Configuring Logical Interfaces

Use the information in this chapter to understand and configure the types of logical, or virtual, interfaces supported on Cisco routers and access servers. This chapter includes the following sections:

For examples of configuration tasks, see the "Logical Interface Configuration Examples" section.

For hardware technical descriptions and information about installing interfaces, refer to the hardware installation and configuration publication for your product. For complete descriptions of the logical interface commands, refer to the "Interface Commands" chapter of the *Cisco IOS Interface Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

To identify the hardware platform or software image information associated with a feature, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release. For more information, see the "Identifying Supported Platforms" in "Using Cisco IOS Software."

Configuring a Loopback Interface

You can specify a software-only interface called a loopback interface to emulate an interface. Loopback interfaces are supported on all platforms. A loopback interface is a virtual interface that is always up and allows Border Gateway Protocol (BGP) and remote source-route bridging (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 to establish a Telnet session from the device's console to its auxiliary port when all other interfaces are down. You can also use a loopback interface to configure IPX-PPP on asynchronous interfaces. To do so, you must associate an asynchronous interface with a loopback interface configured to run IPX. In applications in which other routers or access servers 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 or access server 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 serves as the Null 0 interface also.

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

To specify a loopback interface and enter interface configuration mode, use one of the following commands in global configuration mode:

Command	Purpose
`Router(config)# interface loopback number`	Enters interface configuration.
`Router(config)# interface loopback slot/port`	Enters interface configuration for Cisco 7200 series or Cisco 7500 series routers.
`Router(config)# interface loopback slot/port-adapter/port`	Enters interface configuration for Cisco 7500 series routers.

For more general information about loopback interfaces, see the "Running Interface Loopback Diagnostics" section in the "Features for Any Interface" chapter.

Troubleshooting Channelized E1 and Channelized T1

When troubleshooting channelized T1 or E1, you must first determine if the problem is with a particular channel group or with the T1 or E1 line.

If the problem is with a single channel group, you have a potential interface problem.

If the problem is with the T1 or E1 line, or with all channel groups, you have a potential controller problem.

The following sections describe how to determine whether the problem affects an interface or a controller:

- Running Controller Loopback Diagnostic Tests
- Channelized E1 Controller Loopback

When you troubleshoot E1 or T1 controllers, first check that the configuration is correct. The framing type and line code should match what the service provider has specified. Then check channel group and PRI-group configurations, especially to verify that the time slots and speeds are what the service provider has specified.

At this point, the **show controllers t1** or **show controllers e1** commands should be used to check for T1 or E1 errors. Use the command several times to determine if error counters are increasing, or if the line status is continually changing. If these errors are occurring, you need to work with the service provider.

Note Cisco routers do not have CSU capability and do not react to any remote loopback codes at the T1 or E1 level.

Running Controller Loopback Diagnostic Tests

Controller loopback tests are a means to isolate problems and are available for both channelized T1 controllers and channelized E1 controllers. The following loopback tests are documented for isolating T1 and E1 controller issues:

- Local Loopback
- Remote Loopback
- Channelized E1 Controller Loopback

Local Loopback

The local loopback loops the controller both toward the router and toward the line. Because the loopback is done internally to the router, the controller should make the transition to the UP state within approximately 10 seconds, and no further T1 errors should be detected.

All channel groups will be looped back; if the encapsulation on that channel group supports loopbacks (for example, HDLC and PPP), you can test that channel group by pinging the interface address. For example, if you have assigned an IP address to the serial interface defined for a channel group, you can ping that IP address.

To place the controller into local loopback, use the following command in controller configuration mode:

Command	Purpose
`Router(config-controller)# loopback local` *controller*	Loops the T1 controller toward the router and toward the line.

To test a channel group, use the following command in EXEC mode:

Command	Purpose
`Router# ping` *protocol protocol-address*	Pings the interface address.

To check errors, use the following command in EXEC mode:

Command	Purpose
`Router> show controllers t1`	Checks errors.

If any errors occur, or the controller fails to change to the up state, contact the Cisco Technical Assistance Center (TAC).

Because the controller local loopback is bidirectional, the service provider can test the line integrity using a T1 bit error rate tester (BERT) test set.

Remote Loopback

The second T1 controller loopback is a remote loopback. This loopback can be used only if the *entire* T1 goes to a remote CSU. This is not the case with 99.9 percent of channelized T1. When the **loopback remote controller** command is executed, an in-band CSU loop-up code will be sent over the entire T1, which will attempt to loop up the remote CSU. To place the controller in remote loopback, use the following command in controller configuration mode:

Command	Purpose
`Router(config-controller)# loopback remote controller`	Places the T1 controller in remote loopback.

