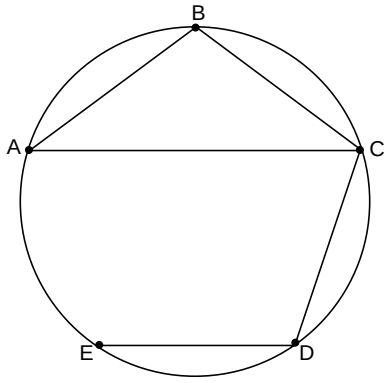
A subinterface with a single Frame Relay PVC is modeled as a point-to-point link. A subinterface with multiple Frame Relay PVCs is modeled as a LAN.

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. Additionally, certain protocols such as AppleTalk and transparent bridging could not be supported on partially meshed networks because they require “split horizon,” in which a packet received on an interface cannot be transmitted out the same interface even if the packet is received and transmitted on different virtual circuits.

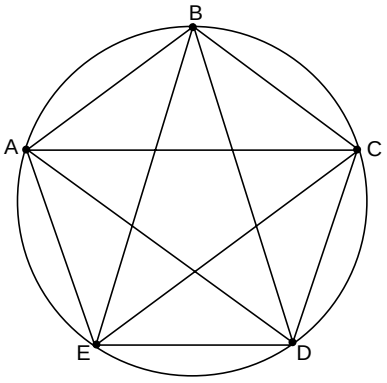
Subinterfaces address these limitations by providing a way to subdivide a partially meshed Frame Relay network into a number of smaller, fully meshed (or point-to-point) subnetworks. Each subnetwork is assigned its own network number and appears to the protocols as if it is reachable through a separate interface. (Note that point-to-point subinterfaces can 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 Figure 6-1) that is partially meshed (Network A). 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. One way to make this work fully is to create a fully meshed network (Network B), but that requires a large number of PVCs, which may not be economically feasible.

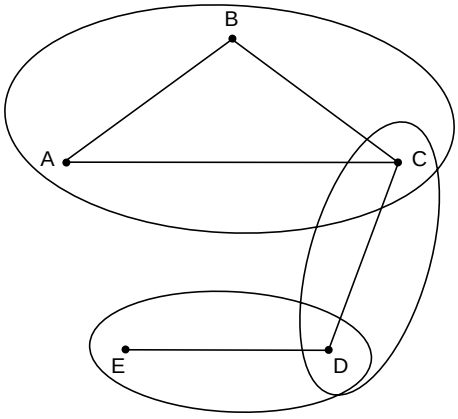
Using subinterfaces, the Frame Relay network can be subdivided into three smaller networks (Network C) with separate network numbers. Nodes A, B, and C are connected to a fully meshed network, and nodes C and D, as well as nodes D and E are connected via point-to-point networks. In this configuration, nodes C and D would see two subinterfaces, allowing them to forward packets without violating split horizon rules. If transparent bridging is being used, each subinterface is viewed as a separate bridge port.



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

Figure 6-1 Using Subinterfaces to Provide Full Connectivity on a Partially Meshed Frame Relay Network

To configure subinterfaces on a Frame Relay network, 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-number.subinterface-number* [**multipoint** | **point-to-point**]
Step 4	Configure the feature you want the subinterface to support.	See the examples that follow.

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
```

To use Frame Relay DLCIs 42, 64, and 73 as separate point-to-point links and run transparent bridging over them, the configuration might look like the following example:

```
interface serial0
encapsulation frame-relay
interface serial 0.1 point-to-point
bridge-group 1
frame-relay interface-dlci 42 broadcast
interface serial 0.2 point-to-point
bridge-group 1
frame-relay interface-dlci 64 broadcast
interface serial 0.3 point-to-point
bridge-group 1
frame-relay interface-dlci 73 broadcast
```

From the bridging spanning tree algorithm's point of view, each PVC is a separate bridge port, and a frame arriving on a PVC can be relayed back out a separate PVC.

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. Because tunnels are point-to-point links, you must configure a separate tunnel for each link.

Tunneling has three primary components:

- Passenger protocol, which is the protocol you are encapsulating (IPX, IP, DECnet, CLNP, or AppleTalk)

- 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
  - EON, a standard for carrying CLNP over IP networks
  - 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 6-2 illustrates IP tunneling terminology and concepts.

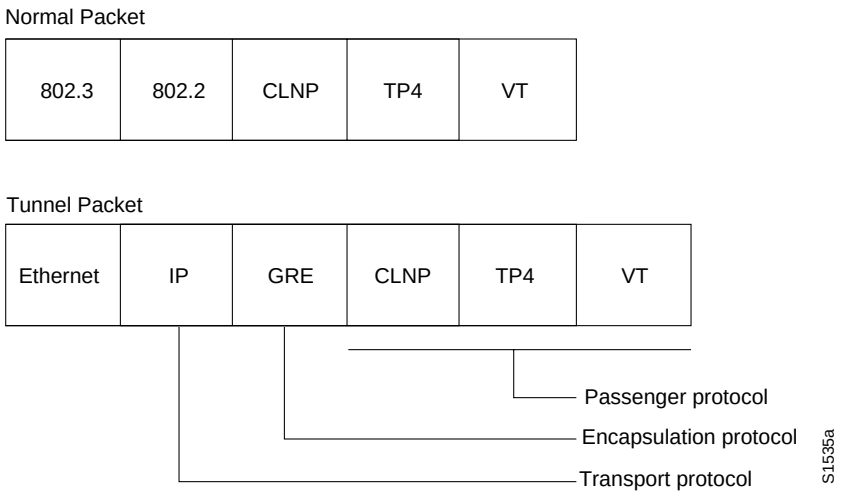


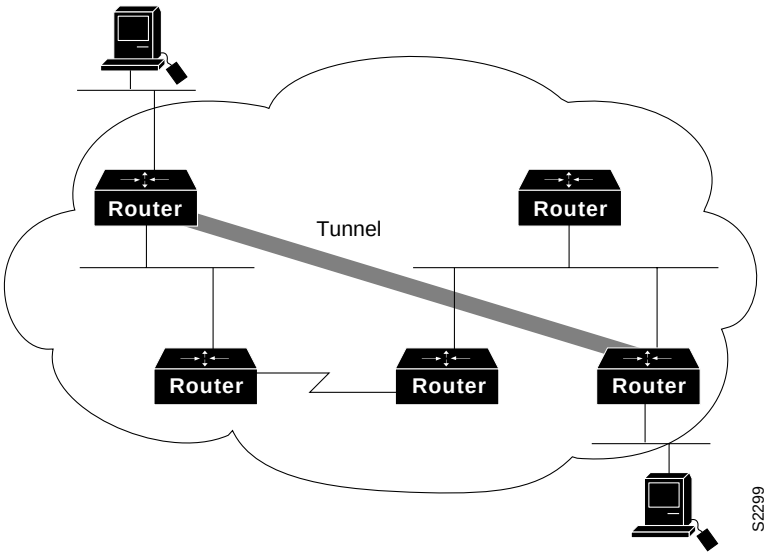
Figure 6-2 IP Tunneling Terminology and Concepts

To understand the process of tunneling, consider connecting two AppleTalk networks with a non-AppleTalk backbone, such as IP. The relatively high bandwidth consumed by the broadcasting of Routing Table Maintenance Protocol (RTMP) data packets can severely hamper the backbone's network performance. This problem can be solved by tunneling AppleTalk through a foreign protocol, such as IP. Tunneling encapsulates an AppleTalk packet inside the foreign protocol packet, which is then sent across the backbone to a destination router. The destination router then de-encapsulates the AppleTalk packet and, if necessary, routes the packet to a normal AppleTalk network. Because the encapsulated AppleTalk packet is sent in a directed manner to a remote IP address, bandwidth usage is greatly reduced. Furthermore, the encapsulated packet benefits from any features normally enjoyed by IP packets, including default routes and load balancing.

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 6-3)
- To connect discontinuous subnetworks
- To allow virtual private networks across wide-area networks (WANs)



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.

Figure 6-3 Providing Workarounds for Networks with Limited Hop Counts

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 routers.
- Tunneling might create problems with transport protocols with limited timers (for example, DECnet) due to increased latency.
- Be aware of the environments across which you create tunnels. You might be tunneling across fast FDDI rings or through slow 9600-bps phone lines; some passenger protocols behave poorly in mixed media networks.
- 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 6-4, 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.

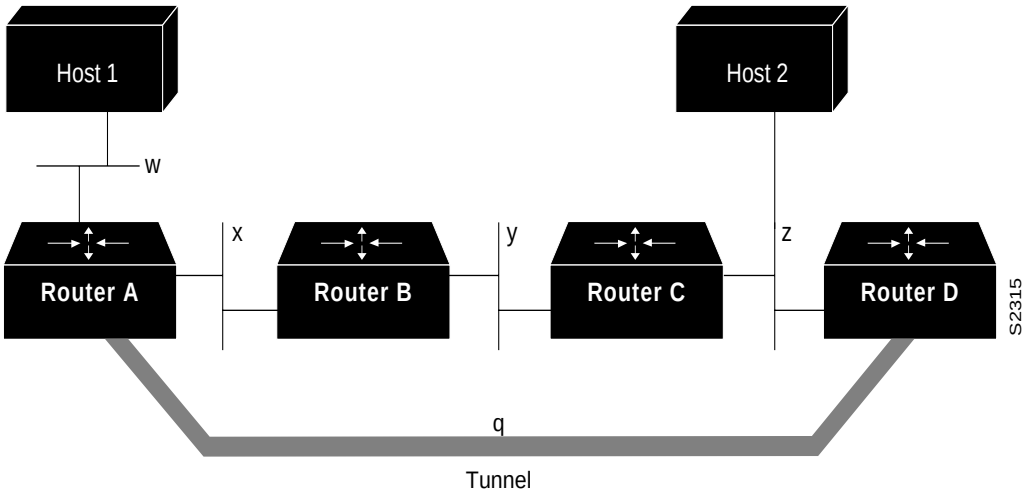


Figure 6-4 Tunnel Precautions: Hop Counts

- 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 might 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 following sections. 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 Checksumming
- Configure a Tunnel Identification Key
- Configure a Tunnel Interface to Drop Out-of-Order Datagrams
- Monitor IP Tunnels

For examples of configuring tunnels, see the section “Examples of IP Tunneling” at the end of this chapter.

Configure the Tunnel Interface

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

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

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 ip** | **nos**}

If you are tunneling AppleTalk, you must use either Cayman or GRE tunneling mode. Cayman tunneling is designed by Cayman Systems, and enables routers to interoperate with Cayman GatorBoxes. You can have our routers at either end of the tunnel, or you can have a GatorBox at one end and our router at the other end.

Caution Do not configure a Cayman tunnel with an AppleTalk network address.

If you use GRE, you must have only our routers at both ends of the tunnel connection. When using GRE to tunnel AppleTalk, you must configure an AppleTalk network address and a zone. Perform the following tasks to tunnel AppleTalk using GRE:

Task		Command
Step 1	Enable tunneling on the interface.	**interface tunnel***n*
Step 2	Assign a cable range to an interface.	**appletalk cable-range** *start-end* [*network.node*][1]
Step 3	Set a zone name for the connected AppleTalk network.	**appletalk zone** *zone-name*[2]
Step 4	Specify the interface out which the encapsulated packets will be sent, or specify the router's IP address.	**tunnel source** [*interface* | *ip-address*]
Step 5	Specify the IP address of the router at the far end of the tunnel.	**tunnel destination** *ip-address*
Step 6	Enable GRE tunneling.	**tunnel mode gre ip**

1. This command is documented in the "AppleTalk Commands" chapter of the *Router Products Command Reference* publication.
2. This command is documented in the "AppleTalk Commands" chapter of the *Router Products Command Reference* publication.

Configure End-to-End Checksumming

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 routers 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 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 GRE only.

To use this option, 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 interfaces tunnel** *unit* [**accounting**]
List the routes that go through the tunnel.	**show** *protocol* **route**[1]
List the route to the tunnel destination.	**show ip route**[2]

1. This command is documented in a separate chapter for each protocol. For example, the command **show clns route** is documented in the "ISO CLNS Commands" chapter of the *Router Products Command Reference* publication.
2. This command is documented in the "IP Commands" chapter of the *Router Products Command Reference* publication.

Configure Synchronous Serial Features

The optional tasks in this section configure features on a synchronous serial interface.

- Reenable HDLC Serial Encapsulation
- Configure Compression of LAPB Data
- Configure the CRC
- Use the NRZI Line-Coding Format
- Enable the Internal Clock
- Invert the Transmit Clock Signal
- Set Transmit Delay
- Configure DTR Signal Pulsing
- Configure the Clock Rate on DCE Appliques
- Specify the Serial Network Interface Module Timing

Reenable HDLC Serial Encapsulation

The router 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**

Configure Compression of LAPB Data

You can configure point-to-point software compression on serial interfaces that use LAPB or multi-LAPB encapsulation. Compression reduces the size of a LAPB or multi-LAPB frame via lossless data compression. The compression algorithm used is a predictor algorithm (the RAND algorithm), which uses a compression dictionary to predict what the next character in the frame will be.

Compression is performed in software and may significantly affect system performance. We recommend that you disable compression if CPU load exceeds 65%. To display the CPU load, use the **show process cpu** EXEC command.

If the majority of your traffic is already compressed files, it is recommended that you not use compression.

To configure compression over LAPB, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Enable encapsulation of a single protocol on the serial line.	**encapsulation lapb** or **encapsulation lapb-dce**
Step 2	Enable compression.	**compress predictor**

To configure compression over multi-LAPB, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Enable encapsulation of multiple protocols on the serial line.	**encapsulation multi-lapb** or **encapsulation multi-lapb-dce**
Step 2	Enable compression.	**compress predictor**

When using compression, the MTU for the serial interface and the LAPB N1 parameter should be adjusted as in the following example, in order to avoid informational diagnostics regarding excessive MTU or N1 sizes:

```
interface serial 0
encapsulation lapb
compress predictor
mtu 1509
lapb n1 12072
```

Configure the CRC

The cyclic redundancy check (CRC) on a serial interface defaults to a length of 16 bits. To change the length of the CRC to 32 bits on an FSIP or HIP of the Cisco 7000 series only, complete the following task in interface configuration mode:

Task	Command
Set the length of the CRC.	**crc** *size*

Use the NRZI Line-Coding Format

Many serial interfaces—including the Hitachi-based interfaces in the Cisco 2500 series, the 4T processor modules, and the Cisco 3000 series with dual serial ports, and all FSIP interface types on the Cisco 7000— support nonreturn to zero (NRZ) and nonreturn to zero inverted (NRZI) format. This is a line-coding format that is required for serial connections in some environments. NRZ encoding is most common. NRZI encoding is used primarily with RS-232 connections in IBM environments.

The default configuration for all serial interfaces is NRZ format. The default is **no nrzi-encoding**. To enable NRZI format, complete the following task in interface configuration mode:

Task	Command
Enable NRZI encoding format.	**nrzi-encoding**

Enable the Internal Clock

When a DTE does not return a transmit clock, use the following interface configuration command on the Cisco 7000 to enable the internally generated clock on a serial interface:

Task	Command
Enable the internally generated clock on a serial interface.	**transmit-clock-internal**

Invert the Transmit Clock Signal

Delays between the SCTE clock and data transmission indicate that the transmit clock signal might not be appropriate for the interface rate and length of cable being used. Different ends of the wire may have variances that differ slightly. Invert the clock signal to compensate for these factors by completing the following task in interface configuration mode on a Cisco 7000:

Task	Command
Invert the clock signal on an interface.	**invert-transmit-clock**

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 and for the HSSI. Perform one of the following tasks, 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*
Set the transmit delay on the HSSI.	**transmitter-delay** *hdlc-flags*

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.	**clock rate** *bps*

Specify the Serial Network Interface Module Timing

On the Cisco 4000 platform, you can specify the serial Network Interface Module timing signal configuration. When the board is operating as a DCE and the DTE provides terminal timing (SCTE or TT), you can configure the DCE to use SCTE from the DTE. When running the line at high speeds and long distances, this strategy prevents phase shifting of the data with respect to the clock.

To configure the DCE to use SCTE from the DTE, perform the following task in interface configuration mode:

Task	Command
Configure the DCE to use SCTE from the DTE.	**dce-terminal-timing enable**

When the board is operating as a DTE, you can invert the TXC clock signal it gets from the DCE that the DTE uses to transmit data. Invert the clock signal if the DCE cannot receive SCTE from the DTE, the data is running at high speeds, and the transmission line is long. Again, this prevents phase shifting of the data with respect to the clock.

To configure the interface so that the router inverts the TXC clock signal, perform the following task in interface configuration mode:

Task	Command
Specify timing configuration to invert TXC clock signal.	**dte-invert-txc**

Select Ethernet Encapsulation

Ethernet interfaces on the router 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**

For an example of selecting Ethernet encapsulation, see the section “Example of Enabling Ethernet Encapsulation” at the end of this chapter.

Configure the Ethernet Network Interface Module on the Cisco 4000

You can specify the type of Ethernet Network Interface Module configuration on the Cisco 4000 model. To do so, perform the following task in interface configuration mode:

Task	Command
Select a 15-pin Ethernet connector.	**media-type aui**
Select an RJ45 Ethernet connector.	**media-type 10baset**

Extend the 10BaseT Capability on the Cisco 4000

You can extend the twisted-pair 10BaseT capability beyond the standard 100 meters by reducing the squelch (signal cutoff time). To do so, perform the first task that follows in interface configuration mode. You can later restore the squelch by performing the second task.

Task	Command
Reduce the squelch.	**squelch reduced**
Return squelch to normal.	**squelch normal**

Configure a Hub

The Cisco 2500 series includes routers that have hub functionality for an Ethernet interface. The hub is a multiport repeater. The advantage of having an Ethernet interface connected to a hub is to have star wiring (with 10BaseT) capability. The router models with hub ports and their configurations are as follows:

- Cisco 2505 - 8 Ethernet ports and 2 serial ports
- Cisco 2507- 16 Ethernet ports and 2 serial ports

We provide SNMP management of the Ethernet hub per RFC 1516.

To control hub functionality on an Ethernet interface, perform the tasks in the following sections. The first task is required; the remaining are optional.

- Enable a Hub Port
- Disable or Enable Automatic Receiver Polarity Reversal (Ethernet only)
- Disable or Enable the Link Test Function (Ethernet only)
- Enable Source Address Control (Ethernet only)

Enable a Hub Port

To enable a hub port, perform the following task in global configuration mode. Because there is only one hub connected to the single Ethernet interface, the hub *number* will be 0. The *port* numbers range from 1 through 8 or 1 through 16, depending on which router model you are configuring. The second port number is used for a range of contiguous hub ports.

Task	Command
Specify the hub number and the hub port (or range of hub ports).	**hub ether** *number port* [*port*]
Enable the hub port(s).	**no shutdown**

See the examples of enabling a hub at the end of this chapter.

Disable or Enable Automatic Receiver Polarity Reversal

On Ethernet hub ports only, the hub ports can invert, or correct, the polarity of the received data if the port senses that the received data packet waveform polarity is reversed due to a wiring error. This receive circuitry polarity correction allows the hub to repeat subsequent packets with correct polarity. When enabled, this function is executed once after reset of a link fail state.

Automatic receiver polarity reversal is enabled by default. To disable this feature on a per-port basis, perform the following task in hub configuration mode:

Task	Command
Disable automatic receiver polarity reversal.	**no auto-polarity**

To re-enable automatic receiver polarity reversal on a per-port basis, perform the following task in hub configuration mode:

Task	Command
Re-enable automatic receiver polarity reversal.	**auto-polarity**

Disable or Enable the Link Test Function

This feature applies to Ethernet hub ports only. The Ethernet ports implement the Link Test function as specified in the 802.3 10BaseT standard. The hub ports will transmit Link Test pulses to any attached twisted pair device if the port has been inactive for more than 8 to 17 milliseconds.

If a hub port does not receive any data packets or Link Test pulses for more than 65 to 132 milliseconds and the link test function is enabled for that port, that port will enter link fail state and be disabled from transmit and receive functions. The hub port will be re-enabled when it receives four consecutive Link Test pulses or a data packet.

The link test function is enabled by default. To allow the hub to interoperate with pre-10BaseT twisted pair networks that do not implement the Link Test function, the hub's Link Test receive function can be disabled on a per-port basis. To do so, perform the following task in hub configuration mode:

Task	Command
Disable the link test function.	**no link-test**

To re-enable the link test function on a hub port connected to an Ethernet interface, perform the following task in hub configuration mode:

Task	Command
Enable the link test function.	**link-test**

Enable Source Address Control

On an Ethernet hub port only, you can configure a security measure so that the port accepts packets only from a specific MAC address. For example, suppose your workstation is connected to port 3 on a hub, and source address control is enabled on port 3. Your workstation has access to the network because the hub accepts from port 3 any packet bearing your workstation's MAC address. Any packets arriving with a different MAC address cause the port to be disabled. The port is then re-enabled after 1 minute and again the MAC address of incoming packets is checked.

To enable source address control on a per-port basis, perform the following task in hub configuration mode:

Task	Command
Enable source address control.	**source-address** [*mac-address*]

If you omit the optional MAC address, the hub remembers the first MAC address it receives on the selected port, and allows only packets from the learned MAC address.

See the examples of establishing source address control at the end of this chapter.

Configure ATM-DXI

You can map a virtual path identifier (vpi) and virtual channel identifier (vci) to a Frame Relay DLCI by performing the following tasks in interface configuration mode:

Task		Command
Step 1	Specify the encapsulation method.	**encapsulation atm-dxi**
Step 2	Map a given VPI and VCI to a Frame Relay DLCI.	**atm-dxi map** *protocol address vpi vci* [**broadcast**]

Convert HSSI to Clock Master

You can convert the HSSI interface into a 45 MHz clock master by performing the following task in interface configuration mode:

Task	Command
Convert the HSSI interface into a 45 MHz clock master.	**hssi internal-clock**

Configure MOP

Perform the tasks in this section to configure the Maintenance Operation Protocol (MOP).

Enable MOP

You can enable 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**

Configure Token Ring Features

Perform the tasks in this section to configure Token Ring features.

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*

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 ISDN Features

The tasks in this section configure ISDN features. If you configure an ISDN BRI interface, you must select an ISDN BRI switch type.

- Select the ISDN BRI Switch Type
- Define ISDN Service Profile Identifiers (SPIDs)
- Define ISDN TEI Negotiation
- Configure ISDN Caller ID Screening

Select the ISDN BRI Switch Type

ISDN BRI supports a variety of central office switches. Table 6-4 lists the ISDN switch types supported by the ISDN interface. If you configure an interface with the **interface bri** command, you must also select a switch. Perform the following task in global configuration mode:

Task	Command
Select the central office switch type.	**isdn switch-type** *switch-type*

Table 6-4 ISDN Office Switch Types

Keyword	Switch Type
basic-1tr6	German 1TR6 ISDN switches
basic-5ess	AT&T basic rate switches
basic-dms100	NT DMS-100 basic rate switches
basic-net3	NET3 ISDN switches (UK and others)
basic-ni1	National ISDN-1 switches
basic-nwnet3	Norway Net3 switches
basic-nznet3	New Zealand Net3 switches
basic-ts013	Australian TS013 switches
none	No switch defined
ntt	Japanese NTT ISDN switches
vn2	French VN2 ISDN switches
vn3	French VN3 ISDN switches

Define ISDN Service Profile Identifiers (SPIDs)

All ISDN devices subscribe to services provided by an ISDN service provider, usually a telephone company. However, only some service providers use Service Profile Identifiers (SPIDs) to define the services subscribed to by the ISDN device that is accessing the ISDN service provider. The service provider assigns the ISDN device one or more SPIDs when you first subscribe to the service. If you are using a service provider that requires SPIDs, your ISDN device cannot place or receive calls until it sends a valid, assigned SPID to the service provider when accessing the switch to initialize the connection.

