

# Configuring Frame Relay

---

This chapter describes the tasks for configuring Frame Relay on the router. For a complete description of the commands mentioned in this chapter, refer to the “Frame Relay Commands” chapter in the *Router Products Command Reference* publication. For historical background and a technical overview of Frame Relay, see the *Internetworking Technology Overview* publication.

Although Frame Relay access was originally restricted to leased lines, dial-up access is now supported. For more information, see the “Configure DDR over Frame Relay” section in the “Configuring DDR” chapter of this publication.

To install software on a new router by downloading software from a central server over an interface that supports Frame Relay, see the “Loading System Images, Microcode Images, and Configuration Files” chapter in this publication.

To configure access between SNA devices over a Frame Relay network, see the “Configuring SNA Frame Relay Access Support” chapter of this publication.

## Cisco’s Implementation of Frame Relay

Cisco’s Frame Relay implementation currently supports routing on IP, DECnet, AppleTalk, Xerox Network Service, Novell IPX, ISO CLNS, Banyan VINES, and transparent bridging.

The Frame Relay software provides the following capabilities:

- Support for the three generally implemented specifications of Frame Relay Local Management Interfaces (LMIs):
  - The *Frame Relay Interface* joint specification produced by Northern Telecom, Digital Equipment Corporation, StrataCom, and Cisco Systems
  - The ANSI-adopted Frame Relay signal specification, T1.617 Annex D
  - The International Telecommunication Union Telecommunication Standardization Sector (ITU-T)-adopted Frame Relay signal specification, Q.933 Annex A

---

**Note** The ITU-T carries out the functions of the former Consultative Committee for International Telegraph and Telephone (CCITT).

---

- Conformity to ITU-T I-series (ISDN) recommendation as I122, “Framework for Additional Packet Mode Bearer Services.”
  - The ANSI-adopted Frame Relay encapsulation specification, T1.618

- The ITU-T-adopted Frame Relay encapsulation specification, Q.922 Annex A
- Conformity to Internet Engineering Task Force (IETF) encapsulation in accordance with RFC 1490, except bridging.
- Support for a keepalive mechanism, a multicast group, and a status message, as follows:
  - The keepalive mechanism provides an exchange of information between the network server and the switch to verify that data is flowing.
  - The multicast mechanism provides the network server with a local data link connection identifier (DLCI) and a multicast DLCI. This feature is specific to our implementation of the Frame Relay joint specification.
  - The status mechanism provides an ongoing status report on the DLCIs known by the switch.
- Transmission of congestion information from Frame Relay to DECnet Phase IV and CLNS. This mechanism promotes Forward Explicit Congestion Notification (FECN) bits from the Frame Relay layer to upper-layer protocols after checking for the FECN bit on the incoming DLCI. Use this Frame Relay congestion information to adjust the sending rates of end hosts. FECN-bit promotion is enabled by default on any interface using Frame Relay encapsulation. No configuration is required.
- Support for Frame Relay Inverse Address Resolution Protocol (InARP) as described in RFC 1293 for the AppleTalk, Banyan VINES, DECnet, IP, and IPX protocols, as well as native Hello packets for DECnet, CLNP, and Banyan VINES. It allows a router running Frame Relay to discover the protocol address of a device associated with the virtual circuit.
- Support for Frame Relay switching, whereby packets are switched based on the DLCI (a Frame Relay equivalent of a MAC-level address). Routers are configured as a hybrid DTE switch or pure Frame Relay DCE access node in the Frame Relay network. Cisco's implementation of Frame Relay switching allows the following configurations:
  - Switching over an IP tunnel
  - Network to Network Interface (NNI) to other Frame Relay switches
  - Local serial-to-serial switching

Frame Relay switching is used when all traffic arriving on one DLCI can be sent out on another DLCI to the same next-hop address. In such cases, the router does not have to examine the frames individually to discover the destination address, and as a result, the processing load on the router decreases.

- Support for *subinterfaces* associated with a physical interface. The software groups one or more permanent virtual circuits under separate subinterfaces, which in turn are located under a single physical interface. See the “Understand and Define Frame Relay Subinterfaces” and the “Subinterface Examples” sections later in this chapter.
- Support of the Frame Relay DTE MIB specified in RFC 1315. However, the error table is not implemented. To use the Frame Relay MIB, refer to your MIB publications.

## Frame Relay Hardware Configurations

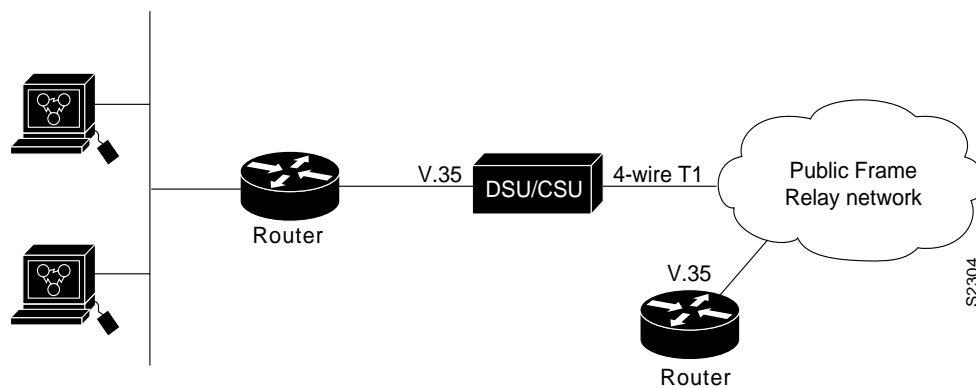
One of the following hardware configurations is possible for Frame Relay connections:

- Routers can connect directly to the Frame Relay switch.
- Routers can connect directly to a Channel Service Unit/Digital Service Unit (CSU/DSU) first, and the CSU/DSU connects to a remote Frame Relay switch.

**Note** A Frame Relay network is not required to support only routers that are connected directly or only routers connected via CSU/DSUs. Within a network, some routers can connect to a Frame Relay switch through a direct connection and others through connections via CSU/DSUs. However, a single router interface configured for Frame Relay can be only one or the other.

The CSU/DSU converts V.35 or RS-449 signals to the properly coded T1 transmission signal for successful reception by the Frame Relay network. Figure 9-1 illustrates the connections between the different components.

**Figure 9-1** Typical Frame Relay Configuration



The Frame Relay interface actually consists of one physical connection between the network server and the switch that provides the service. This single physical connection provides direct connectivity to each device on a network, such as a StrataCom FastPacket wide-area network.