Note If controller loopbacks are used, they will disrupt service for all channel groups on that interface.

Channelized E1 Controller Loopback

For the E1 controller, only the local loopback is available. Local loopback operates the same as the local loopback on the T1 controller, forming a bidirectional loopback, both toward the router and toward the line. To place the E1 controller in local loopback, use the following command in controller configuration mode:

Command	Purpose
`Router(config-controller)# loopback controller`	Places the E1 controller in local loopback toward the router and toward the line.

All channel groups will be looped back; if the encapsulation on that channel group supports loopbacks (for example, HDLC and PPP), you can test that channel group by pinging the interface address. For example, if you have assigned an IP address to the serial interface defined for a channel group, you can ping that IP address.

To place the controller into local loopback, use the following command in controller configuration mode:

Command	Purpose
`Router(config-controller)# loopback local controller`	Loops the T1 controller toward the router and toward the line.

To test a channel group, use the following command in EXEC mode:

Command	Purpose
`Router> ping protocol protocol-address`	Pings the interface address.

To check errors, if any, use the following command in EXEC mode:

Command	Purpose
`Router> show controllers t1`	Checks errors.

If any errors occur, they are most likely a hardware problem; contact the Cisco TAC. In addition, you can ask the service provider to test the line by using a T1 BERT test set.

Channelized E1 Controller Example

The following example configures a Cisco 7500 series router to acknowledge an E1 line:

```
controller e1 3/0
 channel-group 0 timeslots 1
 channel-group 8 timeslots 5-15, 20-30
 channel-group 12 timeslots 2
 channel-group 29 timeslots 31
```

Channelized T1 Controller Examples

The following example applies only to a Cisco 7500 series router. It configures the router to acknowledge a T1 line and its circuits. Four different circuits (and their corresponding serial interfaces) 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 PPP encapsulation:

```
interface serial 4/1:0
 ip address 172.18.13.1 255.255.255.0
 encapsulation ppp
```

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

```
interface serial 4/1:8
 ip address 172.18.1.1 255.255.255.0
 no ip route-cache
```

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 10.1.1.1 255.0.0.0
!
interface serial 4/1:12.2
 ip address 10.2.2.2 255.0.0.0
```

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

```
interface serial 4/1:23
 ip address 10.3.3.3 255.0.0.0
 ip route-cache cbus
```

Configuring a Null Interface

The Cisco IOS software 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 unreachables**.

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

To specify the null interface, use the following command in global configuration mode:

Command	Purpose
`Router(config)# interface null 0`	Enters interface configuration.

Specify null 0 (or null0) as the interface type and number. 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
```

Configuring a Tunnel Interface

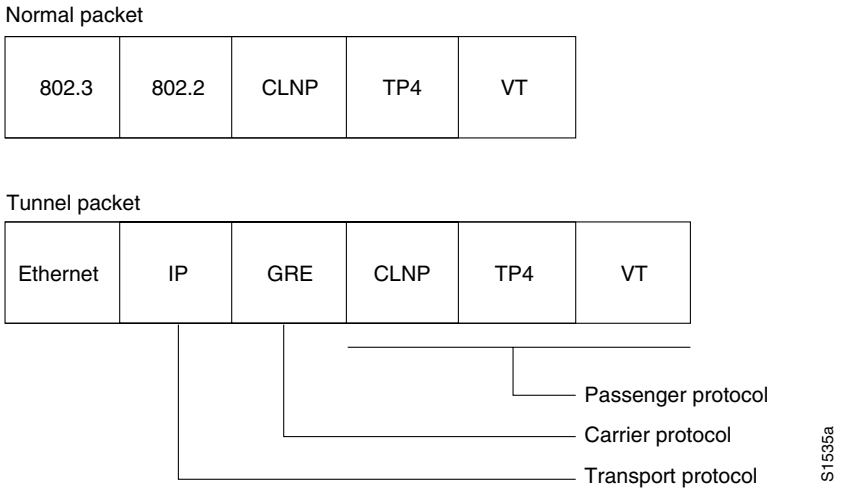
Tunneling provides a way to encapsulate arbitrary packets inside a transport protocol. This feature is implemented as a virtual interface to provide a simple interface for configuration. The tunnel interface is not tied to specific "passenger" or "transport" protocols, but rather, it is an architecture 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 the following three primary components:

- Passenger protocol, which is the protocol that you are encapsulating (AppleTalk, Banyan VINES, CLNS, DECnet, IP, or IPX)
- Carrier protocol, which is one of the following encapsulation protocols:
  - Generic route 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
  - Distance Vector Multicast Routing Protocol (DVMRP) (IP in IP tunnels)
- Transport protocol, which is the protocol used to carry the encapsulated protocol (IP only)

Figure 22 illustrates IP tunneling terminology and concepts.