Currently, only the DMS-100 and NI-1 switch types require SPIDs. The AT&T 5ESS switch type may support a SPID, but it is recommended that you set up that ISDN service without SPIDs. In addition, SPIDs only have significance at the local access ISDN interface. Remote routers are never sent the SPID.

A SPID is usually a ten-digit telephone number with some optional numbers. However, service providers may use different numbering schemes. For the DMS-100 switch type, two SPIDs are assigned, one for each B-channel. Once your service provider has assigned you SPIDs, you must define these SPIDs on the router so that when access to the switch is attempted, the router has the valid information available.

To define the SPIDs and the local directory number (LDN) on the router, perform the following tasks in interface configuration mode (after specifying **interface bri 0**):

Task	Command
Define a SPID and local directory number for the B1-channel	**isdn spid1** *spid-number* [*ldn*]
Define a SPID and local directory number for the B2-channel	**isdn spid2** *spid-number* [*ldn*]

See the chapter entitled “Configuring DDR” for information about configuring dial-on-demand routing (DDR). Refer to the chapter entitled “DDR Commands” in the *Router Products Command Reference* publication for specific DDR commands.

Define ISDN TEI Negotiation

You can determine when Layer 2 ISDN Terminal Endpoint Identifier (TEI) negotiation occurs. The default is for negotiation to occur when the router is powered on. Be careful when determining TEI negotiation.

To define when TEI negotiation will occur, perform the following task in global configuration mode:

Task	Command
Determine when ISDN TEI negotiation occurs.	**isdn tei** [**first-call** | **powerup**]

Configure ISDN Caller ID Screening

Caller ID screening adds a level of security by allowing you to screen incoming calls. You can verify that the calling line ID is from an expected origin. Caller ID screening requires a local switch that is capable of delivering the caller ID to the router. This feature is available only on Cisco 2500 series and Cisco 3000 series routers that have a BRI interface.

To configure caller ID screening, perform the following task in interface configuration mode:

Task	Command
Configure caller ID screening.	**isdn caller** *number*

Configure Channelized T1

The sections that follow show how to perform each step of the T1 configuration process. See the end of this chapter for configuration examples.

Using channelized T1 controller and serial interface configuration commands, you can perform the tasks in the following sections:

- Configure the T1 Controller
- Define the Line Code
- Define the Framing Characteristics
- Define the Clock Source
- Define the T1 Channel Groups
- Configure the T1 Interface

Configure the T1 Controller

To configure the T1 physical characteristics, you first define the physical location of the MIP and CxCT1 in the Cisco 7000 series router. A Cisco 7000 router can have up to four MIP and eight CxCT1 interfaces. A Cisco 7010 router can have up to three MIP and six CxCT1 interfaces. Perform the following task in global configuration mode to define the T1 controller and to enter controller configuration mode:

Task	Command
Define the MIP and CxCT1 locations in the Cisco 7000 by slot and port number.	**controller t1** *slot/port*

Define the Line Code

Perform the following task in controller configuration mode to define the line code as either alternate mark inversion (AMI) or bipolar 8 zero substitution (B8ZS):

Task	Command
Define the line code as either AMI or B8ZS.	**linecode** {**ami** | **b8zs**}

Contact your local telephone service provider to determine the line code requirements of the physical T1 line. The T1 controller values must match the service provided by the telephone company.

Define the Framing Characteristics

Perform the following task in controller configuration mode to define the framing characteristics as either super frame (SF) or extended super frame (ESF):

Task	Command
Define the framing characteristics as either SF or ESF.	**framing** {**sf** | **esf**}

Contact your local telephone service provider to determine the framing requirements of the physical T1 line. The T1 controller values must match the service provided by the telephone company.

Define the Clock Source

You must define the clock source only when connecting two routers back-to-back for testing purposes. Perform the following task in controller configuration mode:

Task	Command
Define the clock source if you are connecting two MIP cards back-to-back for testing purposes. Under normal circumstances, the clock source will always be line.	**clock source** {**line** | **internal**}

The clock source normally comes from the T1 line rather than from the router interface. When you connect two routers back-to-back for testing purposes, one router will supply an internal clock source.

Define the T1 Channel Groups

You must define the timeslots that belong with each channel-group that the controller supports. Timeslots are numbered 1 to 24, and channel groups are numbered 0 to 23. Perform the following task in controller configuration mode to define the channel-groups and timeslots:

Task	Command
Define the channel-group number and, if needed, circuit speed.	**channel-group** *number* **timeslots** *range* [**speed** {**48** | **56** | **64**}]

Your local telephone service provider determines how many timeslots are in a channel group. The channel-group numbers for each T1 controller can be arbitrarily assigned. In the United States, T1 speeds can be either 56 Kbps or 64 Kbps. The default speed is 56 Kbps. Other countries also support 48 Kbps. The speed you select must match the speed provided by the telephone company.

Configure the T1 Interface

After you define the T1 circuits, you can configure each channel group as a serial interface. In other words, you can think of each circuit as being a virtual serial interface. Subinterface configuration is supported. Perform the following task either in global configuration mode or controller configuration mode to enter interface configuration mode and configure the serial interface that corresponds to a channel group:

Task	Command
Define the serial interface for a T1 circuit.	**interface serial** *slot*/*port*:*channel-group*

Configure FDDI Timers and Features

Using special FDDI interface configuration commands, you can perform the tasks in the following sections:

- Enable FDDI Bridging Encapsulation
- Set the Token Rotation Time
- Set the Transmission Valid Timer
- Control the Transmission Timer
- Modify the C-Min Timer
- Modify the TB-Min Timer
- Modify the FDDI Timeout Timer
- Control SMT Frame Processing
- Enable Duplicate Address Checking
- Set the Bit Control
- Control the CMT Microcode
- Start and Stop FDDI
- Configure FDDI Dual Homing
- Control the FDDI SMT Message Queue Size
- Preallocate Buffers for Bursty FDDI Traffic

Enable FDDI Bridging Encapsulation

The CSC-C2/FCIT interface card and FIP fully support transparent and translational bridging for the following configurations:

- FDDI to FDDI
- FDDI to Ethernet
- FDDI to Token Ring

Enabling FDDI bridging encapsulation places the CSC-C2/FCIT interface or FIP into encapsulation mode when doing bridging. In transparent mode, the FCIT interface or FIP interoperates with earlier versions of the CSC-FCI encapsulating interfaces when performing bridging functions on the same ring. When using the CSC-C2/FCIT interface card or FIP, you can specify the encapsulation method by performing the following task in interface configuration mode:

Task	Command
Specify the encapsulation method for the CSC-C2/FCIT interface card or FIP.	**fddi encapsulate**
Turn off encapsulation bridging and return the FCIT interface or FIP to its translational, nonencapsulating mode.	**no fddi encapsulate**

When you are translationally bridging, you have to route routable protocols and translationally bridge the rest (such as LAT).

The CSC-FCI interfaces are always in encapsulating bridge mode, so disabling applies only to CSC-C2/FCIT interfaces.

Note Bridging between dissimilar media presents several problems that can prevent communications. These problems include bit-order translation (or use of MAC addresses as data), maximum transfer unit (MTU) differences, frame status differences, and multicast address usage. Some or all of these problems might be present in a multimedia-bridged LAN and might prevent communication. These problems are most prevalent when bridging between Token Rings and Ethernets or between Token Rings and FDDI nets. This is because of the different way Token Ring is implemented by the end nodes.

We are currently aware of problems with the following protocols when bridged between Token Ring and other media: AppleTalk, DECnet, IP, Novell IPX, Phase IV, VINES, and XNS. Further, the following protocols may have problems when bridged between FDDI and other medias: Novell IPX and XNS. We recommend that these protocols be routed whenever possible.

Set the Token Rotation Time

You can set the FDDI token rotation time to control ring scheduling during normal operation and to detect and recover from serious ring error situations. To do so, perform the following task in interface configuration mode:

Task	Command
Set the FDDI token rotation time.	**fddi token-rotation-time** *microseconds*

The FDDI standard restricts the allowed time to be greater than 4000 microseconds and less than 165,000 microseconds. As defined in the X3T9.5 specification, the value remaining in the token rotation timer (TRT) is loaded into the token holding timer (THT). Combining the values of these two timers provides the means to determine the amount of bandwidth available for subsequent transmissions.

Set the Transmission Valid Timer

You can set the transmission timer to recover from a transient ring error by performing the following task in interface configuration mode:

Task	Command
Set the FDDI valid transmission timer.	**fddi valid-transmission-time** *microseconds*

Control the Transmission Timer

You can set the FDDI control transmission timer to control the FDDI TL-Min time, which is the minimum time to transmit a Physical Sublayer or PHY line state before advancing to the next Physical Connection Management or PCM state as defined by the X3T9.5 specification. To do so, perform the following task in interface configuration mode:

Task	Command
Set the FDDI control transmission timer.	**fddi tl-min-time** *microseconds*

Modify the C-Min Timer

You can modify the C-Min timer on the PCM from its default value of 1600 microseconds by performing the following task in interface configuration mode:

Task	Command
Set the c-min timer on the PCM.	**fddi c-min** *microseconds*

Modify the TB-Min Timer

You can change the TB-Min timer in the PCM from its default value of 100 milliseconds. To do so, perform the following task in interface configuration mode:

Task	Command
Set TB-Min timer in the PCM.	**fddi tb-min** *milliseconds*

Modify the FDDI Timeout Timer

You can change the FDDI timeout timer in the PCM from its default value of 100 milliseconds. To do so, perform the following task in interface configuration mode:

Task	Command
Set the timeout timer in the PCM.	**fddi t-out** *milliseconds*

Control SMT Frame Processing

You can disable and reenable SMT frame processing for diagnostic purposes. To do so, perform the following task in interface configuration mode:

Task	Command
Disable SMT frame processing.	**no fddi smt-frames**
Enable SMT frame processing.	**fddi smt-frames**