## Frame Relay Configuration Task List

There are required, basic steps you must follow to enable Frame Relay for your network. In addition, you can customize Frame Relay for your particular network needs and monitor Frame Relay connections. The following sections outline these tasks. The tasks in the first three sections are required.

- Enable Frame Relay Encapsulation
- Configure Dynamic or Static Address Mapping
- Configure the LMI
- Customize Frame Relay for Your Network
- Monitor the Frame Relay Connections

The following sections describe these tasks. See the examples at the end of this chapter for ideas of how to configure Frame Relay on your network. See the “Frame Relay Commands” chapter in the *Router Products Command Reference* publication for information about the commands listed in the tasks.

## Enable Frame Relay Encapsulation

To set Frame Relay encapsulation at the interface level, perform the following tasks beginning in global configuration mode:

Task	Command
Specify the serial interface, and enter interface configuration mode.	<b>interface serial <i>number</i></b> <sup>1</sup>
Enable Frame Relay and specify the encapsulation method.	<b>encapsulation frame-relay [ietf]</b>

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

Frame Relay supports encapsulation of all supported protocols except bridging in conformance with RFC 1490, allowing interoperability between multiple vendors. Use the IETF form of Frame Relay encapsulation if your router is connected to another vendor’s equipment across a Frame Relay network. IETF encapsulation is supported at either the interface level or on a per-DLCI (map entry) basis.

For an example of how to enable Frame Relay and set the encapsulation method, see the sections “IETF Encapsulation Examples” and “Two Routers in Static Mode Example” later in this chapter. Also see the “Configuring Interfaces” chapter if you want to configure subinterfaces on serial interfaces running Frame Relay encapsulation.

## Configure Dynamic or Static Address Mapping

Dynamic address mapping uses Frame Relay Inverse ARP to request the next-hop protocol address for a specific connection, given its known DLCI. Responses to Inverse ARP requests are entered in an address-to-DLCI mapping table on the router; the table is then used to supply the next-hop protocol address or the DLCI for outgoing traffic.

Inverse ARP is enabled by default for all protocols it supports, but can be disabled for specific protocol-DLCI pairs. As a result, you can use dynamic mapping for some protocols and static mapping for other protocols on the same DLCI. You can explicitly disable Inverse ARP for a protocol-DLCI pair if you know the protocol is not supported on the other end of the connection. See the “Disable or Reenable Frame Relay Inverse ARP” section later in this chapter for more information.

### Configure Dynamic Mapping

Because Inverse ARP is enabled by default for all protocols that it supports, no additional command is required to configure dynamic mapping on an interface.

### Configure Static Mapping

A static map links a specified next-hop protocol address to a specified DLCI. Static mapping removes the need for Inverse ARP requests; when you supply a static map, Inverse ARP is automatically disabled for the specified protocol on the specified DLCI.

You must use static mapping if the router at the other end either does not support Inverse ARP at all or does not support Inverse ARP for a specific protocol that you want to use over Frame Relay.

To establish static mapping according to your network needs, perform one of the following tasks in interface configuration mode:

Task	Command
Define the mapping between a next-hop protocol address and the DLCI used to connect to the address.	<b>frame-relay map</b> <i>protocol protocol-address dlci</i> [ <b>broadcast</b> ] [ <b>ietf</b> ] [ <b>cisco</b> ]
Define a DLCI used to send ISO CLNS frames.	<b>frame-relay map clns</b> <i>dlci</i> [ <b>broadcast</b> ]
Define a DLCI used to connect to a bridge.	<b>frame-relay map bridge</b> <i>dlci</i> [ <b>broadcast</b> ]

The supported protocols and the corresponding keywords to enable them are as follows:

- IP—**ip**
- DECnet—**decnet**
- AppleTalk—**appletalk**
- XNS—**xns**
- Novell IPX—**ipx**
- VINES—**vines**
- ISO CLNS—**clns**

You can be greatly simplify the configuration for the Open Shortest Path First (OSPF) protocol by adding the optional **broadcast** keyword when doing this task. See the **frame-relay map** description in the *Router Products Command Reference* publication and the examples at the end of this chapter for more information about using the **broadcast** keyword.

For an example of how to establish static address mapping, see the sections “Two Routers in Static Mode Example,” “DECnet Routing Example,” and “IPX Routing Example” later in this chapter.

## Configure the LMI

Our Frame Relay software supports the industry-accepted standards for addressing the Local Management Interface (LMI), including the Cisco specification. To configure the LMI, complete the tasks in the following sections. The tasks in the first two sections are required.

- Set the LMI Type
- Set the LMI Keepalive Interval
- Set the LMI Polling and Timer Intervals

### Set the LMI Type

If the router is attached to a public data network, the LMI type must match the type used on the public network. Otherwise, you can select an LMI type to suit the needs of your private Frame Relay network.

You can set one of three types of LMIs on our router: ANSI T1.617 Annex D, Cisco, and ITU-T Q.933 Annex A. To do so, perform the following task in interface configuration mode:

Task	Command
Set the LMI type.	<b>frame-relay lmi-type</b> {ansi   cisco   q933a}

The default LMI type is **cisco**.

For an example of how to set the LMI type, see the “Pure Frame Relay DCE Example” section later in this chapter.

### Set the LMI Keepalive Interval

A keepalive interval must be set to enable the LMI. By default, this interval is ten seconds and, per the LMI protocol, must be less than the corresponding interval on the switch. To set the keepalive interval, perform the following task in interface configuration mode:

Task	Command
Set the keepalive interval.	<b>frame-relay keepalive</b> <i>number</i>
Turn off keepalives on networks without an LMI.	<b>no frame-relay keepalive</b>

This command has the same effect as the **keepalive** interface configuration command.

For an example of how to specify an LMI keepalive interval, see the “Two Routers in Static Mode Example” section later in this chapter.

### Set the LMI Polling and Timer Intervals

You can set various optional counters, intervals, and thresholds to fine-tune the operation of your LMI DTE and DCE devices. Set these attributes by performing one or more of the following tasks in interface configuration mode:

Task	Command
Set the DCE and NNI error threshold.	<b>frame-relay lmi-n392dce</b> <i>threshold</i>
Set the DCE and NNI monitored events count.	<b>frame-relay lmi-n393dce</b> <i>events</i>
Set the polling verification timer on a DCE or NNI interface.	<b>frame-relay lmi-t392dce</b> <i>timer</i>
Set a full status polling interval on a DTE or NNI interface.	<b>frame-relay lmi-n391dte</b> <i>keep-exchanges</i>
Set the DTE or NNI error threshold.	<b>frame-relay lmi-n392dte</b> <i>threshold</i>
Set the DTE and NNI monitored events count.	<b>frame-relay lmi-n393dte</b> <i>events</i>

See the “Frame Relay Commands” chapter in the *Router Products Command Reference* publication for details about commands used to set the polling and timing intervals.