Figure 22 IP Tunneling Terminology and Concepts



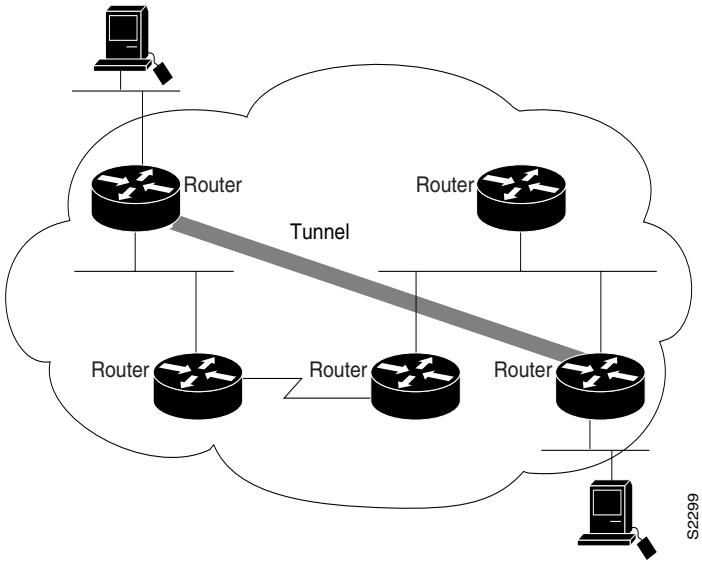
To understand the process of tunneling, consider connecting two AppleTalk networks with a non-AppleTalk backbone, such as IP. The relatively high bandwidth consumed by the broadcasting of Routing Table Maintenance Protocol (RTMP) data packets can severely hamper the backbone's network performance. This 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 removes the encapsulation from 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

The following are several situations in which 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 23).
- To connect discontinuous subnetworks.
- To allow virtual private networks across WANs.

Figure 23 Providing Workarounds for Networks with Limited Hop Counts



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

Special Considerations for Configuring Tunnel Interfaces

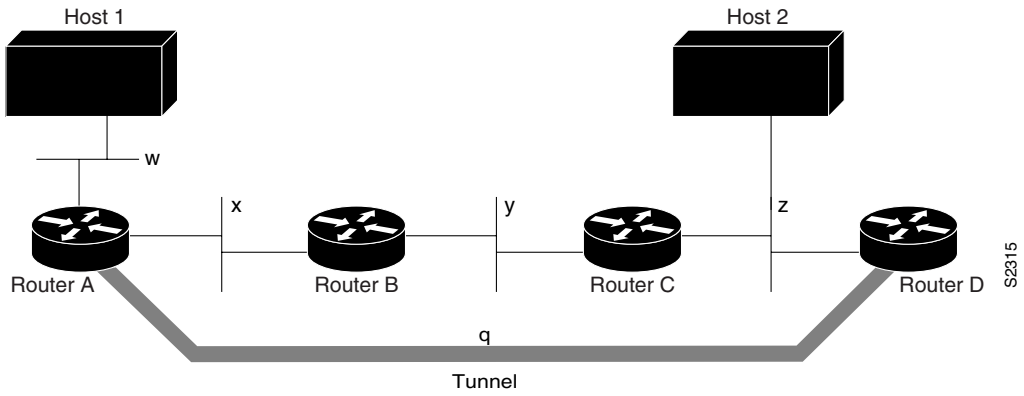
The following are considerations and precautions to observe when you configure tunneling:

- Encapsulation and the removal of encapsulation at the tunnel end points are slow operations; in general, only processor switching is supported. However, fast switching of GRE tunnels was introduced in Cisco IOS Release 11.1 for the Cisco 2500 series and the Cisco 4000 series of routers.
- Consider security and topology issues. Be careful not to violate access control lists. You can configure a tunnel with a source and destination that are not restricted by firewall routers.
- Tunneling might create problems with transport protocols that have limited timers (for example, DECnet) because of 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 function 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 24, 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.