Enable Duplicate Address Checking

You can enable the duplicate address detection capability on the FDDI. If the FDDI finds a duplicate address, it displays an error message and shuts down the interface. To enable duplicate address checking, perform the following task in interface configuration mode:

Task	Command
Enable duplicate address checking capability.	**fddi duplicate-address-check**

Set the Bit Control

You can set the FDDI bit control to control the information transmitted during the Connection Management (CMT) signaling phase. To do so, perform the following task in interface configuration mode:

Task	Command
Set the FDDI bit control.	**fddi cmt-signal-bits** *signal-bits* [**phy-a** | **phy-b**]

Control the CMT Microcode

You can control whether the CMT onboard functions are on or off. The CSC-FCI and CSC-C2/FCIT interface cards and FIP provide CMT functions in microcode. These functions are separate from those provided on the processor card and are accessed through EXEC commands.

The default is for the FCIT and FIP CMT functions to be on. A typical reason to disable is when you work with new FDDI equipment and have problems bringing up the ring. If you disable the CMT microcode, the following actions occur:

- The FCIT or FIP CMT microcode is disabled.
- The main system code performs the CMT function while debugging output is generated.

To disable the CMT microcode, perform the following task in interface configuration mode:

Task	Command
Disable the FCIT CMT functions.	**no fddi if-cmt**

Start and Stop FDDI

In normal operation, the FDDI interface is operational once the interface is connected and configured. You can start and stop the processes that perform the CMT function and allow the ring on one fiber to be stopped. To do so, perform either of the following tasks in EXEC mode:

Task	Command
Start CMT processes on FDDI ring.	**cmt connect** [*interface-name* [**phy-a** | **phy-b**]]
Stop CMT processes on FDDI ring.	**cmt disconnect** [*interface-name* [**phy-a** | **phy-b**]]

Do not do either of the preceding tasks during normal operation of FDDI; they are performed during interoperability tests.

Configure FDDI Dual Homing

FDDI interface configuration is not required for dual homing. The FDDI interface recognizes that it is attached to two M ports on the concentrators and automatically supports dual homing.

Control the FDDI SMT Message Queue Size

You can set the maximum number of unprocessed FDDI Station Management (SMT) frames that will be held for processing. Setting this number is useful if the router you are configuring gets bursts of messages arriving faster than the router can process them. To set the number of frames, perform the following task in global configuration mode:

Task	Command
Set SMT message queue size.	**smt-queue-threshold** *number*

Preallocate Buffers for Bursty FDDI Traffic

The FCI card preallocates three buffers to handle bursty FDDI traffic (for example. NFS bursty traffic). You can change the number of preallocated buffers by performing the following task in interface configuration mode:

Task	Command
Preallocate buffers to handle bursty FDDI traffic.	**fddi burst-count**

Configure PPP

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 current implementation of PPP supports option 3, authentication using CHAP or PAP, option 4, Link Quality Monitoring and option 5, Magic Number configuration options. The software always sends option 5 and will negotiate for options 3 and 4 if so configured. All other options are rejected.

We support the following upper-layer protocols: AppleTalk, Bridging, CLNS, DECnet, IP, IPX, VINES, and XNS.

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

Enable PPP Encapsulation

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

Task	Command
Enable 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 CHAP or PAP Authentication

Access control using Challenge Handshake Authentication Protocol (CHAP) or Password Authentication Protocol (PAP) is available on all serial interfaces. The authentication feature reduces the risk of security violations on your router. You can configure either CHAP or PAP for the interface.

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

When CHAP is enabled on an interface, the local router sends a CHAP packet and the remote device (a PC, workstation, or server) attempting to connect to the router 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 router.

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

When PAP is enabled, the remote router attempting to connect to the local router is required to send an authentication request. Unlike CHAP, the remote router initiates the sequence. If the username and password specified in the authentication request are accepted, the router sends an authentication acknowledgment.

To use CHAP or PAP, perform the following tasks:

- Enable CHAP or PAP on the interface. Once you have done so, the local router requires a password from remote devices that are calling in. If the remote device does not support the authentication method, no traffic will be passed to that device.
- Configure server host-name authentication. Configure the secret or password for each remote system for which authentication is required.

To enable CHAP or PAP on an interface configured for PPP encapsulation, perform one of the following tasks in interface configuration mode:

Task	Command
Enable CHAP on an interface.	**ppp authentication chap**
Enable PAP on an interface.	**ppp authentication pap**

To specify the password to be used in CHAP or PAP caller identification, perform the following task in global configuration mode:

Task	Command
Configure authentication.	**username** *name* **password** *secret*[1]

1. This command is documented in the "System Management Commands" chapter of the *Router Products Command Reference* publication.

Make sure that this password does not contain spaces or underscores.

For an example of CHAP, see the section "Example of CHAP with an Encrypted Password" at the end of this chapter. CHAP and PAP are specified in the IETF RFC 1334 "The PPP Authentication Protocols." 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 will monitor the link quality, and if the quality drops below a configured percentage, the link will be 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."

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.

PPP Magic Number Support

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 use of the **down-when-looped** command.

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 modem, CSU/DSU device, or ISDN terminal adapter (TA) attached to a circuit-switched service be connected on the secondary serial interface. The external device must be capable of responding to a DTR signal (DTR active) by auto-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, CSU/DSU, or ISDN TA must be set to dial the remote site. When that connection is made, the routing protocol defined for the serial line will continue the job of transmitting traffic over the dialup line.

You can also 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.

You can also specify a value that defines when the secondary line should be disabled and the amount of time the secondary line can take going up or down.

To configure dial backup, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Select a serial interface as a backup line. On a Cisco 7000:	**backup interface** *interface-name* **backup interface** *type slot/port*
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	Define how much time should elapse before a secondary line is set up or taken down (after a primary line transitions).	**backup delay** {*enable-delay* | **never**} {*disable-delay* | **never**}

See examples of configuring dial backup service in the sections "Examples of Dial Backup Service When the Primary Line Goes Down," "Examples of Dial Backup Service When the Primary Line Reaches Threshold," and "Examples of Dial Backup Service When the Primary Line Exceeds Threshold" at the end of this chapter.

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, you should not have loopback detection configured, or packets will not be transmitted out the interface that is being tested.

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, it 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 an interface.	**delay** *tens-of-microseconds*

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

Adjust Timers

To adjust the frequency of update messages, perform the following the following task in interface configuration mode:

Task	Command
Adjust the frequency with which the router sends messages to itself (Ethernet and Token Ring) or to the other end (HDLC-serial and PPP-serial links) to ensure that a network interface is alive for a specified interface.	**keepalive** [*seconds*]

You also can configure the *keepalive* interval, the frequency at which the router sends messages to itself (Ethernet and Token Ring) or to the other end (HDLC-serial, PPP-serial) to ensure that a network interface is alive. The interval in some previous software versions was 10 seconds; it is now adjustable in one-second increments down to one second. An interface is declared down after three update intervals have passed without receiving a keepalive packet.

When adjusting the keepalive timer for a very low bandwidth serial interface, large packets can delay the smaller keepalive packets long enough to cause the line protocol to go down. You might need to experiment to determine the best value.

Limit Transmit Queue Size

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 tasks in the following sections to monitor and maintain the interfaces:

- Monitor Interface Status
- Monitor the Interface Port
- Monitor the T1 Interface
- Clear and Reset the Interface
- Shut Down and Restart an Interface
- Run Interface Loopback Diagnostics

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 *Router Products Command Reference* publication.

Perform the following commands in EXEC mode:

Task	Command
Display the status of the asynchronous interface.	**show async status**
Display compression statistics on a serial interface.	**show compress**
Display current internal status information for the interface controller cards.	**show controllers** {**bri** | **cbus** | **fddi** | **lance** | **mci** | **pcbus** | **serial** | **token**}
For the Cisco 7000.	**show controllers** {**cxbus** | **fddi** | **serial** | **t1** | **token**}
Display the number of packets of each protocol type that have been sent through the interface.	**show interfaces** [*type* {*unit*}] [*first*] [*last*] [**accounting**]
For the Cisco 7000.	**show interfaces** [*type slot/port*] [**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 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**[1]

1. This command is documented in the "System Image, Microcode Image, and Configuration File Load Commands" chapter of the *Router Products Command Reference* publication.

Monitor the Interface Port

This section applies to the Cisco 7000 series only. The port adapter cable connected to each port determines the electrical interface type and mode of the port. The default mode of the ports is DCE, which allows you to perform a loopback test on any port without having to attach a port adapter cable. Although DCE is the default, there is no default clock rate set on the interfaces. When there is no cable attached to a port, the software actually identifies the port as "Universal, Cable Unattached" rather than either as a DTE or DCE interface.

Use the **show controller cxbus** command to show information about the interface port. The following example shows an interface port (2/0) that has an RS-232 DTE cable attached and a second port (2/1) that does not have a cable attached:

```
7000# show controller cxbus

Switch Processor 7, hardware version 11.1 microcode version 1.4
 512 Kbytes of main memory, 128 Kbytes cache memory, 299 1520 byte buffers
 Restarts: 0 line down, 0 hung output, 0 controller error
FSIP 2, hardware version 3, microcode version 1.0
 Interface 16 - Serial2/0, electrical interface is RS-232 DTE
  31 buffer RX queue threshold, 101 buffer TX queue limit, buffer size 1520
  Transmitter delay is 0 microseconds
 Interface 17 -Serial2/1, electrical interface is Universal (cable unattached)
  31 buffer RX queue threshold, 101 buffer TX queue limit, buffer size 1520
```

To change the electrical interface type or mode of a port online, replace the serial adapter cable and use software commands to restart the interface and, if necessary, reconfigure the port for the new interface. At system startup or restart, the FSIP polls the interfaces and determines the electrical interface type of each port (according to the type of port adapter cable attached). However, it does

not necessarily repoll an interface when you change the adapter cable online. To ensure that the system recognizes the new interface type, shut down and reenable the interface after changing the cable.

Monitor the T1 Interface

This section applies to the Cisco 7000 series only. Because the T1 line itself is viewed as the controller, perform the following task in EXEC mode to display information about activity on the T1 line.

Task	Command
Display information about the T1 line.	**show controller t1**

Alarms, line conditions, and other errors are displayed. The data is updated every 10 seconds. Every 15 minutes, the cumulative data is stored and retained for 24 hours. This means at any one time, up to 96 15-minute accumulations are counted in the data display.

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*]
Clear interface counters for the Cisco 7000.	**clear counters** [*type slot/port*]

The command clears all the current interface counters from the interface unless the optional arguments (for the Cisco 7000, *type*, *slot*, and *port)* are specified to clear only a specific interface type from a specific slot and port 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*][1]
Clear the entire Token Ring RIF cache.	**clear rif-cache**

1. This command is documented in the "User Interface Commands" chapter of the *Router Products Command Reference* publication.

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, shutting down an interface causes the DTR signal to be dropped. On Token Ring interfaces, shutting down an interface causes the interface to deinsert from the ring. On FDDIs, shutting down an interface causes the optical bypass switch, if present, to go into bypass mode.

To shut down an interface and then restart it, 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 "Examples of Interface Shutdown" at the end of this chapter.

One reason to shut down an interface is if you want to change the electrical interface type or mode of a Cisco 7000 port online. You replace the serial adapter cable and use software commands to restart the interface, and if necessary, reconfigure the port for the new interface. At system startup or restart, the FSIP polls the interfaces and determines the electrical interface type of each port (according to the type of port adapter cable attached). However, it does not necessarily repoll an interface when you change the adapter cable online. To ensure that the system recognizes the new interface type, shut down using the **shutdown** command, and reenable the interface after changing the cable. Refer to your hardware documentation for more details.

Monitor and Maintain a Hub

You can perform the tasks in the following sections to monitor and maintain the hub:

- Shut Down the Hub Port
- Reset the Hub or Clear the Hub Counters
- Monitor the Hub

Shut Down the Hub Port

To shut down a hub port, perform the following tasks beginning in global configuration mode:

Task	Command
Enter hub configuration mode.	**hub ether** *number port* [*port*]
Shut down the hub port.	**shutdown**

See the examples of shutting down a hub port at the end of this chapter.

Reset the Hub or Clear the Hub Counters

To reset the hub or clear the hub counters, perform one of the following tasks in EXEC mode:

Task	Command
Reset and reinitialize the hub hardware.	**clear hub ether** *number*
Clear the hub counters displayed by the **show hub** command.	**clear hub counters** [**ether** *number* [*port* [*port*]]]

Monitor the Hub

To display hub information, perform the following task in EXEC mode:

Task	Command
Display hub statistics.	**show hub** [**ether** *number* [*port* [*port*]]]

Run Interface Loopback Diagnostics

You can use a loopback test on lines to detect and distinguish equipment malfunctions between line and modem or Channel Service Unit/Digital Service Unit (CSU/DSU) 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 router 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 router. The **ping** command can also be useful during loopback operation.

The loopback tests are available on the following interfaces:

- High-Speed Serial Interface (HSSI), including the High-Speed Communications Interface (HSCI) card ribbon cable
- 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
- Channelized T1 interfaces
- The FDDI (CSC-FCI) card
- 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.

Enable Loopback Testing on the HSSI

The HSSI allows you to do the following:

- Enable Loopback Test of the HSSI Applique
- Enable Loopback Test to the DTE
- Enable Loopback Test through the CSU/DSU
- Enable Loopback Test over Remote DS-3 Link
- Enable HSSI Externally Requested Loopback
- Perform HSCI Card Ribbon Cable Loopback Test

These tests apply only when the device supports them and are used to check the data communications channels. The tests are usually performed at the line port rather than the DTE port of the remote CSU/DSU.

The internal loopback concepts are illustrated in Figure 6-5.

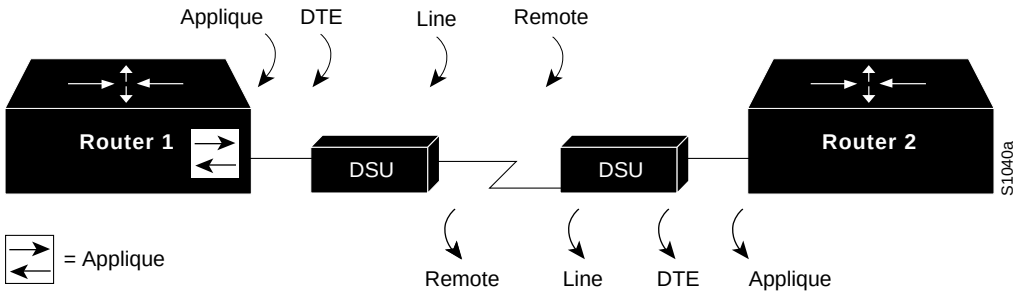


Figure 6-5 HSSI Loopback Testing

Enable Loopback Test of the HSSI Applique

You can configure an internal loop on the HSSI applique by performing the following task in interface configuration mode:

Task	Command
Loop internally on the HSSI applique.	**loopback applique**

Once enabled, the **loopback applique** command loops the packets on the applique, thereby establishing a loopback inside the router. This command is useful for sending pings to yourself to check the functionality of the applique. The HSSI applique (HSA card) uses an internal 44.736-MHz crystal clock during the applique loopback to drive its internal circuits. Refer to your hardware installation and maintenance publication for more information.

This command is functionally equivalent to entering the **loopback** command with no arguments; however, when the HSCI card is installed, the configuration displayed after the **write terminal** command is entered will show loopback applique set.

Enable Loopback Test to the DTE

You can loop packets to DTE within the CSU/DSU at the DTE interface, when the device supports this function. Doing so is useful for testing the DTE-to-DCE cable. To loop the packets to DTE, perform the following task in interface configuration mode:

Task	Command
Loop packets to DTE internally.	**loopback dte**

Enable Loopback Test through the CSU/DSU

You can loop packets completely through the CSU/DSU to configure a CSU loop, when the device supports this feature. Doing so is useful for testing the DCE device (CSU/DSU) itself. To configure a CSU loop, perform the following task in interface configuration mode:

Task	Command
Loop packets completely through the CSU/DSU.	**loopback line**

Enable Loopback Test over Remote DS-3 Link

You can loop packets through the CSU/DSU, over the Digital signal level 3 (DS-3) link, and to the remote CSU/DSU and back. To do so, perform the following task in interface configuration mode:

Task	Command
Loop packets through the CSU/DSU to a remote CSU/DSU over the DS-3 link.	**loopback remote**

This command applies only when the device supports the remote function. It is used for testing the data communication channels. The loopback usually is performed at the line port, rather than the DTE port, of the remote CSU/DSU.

Enable HSSI Externally Requested Loopback

The HSA applique on the HSSI contains an LED that indicates the LA, LB, and LC signals transiting through the devices. The CSU/DSU uses the LC signal to request a loopback from the router. The CSU/DSU might want to do this so that its own network management diagnostics can independently check the integrity of the connection between the CSU/DSU and the router.

When the CSU/DSU asserts the LC signal and the router enables the external loopback, the connection is blocked by the loopback and the router no longer has access to the data communication channel.

Figure 6-6 illustrates the extent of the signal during an external loopback request.

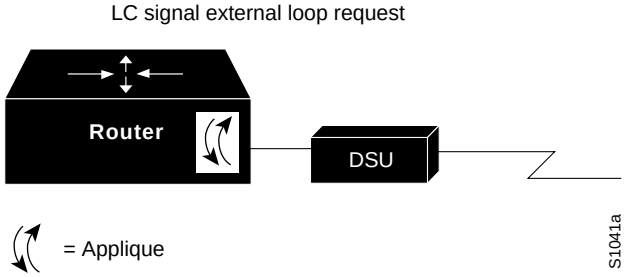


Figure 6-6 HSSI External Loopback Request

By default, this feature is disabled on the router. To enable this feature to support those CSU/DSUs that support this function, perform the following task in interface configuration mode:

Task	Command
Enable a two-way internal and external loopback request on HSSI from DSU/CSU.	**hssi external-loop-request**

If your CSU/DSU does not support this feature, it should not be enabled in the router. This prevents spurious line noise from accidentally tripping the external loopback request line, which would interrupt the normal data flow.

Perform HSCI Card Ribbon Cable Loopback Test

A useful diagnostic is available that allows fault isolation of possible defects on the HSCI card. This diagnostic is not part of the normal system diagnostics, but is offered to help technicians test for controller defects at installation or when the system is upgraded. The diagnostic involves recabling the HSCI card and then entering a diagnostic script. The tasks required to perform this diagnostic are described in the hardware installation and maintenance publication for your router.

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 router. To enable loopback mode on them, perform the following 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 router software provides an Ethernet loopback server that supports Digital Equipment Corporation (Digital), Intel, and Xerox systems specified by the "blue book," a joint specification written by 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 DEC systems that do not support the IP **ping** command, such as DECnet-only VMS systems.

To originate a loop test on your VMS system with a Cisco server, use the 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 the Channelized T1 Interface

To place the channelized T1 interface into the local or remote loopback mode, perform one of the following tasks in controller configuration mode:.

Task	Command
Enable a local loopback in both directions at the CxCT1 port.	**loopback local**
Enable a remote loopback between the router and the far-side CSU.	**loopback remote**

Enable Loopback on the CSC-FCI FDDI Card

You can place the FDDI (CSC-FCI) into loopback mode by performing the following task in interface configuration mode:

Task	Command
Enable loopback to verify that the FDDI (CSC-FCI) interface receives back every packet it sends.	**loopback**

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 help you understand some aspects of interface configuration. More complex and realistic examples appear in the chapters that describe special interface configuration and routing and bridging configuration.

- Examples of Enabling Interface Configuration
- Example of Enabling Ethernet Encapsulation
- Example of a Dedicated Asynchronous Interface
- Example of Restricting Access on the Asynchronous Interface
- Example of Asynchronous Routing and Dynamic Addressing
- Example of a PPP Connection
- Examples of SLIP Connections
- Examples of Interface Descriptions
- Examples of Interface Shutdown
- Examples of IP Tunneling
- Example of CHAP with an Encrypted Password
- Examples of Enabling a Hub
- Examples of Configuring a Source Address for an Ethernet Hub Port
- Examples of Shutting Down a Hub Port
- Examples of Dial Backup Service When the Primary Line Goes Down
- Examples of Dial Backup Service When the Primary Line Reaches Threshold
- Examples of Dial Backup Service When the Primary Line Exceeds Threshold
- Examples of Channelized T1 Controller and Interface

Examples 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
```