## Customize Frame Relay for Your Network

Perform the tasks in the following sections to customize Frame Relay:

- Understand and Define Frame Relay Subinterfaces
- Configure Frame Relay Switching
- Disable or Reenable Frame Relay Inverse ARP (multipoint communication only)
- Create a Broadcast Queue for an Interface
- Configure TCP/IP Header Compression
- Configure Discard Eligibility

### Understand and Define Frame Relay Subinterfaces

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

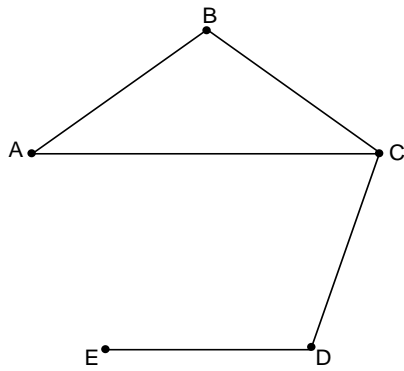
Subinterfaces provide a mechanism for supporting partially meshed Frame Relay networks. In the past, a single network number (such as an IP subnet or an IPX network number) was assigned to an entire Frame Relay network. Most protocols assume transitivity on a logical network; that is, if station A can 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.)

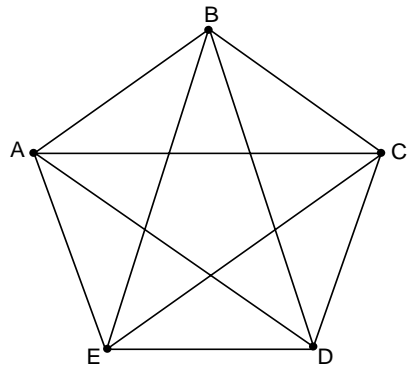
For example, suppose you have a five-node Frame Relay network (see Figure 9-2) that is partially meshed (Network A). If the entire network is viewed as a single subnetwork (with a single network number assigned), most protocols assume that node A can transmit a packet directly to node E, when in fact 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.

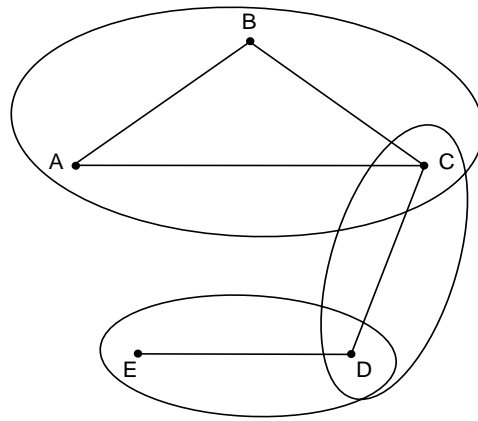
**Figure 9-2 Using Subinterfaces to Provide Full Connectivity on a Partially Meshed Frame Relay Network**



Network A: Partially meshed Frame Relay network without full connectivity



Network B: Fully meshed Frame Relay network with full connectivity



Network C: Partially meshed Frame Relay network with full connectivity (configuring subinterfaces)

93299



## Define Frame Relay Subinterfaces

To configure subinterfaces on a Frame Relay network, perform the following tasks:

Task	Command
<b>Step 1</b> Specify a serial interface.	<b>interface serial</b> <i>number</i> <sup>1</sup>
<b>Step 2</b> Configure Frame Relay encapsulation on the serial interface.	<b>encapsulation frame-relay</b>
<b>Step 3</b> Specify a subinterface.	<b>interface serial</b> <i>number.subinterface-number</i> <b>[multipoint   point-to-point]</b> <sup>1</sup>

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

Subinterfaces can be configured for multipoint or point-to-point communication. Multipoint is the default.

Once you have defined the subinterface, perform the tasks in the following sections in interface configuration mode. You must perform one of the tasks in the first three sections; the task in the fourth section is optional:

- Associate a DLCI with a Subinterface (for point-to-point communications)
- Configure Dynamic or Static Address Mapping (for multipoint communications)
- Configure a Backup Interface for a Subinterface

If you define a subinterface for multipoint communication, you do not need to associate a DLCI with the subinterface (and you cannot use the **frame-relay interface-dlci** command). If you define a subinterface for point-to-point communication, you do not need to establish mapping (and you cannot use the **frame-relay map** command). The **frame-relay inverse-arp** command is designed for use with an interface configured for multipoint communication and should not be used for a subinterface configured for point-to-point communication.

---

**Note** If you define a subinterface for point-to-point communication, you cannot reassign *the same subinterface number* to be used for multipoint communication without first rebooting the router. However, you can use a different subinterface number.

---

For an example of how to define a subinterface, see the section “Basic Subinterface Examples” later in this chapter.

## Associate a DLCI with a Subinterface

You must associate a Frame Relay DLCI with a subinterface to use subinterfaces in the Frame Relay network for point-to-point communication. If you associate a DLCI with a point-to-point subinterface, you cannot use the **frame-relay map** command.

To associate a DLCI with a subinterface, perform the following task in interface configuration mode:

Task	Command
Associate a DLCI with a subinterface.	<b>frame-relay interface-dlci</b> <i>dlci</i> [ <i>option</i> ]

For an example of how to associate a DLCI with a subinterface, see the section “Basic Subinterface Examples” later in this chapter.

### Configure Dynamic or Static Address Mapping

You can configure static or dynamic address mapping on subinterfaces, just as you can on major interfaces.

Dynamic address mapping uses Frame Relay Inverse ARP to request the next-hop protocol address for a specific connection, given its known DLCI. Responses to Inverse ARP requests are entered in an address-to-DLCI mapping table on the router; the table is then used to supply the next-hop protocol address or the DLCI for outgoing traffic.

Inverse ARP is enabled by default for all protocols it supports, but can be disabled for specific protocol-DLCI pairs. As a result, you can use dynamic mapping for some protocols and static mapping for other protocols on the same DLCI. You can explicitly disable Inverse ARP for a protocol-DLCI pair if you know the protocol is not supported on the other end of the connection. See the “Disable or Reenable Frame Relay Inverse ARP” section later in this chapter for more information.

### Configure Dynamic Mapping

Because Inverse ARP is enabled by default for all protocols that it supports, no additional command is required to configure dynamic mapping on an interface.