- An even worse problem will occur if routing information from the tunneled network mixes with the information about the transport network. 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 shut down temporarily. 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
```

Figure 24 Tunnel Precautions: Hop Counts



IP Tunneling Configuration Task List

To configure IP tunneling, perform the following tasks. Each task in the list is identified as either required or optional.

- Specifying the Tunnel Interface (Required)
- Configuring the Tunnel Source (Required)
- Configuring the Tunnel Destination (Required)
- Configuring the Tunnel Mode (Optional)
- Configuring End-to-End Checksumming (Optional)
- Configuring a Tunnel Identification Key (Optional)
- Configuring a Tunnel Interface to Drop Out-of-Order Datagrams (Optional)
- Configuring Asynchronous Host Mobility (Optional)

For commands that monitor IP tunnels, see the "Monitoring and Maintaining the Interface" section in the "Features for Any Interface" chapter. For examples of configuring tunnels, see the "IP Tunneling: Example" section.

Specifying the Tunnel Interface

To specify a tunnel interface and enter interface configuration mode, use one of the following commands in global configuration mode:

Command	Purpose
`Router(config)# interface tunnel number`	Enters interface configuration.
`Router(config)# interface tunnel slot/port`	Enters interface configuration for Cisco 7200 series routers.
`Router(config)# interface tunnel slot/port-adapter/port`	Enters interface configuration for Cisco 7500 series routers.

Configuring the Tunnel Source

To specify the source address for the tunnel interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# tunnel source {ip-address | type number}`	Configures the tunnel source.

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

Configuring the Tunnel Destination

To specify the destination for the tunnel interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# tunnel destination {hostname | ip-address}`	Configures the tunnel destination.

Configuring the Tunnel Mode

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

Command	Purpose
`Router(config-if)# tunnel mode {aurp | cayman | dvmrp | eon | gre ip | nos}`	Configures the tunnel mode.

If you are tunneling AppleTalk, you must use the AppleTalk Update Routing Protocol (AURP), Cayman, or GRE tunneling mode. Cayman tunneling is designed by Cayman Systems and enables routers and access servers to interoperate with Cayman GatorBoxes. You can have Cisco devices at either end of the tunnel, or you can have a GatorBox at one end and a Cisco router or access server at the other end. Use Distance Vector Multicast Routing Protocol (DVMRP) mode when a router or access server connects to a mrouted router to run DVMRP over a tunnel. You must configure Protocol-Independent Multicast (PIM) and an IP address on a DVMRP tunnel.

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

Note For an example configuration of a GRE tunnel, see the Configuring GRE/IPv4 Tunnels: Examples in the Logical Interface Configuration Examples section.

If you use GRE, you must have only Cisco routers or access servers at both ends of the tunnel connection. When you use GRE to tunnel AppleTalk, you must configure an AppleTalk network address and a zone. To tunnel AppleTalk using GRE, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface tunnel number`	Enables tunneling on the interface.
Step 2	`Router(config-if)# appletalk cable-range start-end [network.node]`	Assigns a cable range to an interface.
Step 3	`Router(config-if)# appletalk zone zone-name`	Sets a zone name for the connected AppleTalk network.
Step 4	`Router(config-if)# tunnel source {ip-address | type number}`	Specifies the interface from which the encapsulated packets will be sent, or specifies the IP address of the router.
Step 5	`Router(config-if)# tunnel destination {hostname | ip-address}`	Specifies the IP address of the router at the far end of the tunnel.
Step 6	`Router(config-if)# tunnel mode gre ip`	Enables GRE tunneling.

Configuring 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 Cisco IOS software drops corrupted packets. To enable such checksums on a tunnel interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# tunnel checksum`	Configures end-to-end checksumming.

Configuring a Tunnel Identification Key

You can 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 incorrect configuration or injection of packets from a foreign source.

Note IP multicast traffic is not supported when a tunnel ID key is configured unless the traffic is process-switched. You must configure the **no ip mroute-cache** command in interface configuration mode on the interface if an ID key is configured. This note applies only to Cisco IOS Release 12.0 and earlier releases.

The tunnel ID key is available with GRE only.

Note When GRE is used, 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, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# tunnel key` *key-number*	Configures a tunnel identification key.