The same example on a Cisco 7000 requires the following commands:

```
interface serial 1/0
encapsulation ppp
```

Example of Enabling Ethernet Encapsulation

These commands enable standard Ethernet Version 2.0 encapsulation on the Ethernet interface processor in slot 4 on port 2 of a Cisco 7000:

```
interface ethernet 4/2
encapsulation arpa
```

Example of a Dedicated Asynchronous Interface

The following example assigns an IP address to an asynchronous interface and places the line in dedicated network mode:

```
interface async 1
async default ip address 182.32.7.51
async mode dedicated
```

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
!
line 1
speed 19200
flow hardware
modem inout
interface async 1
async mode interactive
async dynamic address
ip access-group 1 out
ip access-group 2 in
```

Example of Asynchronous Routing and Dynamic Addressing

The following example shows a simple configuration that allows routing and dynamic addressing. In this configuration, the router will act as either a telecommuting server or a router, depending on whether the user specifies **/routing** in the EXEC **slip** or **ppp** command.

```
interface async 1
async dynamic routing
async dynamic address
async mode interactive
```

Example of a PPP Connection

In the following example, a line that is in asynchronous mode is using PPP encapsulation (see Figure 6-7). The IP address of the PC is ntpc (assuming that the name ntpc is in the DNS so that it can be resolved to a real IP address). The person typing this command is on a PC running a terminal emulator program.

```
Router> ppp ntpc@server1 /routing /compressed
```

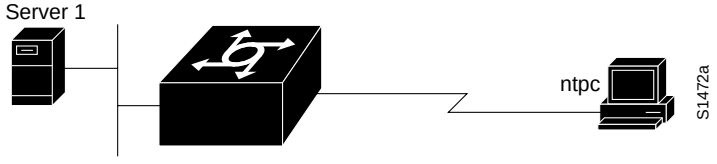


Figure 6-7 Using the PPP EXEC Command

Examples of SLIP Connections

The following example shows how to make a connection when a permanent address has been assigned and dynamic addressing is not allowed. An authentication request is sent to the TACACS server, and if it is approved, the line is placed in SLIP mode and the IP address is displayed.

```
Router> slip
Entering SLIP mode.
Your IP address is 192.31.7.28, MTU is 1500 bytes
```

The following example illustrates the prompts displayed and the response required when dynamic addressing is used to assign the SLIP address:

```
Router> slip
IP address or hostname? 192.31.6.15
Password:
Entering SLIP mode
Your IP address is 192.31.6.15, MTU is 1500 bytes
```

The following example illustrates the implementation of header compression on the interface with the IP address 128.66.2.1:

```
Router> slip /compressed 128.66.2.1
Password:
Entering SLIP mode.
Interface IP address is 128.66.2.1, MTU is 1500 bytes.
Header compression is On.
```

In the following example, header compression is configured as passive, so the status of header compression will be assigned by the user-level **slip** or **ppp** command:

```
Router> slip 1.0.0.1@check
Password:
Entering SLIP mode.
Interface IP address is 1.0.0.1, MTU is 1500 bytes
Header compression will match your system.
```

Examples of Interface Descriptions

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
```

The following example for a Cisco 7000 describes an administration network attached to the Ethernet processor in slot 2, port 4:

```
interface ethernet 2/4
description 2nd floor administration net
```

Examples of Interface Shutdown

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
```

The following example shuts down a T1 circuit number 23 running on a Cisco 7000:

```
interface serial 4/0:23
shutdown
```

The following next example shuts down the entire T1 line physically connected to a Cisco 7000:

```
controller t1 4/0
shutdown
```

Examples of IP Tunneling

The following example shows an IP tunneling configuration with commented (!) explanations:

```
!Creates the interface
interface tunnel 0
!enables IPX on the interface
novell network 1e
!enables appletalk
appletalk cable-range 4001-4001 128
!enables IP
ip address 10.1.2.3. 255.255.255.0
!enables DECnet
DECnet cost 4
!sets the source address, or interface, for packets
tunnel source ethernet 0
!determines where the encapsulated packets are to go
tunnel destination 131.108.14.12
!sets the encapsulator protocol
tunnel mode gre
!computes a checksum on passenger packets if protocol doesn’t already have reliable
!checksum
tunnel checksum needed
!sets the id key
tunnel key 42
!set to drop out of order packets
tunnel sequence-datagrams
```

Example of Routing Two AppleTalk Networks across an IP-Only Backbone

Figure 6-8 is an example of connecting multiprotocol subnetworks across a single-protocol backbone. The configurations of Router A and Router B follow.

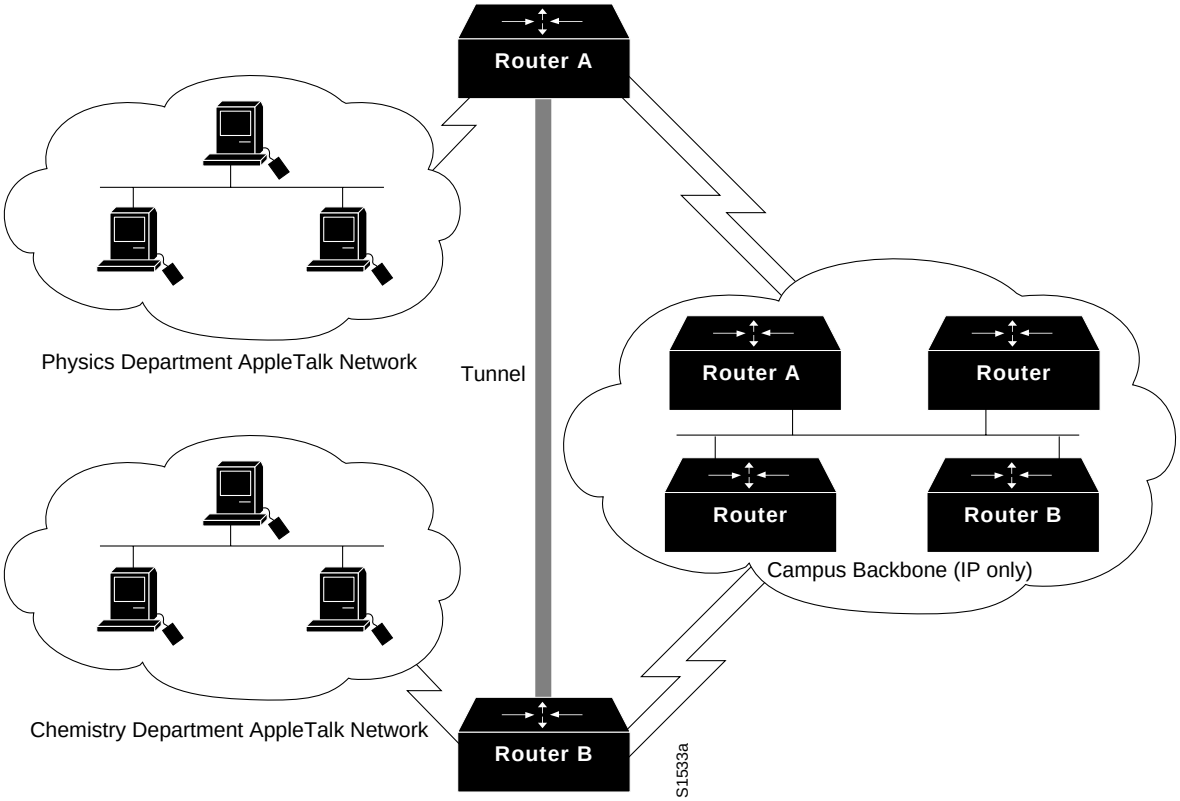


Figure 6-8 Connecting Multiprotocol Subnetworks across a Single-Protocol Backbone

Router A

```
interface ethernet 0
description physics department AppleTalk lan
AppleTalk cable-range 4001-4001 32
!
interface fddi 0
description connection to campus backbone
ip address 36.0.8.108 255.255.255.0
interface tunnel 0
tunnel source fddi 0
tunnel destination 36.0.21.20
appletalk cable-range 5313-5313 1
```

Router B

```
interface ethernet 0
description chemistry department appletalk lan
AppleTalk cable-range 9458-9458 3
!
interface fddi 0
description connection to campus backbone
ip address 36.0.21.20 255.255.255.0
interface tunnel 0
tunnel source fddi 0
tunnel destination 36.0.8.108
appletalk cable-range 5313-5313 2
```

Example of Routing a Private IP Network and a Novell Net across a Public Service Provider

Figure 6-9 is an example of routing a private IP network and a Novell network across a public service provider.

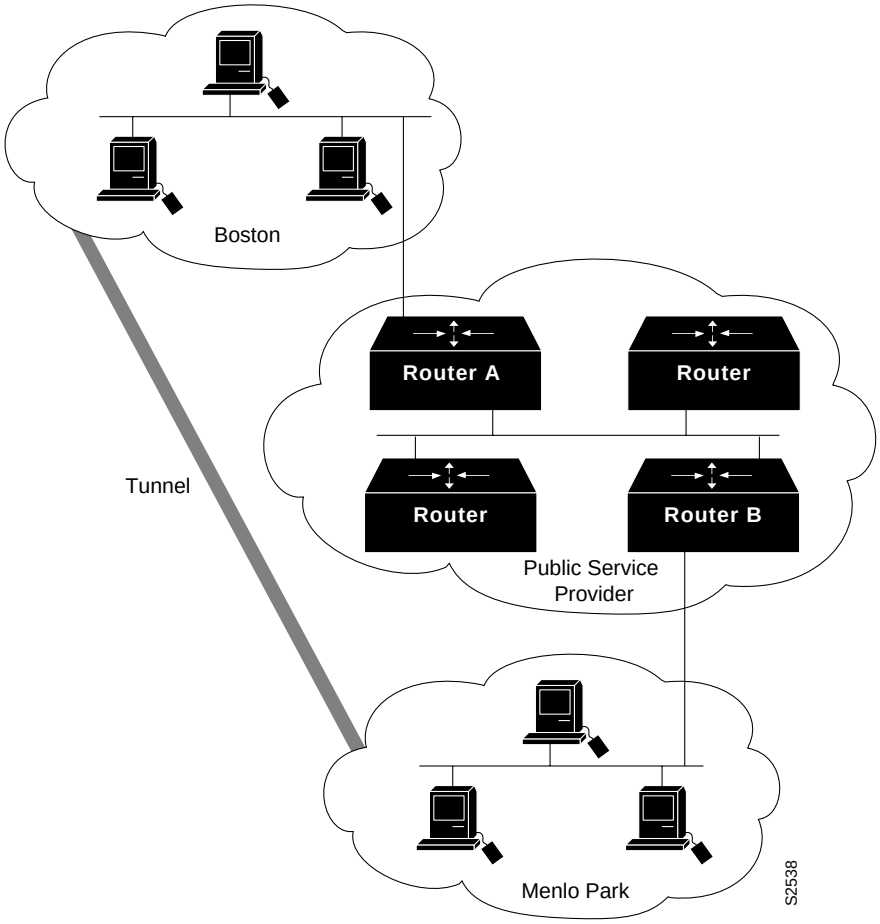


Figure 6-9 Creating Virtual Private Networks across WANs

Router A

```
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
```

Router B

```
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
```

Example of CHAP with an Encrypted Password

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

Configuration of Router yyy

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

Configuration of Router xxx

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

Configuration of Router 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
```

Examples of Enabling a Hub

The following example configures port 1 on hub 0 of Ethernet interface 0:

```
hub ether 0 1
no shutdown
```

The following example configures ports 1 through 8 on hub 0 of Ethernet interface 0:

```
hub ether 0 1 8
no shutdown
```

Examples of Configuring a Source Address for an Ethernet Hub Port

The following example configures the hub to allow only packets from MAC address 1111.2222.3333 on port 2 of hub 0:

```
hub e 0 2
source-address 1111.2222.3333
```

The following example configures the hub to remember the first MAC address received on port 2, and allow only packets from that learned MAC address:

```
hub e 0 2
source-address
```

Examples of Shutting Down a Hub Port

The following example shuts down ports 3 through 5 on hub 0:

```
hub e 0 3 5
shutdown
```

The following example shuts down port 3 on hub 0:

```
hub e 0 3
shutdown
```

Examples 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 because of load on 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.

The same example on the Cisco 7000 would be as follows:

```
interface serial 1/1
backup interface serial 2/2
backup delay 30 60
```

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.

The same example on the Cisco 7000 would be as follows:

```
interface serial 1/1
backup interface serial 2/2
backup load 75 5
```

Examples of Dial Backup Service When the Primary Line Exceeds Threshold

The following 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.

The same example on the Cisco 7000 is as follows:

```
interface serial 1/1
backup interface serial 2/2
backup load 25 5
backup delay 10 60
```

Examples of Channelized T1 Controller and Interface

This example applies only to a Cisco 7000 series. It configures the router to acknowledge a T1 line and its circuits. Four different circuits are defined for the second CxCT1 attached to the MIP in slot 4.

```
controller t1 4/1
framing esf
linecode b8zs
channel-group 0 timeslots 1
channel-group 8 timeslots 5,7,12-15, 20 speed 64
channel-group 12 timeslots 2
channel-group 23 timeslots 24
```

The following example configures circuit 0 for Point-to-Point (PPP) encapsulation:

```
interface serial 4/1:0
ip address 131.108.13.1 255.255.255.0
encapsulation ppp
```

The following example configures circuit 8 for IP routing and disables IP route cache:

```
interface serial 4/1:8
ip address 131.108.1.1 255.255.255.0
no ip routecache
```

The following example configures circuit 12 for Frame Relay encapsulation and subinterface support:

```
interface serial 4/1:12
encapsulation frame-relay
!
interface serial 4/1:12.1
ip address 1.1.1.1 255.0.0.0
!
interface serial 4/1:12.2
ip address 2.2.2.2 255.0.0.0
```

The following example configures circuit 23 for IP routing and enables autonomous switching:

```
interface serial 4/1:23
ip address 3.3.3.3 255.0.0.0
ip routecache cbus
```