### Configure Static Mapping

A static map links a specified next-hop protocol address to a specified DLCI. Static mapping removes the need for Inverse ARP requests; when you supply a static map, Inverse ARP is automatically disabled for the specified protocol on the specified DLCI.

You must use static mapping if the router at the other end either does not support Inverse ARP at all or does not support Inverse ARP for a specific protocol that you want to use over Frame Relay.

To establish static mapping according to your network needs, perform one of the following tasks in interface configuration mode:

Task	Command
Define the mapping between a next-hop protocol address and the DLCI used to connect to the address.	<b>frame-relay map</b> <i>protocol protocol-address dlci</i> [ <b>broadcast</b> ] [ <b>ietf</b> ] [ <b>cisco</b> ]
Define a DLCI used to send ISO CLNS frames.	<b>frame-relay map clns</b> <i>dlci</i> [ <b>broadcast</b> ]
Define a DLCI used to connect to a bridge.	<b>frame-relay map bridge</b> <i>dlci</i> [ <b>broadcast</b> ]

The supported protocols and the corresponding keywords to enable them are as follows:

- IP—**ip**
- DECnet—**decnet**
- AppleTalk—**appletalk**
- XNS—**xns**

- Novell IPX—**ipx**
- VINES—**vines**
- ISO CLNS—**clns**

You can be greatly simplify the configuration for the Open Shortest Path First (OSPF) protocol by adding the optional **broadcast** keyword when doing this task. See the **frame-relay map** description in the *Router Products Command Reference* publication and the examples at the end of this chapter for more information about using the **broadcast** keyword.

For an example of how to establish static address mapping, see the sections “Two Routers in Static Mode Example,” “DECnet Routing Example,” and “IPX Routing Example” later in this chapter.

### Configure a Backup Interface for a Subinterface

Both point-to-point and multipoint Frame Relay subinterfaces can be configured with a backup interface. This allows individual PVCs to be backed up in case of failure rather than depending on the entire Frame Relay connection to fail before the backup takes over. You can configure a subinterface for backup on failure only, not for backup based on loading of the line.

If the serial interface has a backup interface, it will have precedence over the subinterface’s backup interface in the case of complete loss of connectivity with the Frame Relay network. As a result, a subinterface backup is activated only if the serial interface is up, or if the serial interface is down and does not have a backup interface defined. If a subinterface has failed while its backup is in use, and then the serial interface goes down, the subinterface backup stays connected.

To configure a backup interface for a Frame Relay subinterface, perform the following tasks, beginning in global configuration mode:

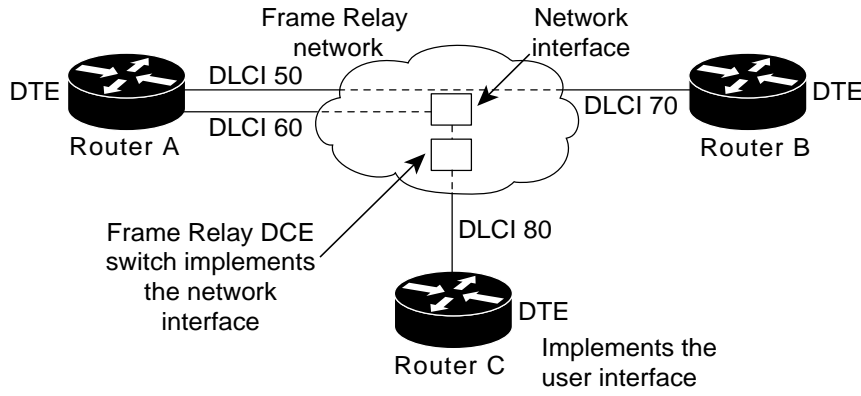
Task	Command
<b>Step 1</b> Specify the interface.	<b>interface serial <i>number</i></b> <sup>1</sup>
<b>Step 2</b> Configure Frame Relay encapsulation.	<b>encapsulation frame-relay</b>
<b>Step 3</b> Configure the subinterface.	<b>interface serial <i>number.subinterface-number point-to-point</i></b> <sup>1</sup>
<b>Step 4</b> Specify a DLCI for the subinterface.	<b>frame-relay interface-dlci <i>dlci</i></b>
<b>Step 5</b> Specify a backup interface for the subinterface.	<b>backup interface serial <i>number</i></b> <sup>1</sup>
<b>Step 6</b> Specify backup enable and disable delay.	<b>backup delay <i>enable-delay disable-delay</i></b> <sup>1</sup>

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

### Configure Frame Relay Switching

Frame Relay switching is a means of switching packets based upon the DLCI, which can be looked upon as the Frame Relay equivalent of a MAC address. The switching is performed by configuring your router as a Frame Relay network. There are two parts to a Frame Relay network: a Frame Relay DTE (the router) and a Frame Relay DCE switch. Figure 9-3 illustrates this concept.

Figure 9-3 Frame Relay Switched Network



S1463a

In Figure 9-3, Routers A, B, and C are Frame Relay DTEs connected to each other via a Frame Relay network. Our implementation of Frame Relay switching allows our routers to be used as depicted in this Frame Relay network.

Perform the tasks in the following sections, as necessary, to configure Frame Relay switching:

- Enable Frame Relay switching.
- Configure a Frame Relay DTE device, DCE switch, or NNI support.
- Specify the static route.

These tasks are described in the following sections.

### Enable Frame Relay Switching

You must enable packet switching before you can configure it on a Frame Relay DTE, DCE, or with Network to Network Interface (NNI) support. Do so by performing the following task in global configuration mode before configuring the switch type:

Task	Command
Enable Frame Relay switching.	<b>frame-relay switching</b>

For an example of how to enable Frame Relay switching, see the switching examples later in this chapter.

### Configure a Frame Relay DTE Device, DCE Switch, or NNI Support

You can configure an interface as a DTE device, DCE switch, or as a switch connected to a switch to support NNI connections. (DCE is the default.) To do so, perform the following task in interface configuration mode:

Task	Command
Configure a Frame Relay DTE device or DCE switch.	<b>frame-relay intf-type [dce   dte   nni]</b>

For an example of how to configure a DTE device or DCE switch, see the section “Hybrid DTE/DCE PVC Switching Example” later in this chapter.

For an example of how to configure NNI support, see the section “Pure Frame Relay DCE Example” later in this chapter.

### Specify the Static Route

You must specify a static route for PVC switching. To do so, perform the following task in interface configuration mode:

Task	Command
Specify a static route for PVC switching.	<b>frame-relay route</b> <i>in-dlci out-interface out-dlci</i>

For an example of how to specify a static route, see the section “Pure Frame Relay DCE Example” later in this chapter.