Configuring 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 function poorly when they receive packets out of order (for example, LLC2-based protocols). This option is available with GRE only. To use this option, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# tunnel sequence-datagrams`	Configures a tunnel interface to drop out-of-order datagrams.

Configuring Asynchronous Host Mobility

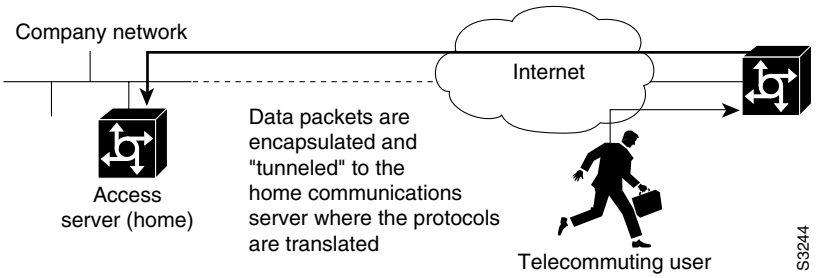
Increasingly, remote users are accessing networks through dial-up telephone connections. In contrast to local users who can connect directly into the network, remote users must first dial in to an access server.

The access server supports a packet tunneling strategy that extends the internetwork—in effect creating a virtual private link for the mobile user. When a user activates asynchronous host mobility, the access server on which the remote user dials into becomes a remote point-of-presence (POP) for the home network of the user. Once logged in, users experience a server environment identical to the one that they experience when they connect directly to the "home" access server.

Once the network layer connection is made, data packets are tunneled at the physical and/or data link layer instead of at the protocol layer. In this way, raw data bytes from dial-in users are transported directly to the "home" access server, which processes the protocols.

Figure 25 illustrates the implementation of asynchronous host mobility on an extended internetwork. A mobile user connects to an access server on the internetwork and, by activating asynchronous host mobility, is connected to a "home" access server configured with the appropriate username. The user sees an authentication dialog or prompt from the "home" system and can proceed as if connected directly to that device.

Figure 25 Asynchronous Host Mobility



The remote user implements asynchronous host mobility by executing the **tunnel** command in user EXEC mode. The **tunnel** command sets up a network layer connection to the specified destination. The access server accepts the connection, attaches it to a virtual terminal (VTY), and runs a command parser capable of running the normal dial-in services. After the connection is established, data is transferred between the modem and network connection with a minimum of interpretations. When communications are complete, the network connection can be closed and terminated from either end.

Refer to the *Cisco Access Connection Guide* for information about setting up the network layer connection with the **tunnel** command.

Configuring IP over a CLNS Tunnel

IP over a CLNS tunnel is a virtual interface that enhances interactions with CLNS (Connectionless Network Service) networks, allowing IP packets to be tunneled through the Connectionless Network Protocol (CLNP) to preserve TCP/IP services.

Configuring an IP over CLNS tunnel (CTunnel) allows you to Telnet to a remote router that has only CLNS connectivity. Other management facilities can also be used, such as Simple Network Management Protocol (SNMP) and TFTP, which otherwise would not be available over a CLNS network.

IP over a CLNS Tunnel is supported on all platforms that support ISO CLNS.

To configure IP over a CLNS Tunnel (CTunnel), use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface ctunnel` *interface-number*	Creates a virtual interface to transport IP over a CLNS tunnel and enters interface configuration mode. The interface number must be unique for each CTunnel interface.

	Command	Purpose
Step 2	`Router(config-if)# ctunnel destination remote-nsap-address`	Configures the destination parameter for the CTunnel. Specifies the destination NSAP[1] address of the CTunnel, where the IP packets are extracted.
Step 3	`Router(config-if)# ip address ip-address mask`	Sets a primary or secondary IP address for an interface.

1. NSAP = network service access point

Note To configure a CTunnel between a single pair of routers, you must enter the foregoing commands on each router. The destination NSAP address for Router A would be the NSAP address of Router B, and the destination NSAP address for Router B would be the NSAP address of Router A. Ideally, the IP addresses used for the virtual interfaces at either end of the tunnel should be in the same IP subnet.

Verifying IP over CLNS Tunnel Configuration

To verify correct configuration of the IP over a CLNS Tunnel feature, perform the following steps:

Step 1 On Router A, ping the IP address of the CTunnel interface of Router B.

Step 2 On Router B, ping the IP address of the CTunnel interface of Router A.

Monitoring and Maintaining IP over a CLNS Tunnel

To display the status of IP over CLNS tunnels, use the following command in privileged EXEC mode:

Command	Purpose
`Router# show interfaces ctunnel interface-number`	Displays information about an IP over CLNS tunnel.

Logical Interface Configuration Examples

This section includes the following examples to illustrate configuration tasks described in this chapter:

- IP Tunneling: Example
- Configuring GRE/IPv4 Tunnels: Examples

IP Tunneling: Example

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 protocol.
 tunnel mode gre
 ! Computes a checksum on passenger packets if protocol does not already have reliable
 ! checksum
 tunnel checksum needed
 ! Sets the ID key.
 tunnel key 42
 ! Sets dropping of out-of-order packets.
 tunnel sequence-datagrams
```