### Disable or Reenable Frame Relay Inverse ARP

Frame Relay Inverse ARP is a method of building dynamic address mappings in Frame Relay networks running AppleTalk, Banyan VINES, DECnet, IP, Novell IPX, and XNS. Inverse ARP allows the router to discover the protocol address of a device associated with the virtual circuit.

Inverse ARP creates dynamic address mappings, as contrasted with the **frame-relay map** command, which defines static mappings between a specific protocol address and a specific DLCI (see the section “Configure Dynamic or Static Address Mapping” earlier in this chapter for more information).

Inverse ARP is enabled by default but can be disabled explicitly for a given protocol and DLCI pair. Disable or reenables Inverse ARP in the following conditions:

- Disable Inverse ARP for a selected protocol and DLCI pair when you know that the protocol is not supported on the other end of the connection.
- Reenable Inverse ARP for a protocol and DLCI pair if conditions or equipment change and the protocol is then supported on the other end of the connection.

---

**Note** If you change from a point-to-point subinterface to a multipoint subinterface, then change the subinterface number. Frame Relay Inverse ARP will be on by default, and no further action is required.

---

You do not need to enable or disable Inverse ARP if you have a point-to-point interface, because there is only a single destination and discovery is not required.

To select Inverse ARP, perform the following task in interface configuration mode:

Task	Command
Enable Frame Relay Inverse ARP for a specific protocol and DLCI pair, only if it was previously disabled.	<b>frame-relay inverse-arp</b> <i>protocol dlci</i>
Disable Frame Relay Inverse ARP for a specific protocol and DLCI pair.	<b>no frame relay inverse-arp</b> <i>protocol dlci</i>

## Create a Broadcast Queue for an Interface

Very large Frame Relay networks might have performance problems when very many DLCIs terminate in a single router and the router must replicate routing updates and service advertising updates on each DLCI. The updates can consume access-link bandwidth and cause significant latency variations in user traffic; the updates can also consume interface buffers and lead to higher packet rate loss for both user data and routing updates.

To avoid such problems, you can create a special broadcast queue for an interface. The broadcast queue is managed independently of the normal interface queue, has its own buffers, and has a configurable size and service rate.

A broadcast queue is given a maximum transmission rate (throughput) limit measured in both bytes per second and packets per second. The queue is serviced to ensure that no more than this maximum is provided. The broadcast queue has priority when transmitting at a rate below the configured maximum, and hence has a guaranteed minimum bandwidth allocation. The two transmission rate limits are intended to avoid flooding the interface with broadcasts. The actual transmission rate limit in any second is the first of the two rate limits that is reached.

To create a broadcast queue, complete the following task in interface configuration mode:

Task	Command
Create a broadcast queue for an interface.	<b>frame-relay broadcast-queue</b> <i>size byte-rate</i> <i>packet-rate</i>

## Configure TCP/IP Header Compression

TCP/IP header compression, as described by RFC 1144, is designed to improve the efficiency of bandwidth utilization over low-speed serial links. A typical TCP/IP packet includes a 40-byte datagram header. Once a connection is established, the header information is redundant and need not be repeated in every packet that is sent. By reconstructing a smaller header that identifies the connection and indicates the fields that changed and the amount of change, fewer bytes can be transmitted. The average compressed header is 10 bytes long.

For this algorithm to function, packets must arrive in order. If packets arrive out of order, the reconstruction will appear to create regular TCP/IP packets but the packets will not match the original. Because priority queuing changes the order in which packets are transmitted, enabling priority queuing on the interface is not recommended.

You can configure TCP/IP header compression in either of two ways, as described in the following sections:

- Configure an individual IP map for TCP/IP header compression.
- Configure an interface for TCP/IP header compression.

The “Disable TCP/IP Header Compression” section describes how to disable this feature.

---

**Note** If you configure an interface with Cisco encapsulation and TCP/IP header compression, Frame Relay IP maps inherit the compression characteristics of the interface. However, if you configure the interface with IETF encapsulation, the interface cannot be configured for compression. Frame Relay maps will have to be configured individually to support TCP/IP header compression.

---

### Configure an Individual IP Map for TCP/IP Header Compression

TCP/IP header compression requires Cisco encapsulation. If you need to have IETF encapsulation on an interface as a whole, you can still configure a specific IP map to use Cisco encapsulation and TCP header compression.

In addition, even if you configure the interface to perform TCP/IP header compression, you can still configure a specific IP map not to compress TCP/IP headers.

You can specify whether TCP/IP header compression is active or passive. Active compression subjects every outgoing packet to TCP/IP header compression. Passive compression subjects an outgoing TCP/IP packet to header compression only if the packet had a compressed TCP/IP header when it was received.

To configure an IP map to use Cisco encapsulation and TCP/IP header compression, perform the following task in interface configuration mode:

Task	Command
Configure an IP map to use Cisco encapsulation and TCP/IP header compression.	<b>frame-relay map ip-address dlcid [broadcast] cisco tcp header-compression {active   passive}</b>

The default encapsulation is **cisco**.

---

**Note** An interface that is configured to support TCP/IP header compression cannot also support priority queuing or custom queuing.

---

For an example of how to configure TCP header compression on an IP map, see the “TCP/IP Header Compression Examples” section later in this chapter.

### Configure an Interface for TCP/IP Header Compression

You can configure the interface with active or passive TCP/IP header compression. Active compression, the default, subjects all outgoing TCP/IP packets to header compression. Passive compression subjects an outgoing packet to header compression only if the packet had a compressed TCP/IP header when it was received on that interface.

To apply TCP/IP header compression to an interface, you must perform the following tasks in interface configuration mode:

Task	Command
Configure Cisco encapsulation on the interface.	<b>encapsulation frame-relay</b>
Enable TCP/IP header compression on the interface.	<b>frame-relay ip tcp header-compression [passive]</b>

---

**Note** If an interface configured with Cisco encapsulation is later configured with IETF encapsulation, all TCP/IP header compression characteristics are lost. To apply TCP/IP header compression over an interface configured with IETF encapsulation, you must configure individual IP maps, as described in the section “Configure an Individual IP Map for TCP/IP Header Compression.”

---

For an example of how to configure TCP header compression on an interface, see the “TCP/IP Header Compression Examples” section later in this chapter.

### Disable TCP/IP Header Compression

You can disable TCP/IP header compression by using either of two commands that have different effects, depending on whether Frame Relay IP maps have been explicitly configured for TCP/IP header compression or have inherited their compression characteristics from the interface.

Frame Relay IP maps that have explicitly configured TCP/IP header compression must also have TCP/IP header compression explicitly disabled.

To disable TCP/IP header compression, perform one of the following tasks in interface configuration mode:

Task	Command
Disable TCP header compression on all Frame Relay IP maps that are not explicitly configured for TCP header compression.	<b>no frame-relay ip tcp header-compression</b>
or	
Disable TCP header compression on a specified Frame Relay IP map.	<b>frame-relay map ip <i>ip-address dlc</i> nocompress</b>

For an example of how to turn off TCP header compression, see the section “Disabling Inherited TCP/IP Header Compression Example.”

### Configure Discard Eligibility

You can specify which Frame Relay packets have low priority or low time sensitivity and will be the first to be dropped when a Frame Relay switch is congested. The mechanism that allows a Frame Relay switch to identify such packets is the discard eligibility (DE) bit.

This feature requires that the Frame Relay network be able to interpret the DE bit. Some networks take no action when the DE bit is set. Other networks use the DE bit to determine which packets to discard. The most desirable interpretation is to use the DE bit to determine which packets should be dropped first and also which packets have lower time sensitivity.

You can define DE lists that identify the characteristics of packets to be eligible for discarding, and you can also specify DE groups to identify the DLCI that is affected.

To define a DE list specifying the packets that can be dropped when the Frame Relay switch is congested, perform the following task in global configuration mode:

Task	Command
Define a DE list.	<b>frame-relay de-list <i>list-number</i> {<b>protocol</b> <i>protocol</i>   <b>interface</b> <i>type number</i>} <i>characteristic</i></b>

You can specify DE lists based on the protocol or the interface, and on characteristics such as fragmentation of the packet, a specific TCP or UDP port, an access list number, or packet size. See the **frame-relay de-list** command for arguments and other information.



To define a DE group specifying the DE list and DLCI affected, perform the following task in interface configuration mode:

Task	Command
Define a DE group.	<b>frame-relay de-group</b> <i>group-number dlc</i>

## Monitor the Frame Relay Connections

To monitor Frame Relay connections, perform any of the following tasks in EXEC mode:

Task	Command
Clear dynamically created Frame Relay maps, which are created by the use of inverse ARP.	<b>clear frame-relay-inarp</b>
Display information about Frame Relay DLCIs and the LMI.	<b>show interfaces serial</b> <i>number</i> <sup>1</sup>
Display LMI statistics.	<b>show frame-relay lmi</b> [ <i>type number</i> ]
Display the current Frame Relay map entries.	<b>show frame-relay map</b>
Display PVC statistics.	<b>show frame-relay pvc</b> [ <i>type number [dlci]</i> ]
Display configured static routes.	<b>show frame-relay route</b>
Display Frame Relay traffic statistics.	<b>show frame-relay traffic</b>

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

## Frame Relay Configuration Examples

This section provides examples of Frame Relay configurations. It includes the following sections:

- IETF Encapsulation Examples
- Static Address Mapping Examples
- Subinterface Examples
- Configuration Providing Backward Compatibility Example
- Booting from a Network Server over Frame Relay Example
- Frame Relay Switching Examples
- TCP/IP Header Compression Examples
- Disabling TCP/IP Header Compression Examples

### IETF Encapsulation Examples

The first example that follows sets IETF encapsulation at the interface level. The second example sets IETF encapsulation on a per-DLCI basis. In the first example, the keyword **ietf** sets the default encapsulation method for all maps to IETF.

```
encapsulation frame-relay IETF
frame-relay map ip 131.108.123.2 48 broadcast
frame-relay map ip 131.108.123.3 49 broadcast
```

In the following example, IETF encapsulation is configured on a per-DLCI basis. This configuration has the same result as the configuration in the first example.

```
encapsulation frame-relay
frame-relay map ip 131.108.123.2 48 broadcast ietf
frame-relay map ip 131.108.123.3 49 broadcast ietf
```

## Static Address Mapping Examples

The following sections provide examples of static address mapping for the IP, AppleTalk, DECnet, and IPX protocols.

### Two Routers in Static Mode Example

The following examples illustrate how to configure two routers for static mode.

#### Configuration for Router 1

```
interface serial 0
!
ip address 131.108.64.2 255.255.255.0
encapsulation frame-relay
keepalive 10
frame-relay map ip 131.108.64.1 43
```

#### Configuration for Router 2

```
interface serial 0
!
ip address 131.108.64.1 255.255.255.0
encapsulation frame-relay
keepalive 10
frame-relay map ip 131.108.64.2 43
```

### AppleTalk Routing Example

The following example illustrates how to configure two routers to communicate with each other using AppleTalk over a Frame Relay network. Each router has a Frame Relay static address map for the other router. The use of the **appletalk cable-range** command indicates that this is extended AppleTalk (Phase II).

#### Configuration for Router 1

```
!
interface Serial0
ip address 172.21.59.24 255.255.255.0
encapsulation frame-relay
appletalk cable-range 10-20 18.47
appletalk zone eng
frame-relay map appletalk 18.225 100 broadcast
```

#### Configuration for Router 1

```
!
interface Serial2/3
ip address 172.21.177.18 255.255.255.0
encapsulation frame-relay
```

```

appletalk cable-range 10-20 18.225
appletalk zone eng
clockrate 2000000
frame-relay map appletalk 18.47 100 broadcast
!

```

## DECnet Routing Example

The following example sends all DECnet packets destined for address 56.4 out on DLCI 101. In addition, any DECnet broadcasts for interface serial 1 will be sent on that DLCI.

```

decnet routing 32.6
!
interface serial 1
encapsulation frame-relay
frame-relay map decnet 56.4 101 broadcast

```

## IPX Routing Example

The following example illustrates how to send packets destined for IPX address 200.0000.0c00.7b21 out on DLCI 102:

```

ipx routing 000.0c00.7b3b
!
interface ethernet 0
ipx network 2abc
!
interface serial 0
ipx network 200
encapsulation frame-relay
frame-relay map ipx 200.0000.0c00.7b21 102 broadcast

```

## Subinterface Examples

The following sections provide basic Frame Relay subinterface examples and variations appropriate for different routed protocols and for bridging.