Configuring GRE/IPv4 Tunnels: Examples

The following example shows a simple configuration of GRE tunneling. Note that Ethernet interface 0/1 is the tunnel source for Router A and the tunnel destination for Router B. Fast Ethernet interface 0/1 is the tunnel source for Router B and the tunnel destination for Router A.

Router A

```
interface Tunnel0
 ip address 10.1.1.2 255.255.255.0
 tunnel source Ethernet0/1
 tunnel destination 192.168.3.2
 tunnel mode gre ip
!
interface Ethernet0/1
 ip address 192.168.4.2 255.255.255.0
```

Router B

```
interface Tunnel0
 ip address 10.1.1.1 255.255.255.0
 tunnel source FastEthernet0/1
```

```
 tunnel destination 192.168.4.2
 tunnel mode gre ip
!
interface FastEthernet0/1
 ip address 192.168.3.2 255.255.255.0
```

The following example configures a GRE tunnel running both IS-IS and IPv6 traffic between Router A and Router B.

Router A

```
ipv6 unicast-routing
clns routing
!
interface Tunnel0
 no ip address
 ipv6 address 2001:0DB8:1111:2222::1/64
 ipv6 router isis
 tunnel source Ethernet0/0
 tunnel destination 10.0.0.2
 tunnel mode gre ip
!
interface Ethernet0/0
 ip address 10.0.0.1 255.255.255.0
!
router isis
 network 49.0000.0000.000a.00
```

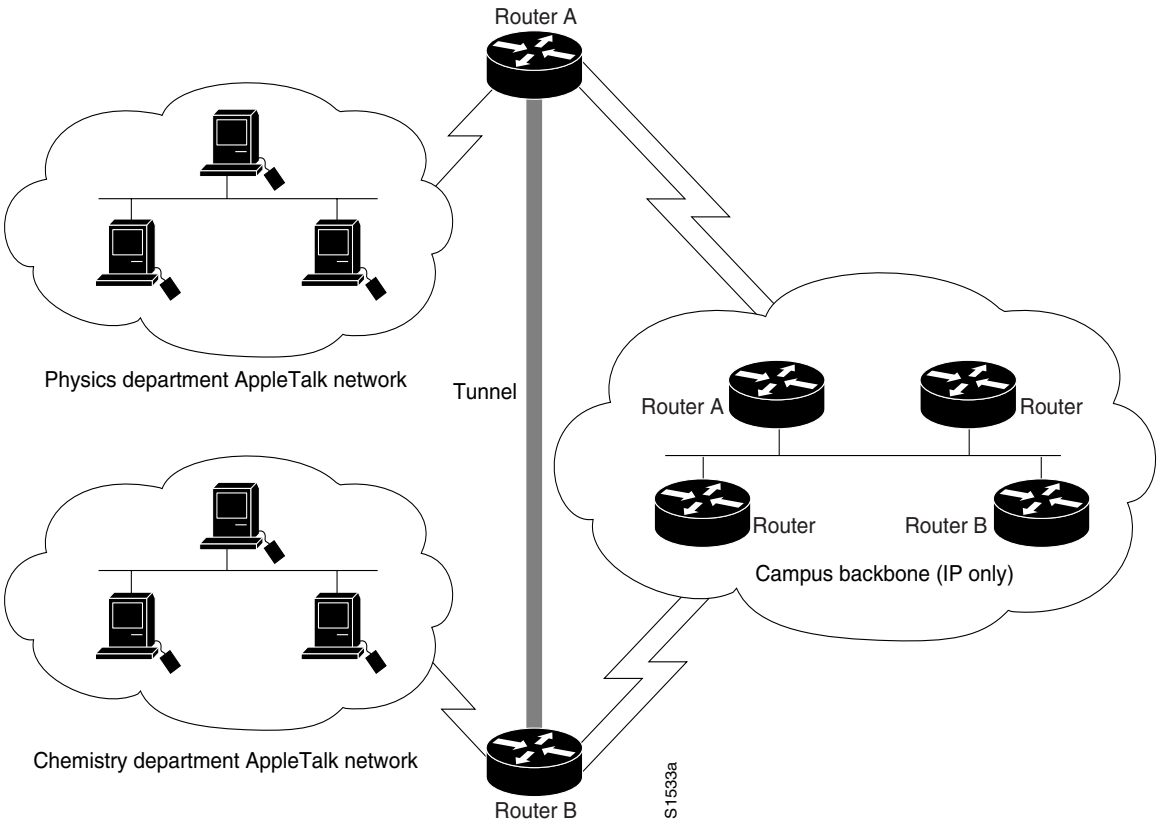
Router B

```
ipv6 unicast-routing
clns routing
!
interface Tunnel0
 no ip address
 ipv6 address 2001:0DB8:1111:2222::2/64
 ipv6 router isis
 tunnel source Ethernet0/0
 tunnel destination 10.0.0.1
 tunnel mode gre ip
!
interface Ethernet0/0
 ip address 10.0.0.2 255.255.255.0
!
router isis
 network 49.0000.0000.000b.00
 address-family ipv6
 redistribute static
 exit-address-family
```