### Basic Subinterface Examples

In the following example, subinterface 1 models a point-to-point subnet and subinterface 2 models a broadcast subnet. For emphasis, the **multipoint** keyword is used for serial subinterface 2, even though a subinterface is multipoint by default.

```

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 10.0.2.2 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.

### IPX Routes over Frame Relay Subinterfaces Example

The following example configures a serial interface for Frame Relay encapsulation and sets up multiple IPX virtual networks corresponding to Frame Relay subinterfaces:

```
ipx routing 0000.0c02.5f4f
!
interface serial 0
encapsulation frame-relay
interface serial 0.1 multipoint
ipx network 1
frame-relay map ipx 1.000.0c07.d530 200 broadcast
ipx network 2
frame-relay map ipx 2.000.0c07.d530 300 broadcast
```

For subinterface serial 0.1, the router at the other end might be configured as follows:

```
ipx routing
interface serial 2 multipoint
ipx network 1
frame-relay map ipx 1.000.0c02.5f4f 200 broadcast
```

### Unnumbered IP over a Point-to-Point Subinterface Example

The following example sets up unnumbered IP over subinterfaces at both ends of a point-to-point connection. In this example, Router A functions as the DTE, and Router B functions as the DCE. Routers A and B are both attached to Token Ring networks.

#### Configuration for Router A

```
frame-relay switching
!
interface token-ring 0
ip address 131.108.177.1 255.255.255.0
!
interface serial 0
no ip address
encapsulation frame-relay IETF
!
interface Serial0.2 point-to-point
ip unnumbered TokenRing0
ip pim sparse-mode
frame-relay interface-dlci 20
```

### Configuration for Router B:

```

frame-relay switching
!
interface token-ring 0
ip address 131.108.178.1 255.255.255.0
!
interface serial 0
no ip address
encapsulation frame-relay IETF
bandwidth 384
clockrate 4000000
frame-relay intf-type dce
!
interface serial 0.2 point-to-point
ip unnumbered TokenRing1
ip pim sparse-mode
bandwidth 384
frame-relay interface-dlci 20

```

### Transparent Bridging Using Subinterfaces Example

In the following example, Frame Relay DLCIs 42, 64, and 73 are to be used as separate point-to-point links with transparent bridging running over them. The bridging spanning tree algorithm views each PVC as a separate bridge port, and a frame arriving on the PVC can be relayed back out a separate PVC. Be sure that routing is not enabled when configuring transparent bridging using subinterfaces.

```

interface serial 0
bridge-group 1
encapsulation frame-relay
interface serial 0.1
bridge-group 1
frame-relay interface-dlci 42
interface serial 0.2
bridge-group 1
frame-relay interface-dlci 64
interface serial 0.3
bridge-group 1
frame-relay interface-dlci 73

```

### Configuration Providing Backward Compatibility Example

The following configuration provides backward compatibility and interoperability with earlier versions that are not compliant with RFC 1490. The **ietf** keyword is used to generate RFC 1490 traffic. This configuration is possible because of the flexibility provided by separately defining each map entry.

```

encapsulation frame-relay
frame-relay map ip 131.108.123.2 48 broadcast ietf
! interoperability is provided by IETF encapsulation
frame-relay map ip 131.108.123.3 49 broadcast ietf
frame-relay map ip 131.108.123.7 58 broadcast
! this line allows the router to connect with a
! device running an older version of software
frame-relay map DECNET 21.7 49 broadcast

```

Configure IETF based on map entries and protocol for more flexibility. Use this method of configuration for backward compatibility and interoperability.

### Booting from a Network Server over Frame Relay Example

When booting from a TFTP server over Frame Relay, you cannot boot from a network server via a broadcast. You must boot from a specific TFTP host. Also, a **frame-relay map** command must exist for the host that you will boot from.

For example, if file `gs3-bfx` is to be booted from a host with IP address `131.108.126.2`, the following commands would need to be in the configuration:

```
boot system gs3-bfx 131.108.126.2

interface Serial 0
encapsulation frame-relay
frame-relay map IP 131.108.126.2 100 broadcast
```

The **frame-relay map** command is used to map an IP address into a DLCI address. In order to boot over Frame Relay, the address of the machine to netboot from must be given explicitly, and a **frame-relay map** entry must exist for that site. For example, if file `gs3-bfx.83-2.0` is to be booted from a host with IP address `131.108.126.111`, the following commands would need to be in the configuration:

```
boot system gs3-bfx.83-2.0 131.108.13.111
!
interface Serial 1
ip address 131.108.126.200 255.255.255.0
encapsulation frame-relay
frame-relay map ip 131.108.126.111 100 broadcast
```

In this case, 100 is the DLCI that can get to host `131.108.126.111`.

The remote router must have the following **frame-relay map** entry:

```
frame-relay map ip 131.108.126.200 101 broadcast
```

This entry allows the remote router to return a boot image (from the netboot host) to the router netbooting over Frame Relay. Here, 101 is a DLCI of the router being netbooted.

## Frame Relay Switching Examples

The following sections provide several examples of configuring one or more routers as Frame Relay switches:

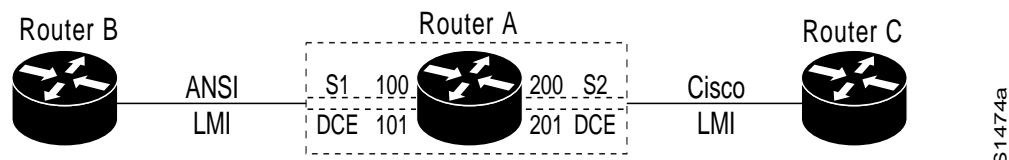
- **PVC Switching Configuration Example**—In this example, one router has two interfaces configured as DCEs; the router switches frames from the incoming interface to the outgoing interface on the basis of the DLCI alone.
- **Pure Frame Relay DCE Example**—In this example, a Frame Relay network is set up with two routers functioning as switches; standard NNI signaling is used between them.
- **Hybrid DTE/DCE PVC Switching Example**—In this example, one router is configured with both DCE and DTE interfaces (a hybrid DTE/DCE Frame Relay switch). It can switch frames between two DCE ports and between a DCE port and a DTE port.
- **Switching over an IP Tunnel Example**—In this example, two routers are configured to switch Frame Relay PVCs over a point-to-point IP tunnel.

### PVC Switching Configuration Example

You can configure your router as a dedicated, DCE-only Frame Relay switch. Switching is based on DLCIs. The incoming DLCI is examined, and the outgoing interface and DLCI are determined. Switching takes place when the incoming DLCI in the packet is replaced by the outgoing DLCI, and the packet is sent out the outgoing interface.

In the following example, the router switches two PVCs between interface serial 1 and 2. Frames with DLCI 100 received on serial 1 will be transmitted with DLCI 200 on serial 2 (see Figure 9-4).

**Figure 9-4 PVC Switching Configuration**



### Configuration for Router A