Routing Two AppleTalk Networks across an IP-Only Backbone Example

Figure 26 is an example of connecting multiprotocol subnetworks across a single-protocol backbone. The configurations of Router A and Router B follow Figure 26.

Figure 26 Connecting AppleTalk Networks Across an IP-Only 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 131.0.8.108 255.255.255.0
interface tunnel 0
 tunnel source fddi 0
 tunnel destination 131.0.21.20
 appletalk cable-range 5313-5313 1
```

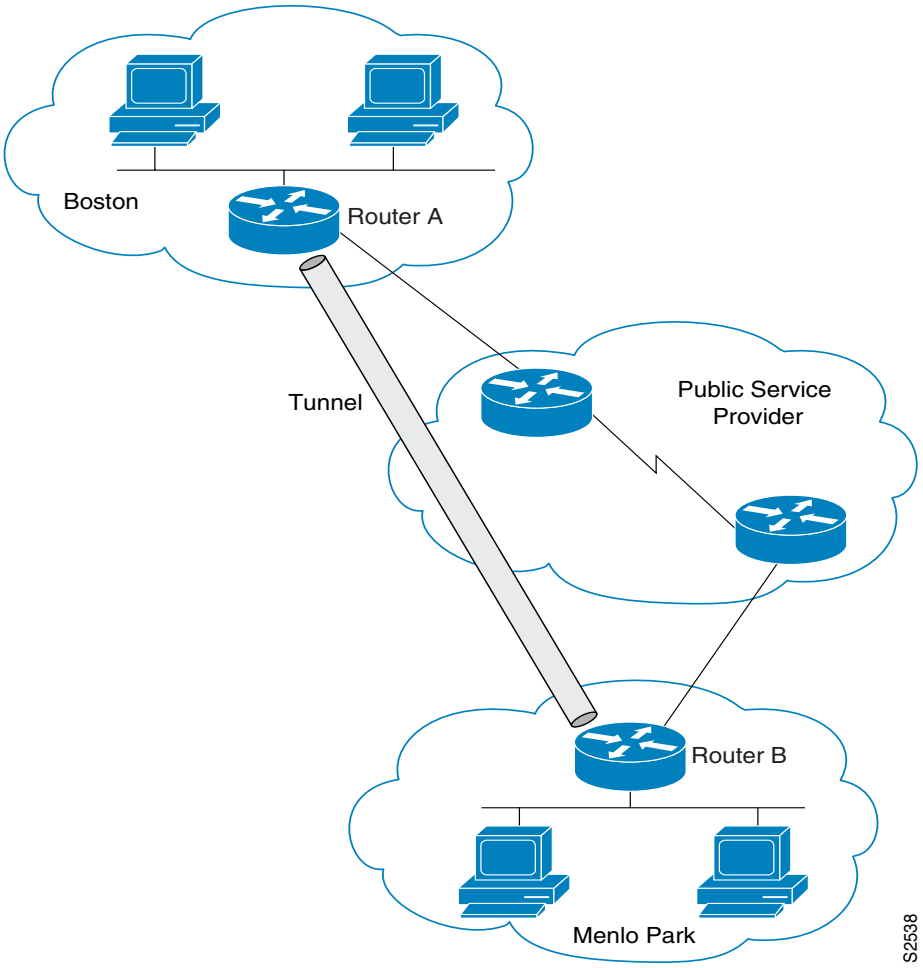
Router B

```
interface ethernet 0
 description chemistry department AppleTalk LAN
 appletalk cable-range 9458-9458 3
 !
interface fddi 0
 description connection to campus backbone
 ip address 131.0.21.20 255.255.255.0
interface tunnel 0
 tunnel source fddi 0
 tunnel destination 131.0.8.108
 appletalk cable-range 5313-5313 2
```

Routing a Private IP Network and a Novell Network Across a Public Service Provider Example

Figure 27 is an example of routing a private IP network and a Novell network across a public service provider. The configuration of Router A and Router B follow Figure 27.

Figure 27 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 131.13.2.1 255.255.255.0
!
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
```

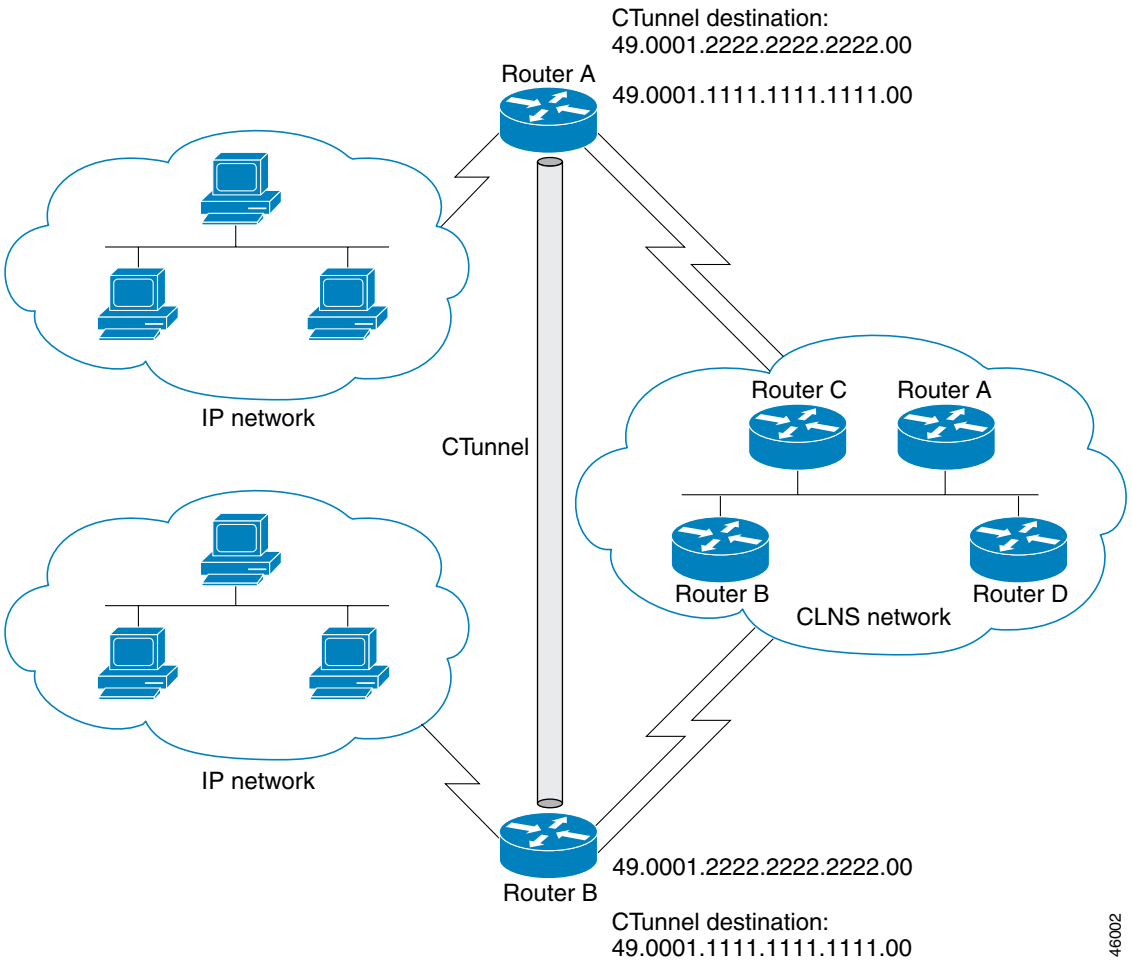
Router B

```
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 131.13.2.1
 ip address 10.1.2.2 255.255.255.0
 novell network 1f
```

Configuring IP over a CLNS Tunnel Example

Figure 28 illustrates the creation of a CTunnel between Router A and Router B, as accomplished in the configuration examples that follow for Router A and Router B:

Figure 28 Creation of a CTunnel



Router A

```
ip routing
clns routing

interface ctunnel 102
 ip address 10.0.0.1 255.255.255.0
 ctunnel destination 49.0001.2222.2222.2222.00

interface Ethernet0/1
 clns router isis

router isis
 net 49.0001.1111.1111.1111.00
```

```
router rip
 network 10.0.0.0
```

Router B

```
ip routing
clns routing

interface ctunnel 201
 ip address 10.0.0.2 255.255.255.0
 ctunnel destination 49.0001.1111.1111.1111.00

interface Ethernet0/1
 clns router isis

router isis
 net 49.0001.2222.2222.2222.00

router rip
 network 10.0.0.0
```