```
frame-relay switching
!
interface Ethernet0
ip address 131.108.160.58 255.255.255.0
!
interface Serial1
no ip address
encapsulation frame-relay
keepalive 15
frame-relay lmi-type ansi
frame-relay intf-type dce
```

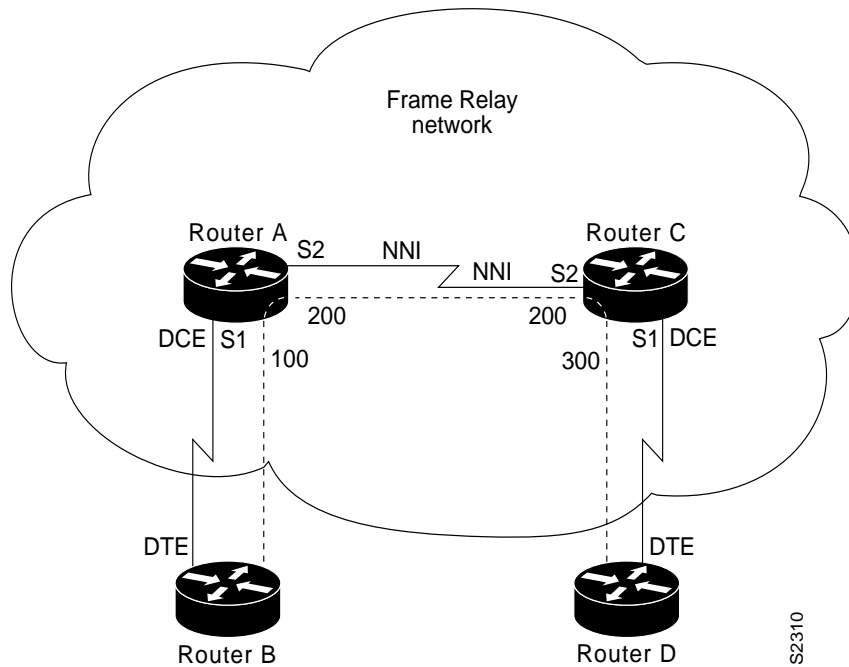
```

frame-relay route 100 interface Serial2 200
frame-relay route 101 interface Serial2 201
clockrate 2000000
!
interface Serial2
encapsulation frame-relay
keepalive 15
frame-relay intf-type dce
frame-relay route 200 interface Serial1 100
frame-relay route 201 interface Serial1 101
clockrate 64000
    
```

Pure Frame Relay DCE Example

Using the PVC switching feature, it is possible to build an entire Frame Relay network using our routers. In the following example, Router A and Router C act as Frame Relay switches implementing a two-node network. The standard Network to Network Interface (NNI) signaling protocol is used between Router A and Router C (see Figure 9-5).

Figure 9-5 Frame Relay DCE Configuration



Configuration for Router A

```

frame-relay switching
!
interface ethernet 0
no ip address
shutdown
!
interface ethernet 1
no ip address
shutdown
!
    
```



```

interface ethernet 2
no ip address
shutdown
!
interface ethernet 3
no ip address
shutdown
!
interface serial 0
ip address 131.108.178.48 255.255.255.0
shutdown
!
interface serial 1
no ip address
encapsulation frame-relay
frame-relay intf-type dce
frame-relay lmi-type ansi
frame-relay route 100 interface serial 2 200
!
interface serial 2
no ip address
encapsulation frame-relay
frame-relay intf-type nni
frame-relay lmi-type q933a
frame-relay route 200 interface serial 1 100
clockrate 2048000
!
interface serial 3
no ip address
shutdown

```

### Configuration for Router C

```

frame-relay switching
!
interface ethernet 0
no ip address
shutdown
!
interface ethernet1
no ip address
shutdown
!
interface ethernet 2
no ip address
shutdown
!
interface ethernet 3
no ip address
shutdown
!
interface serial 0
ip address 131.108.187.84 255.255.255.0
shutdown
!
interface serial 1
no ip address
encapsulation frame-relay
frame-relay intf-type dce
frame-relay route 300 interface serial 2 200
!
interface serial 2
no ip address
encapsulation frame-relay

```

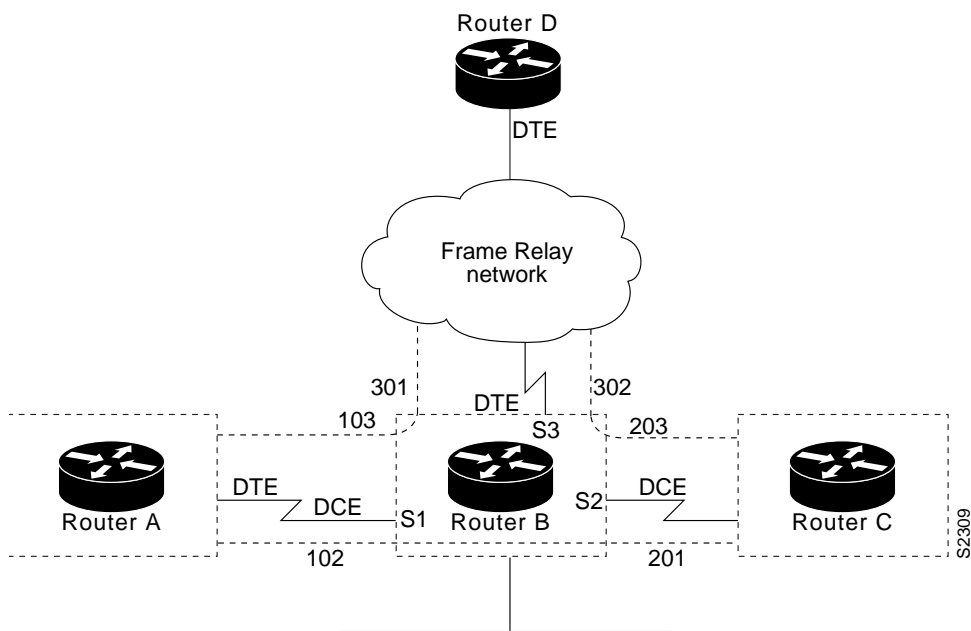
```

frame-relay intf-type nni
frame-relay lmi-type q933a
frame-relay route 200 interface serial 1 300
!
interface serial 3
no ip address
shutdown
    
```

### Hybrid DTE/DCE PVC Switching Example

Routers can also be configured as hybrid DTE/DCE Frame Relay switches (see Figure 9-6).

**Figure 9-6 Hybrid DTE/DCE PVC Switching**



In the following example, Router B acts as a hybrid DTE/DCE Frame Relay switch. It can switch frames between the two DCE ports and between a DCE port and a DTE port. Traffic from the Frame Relay network can also be terminated locally. In the example, three PVCs are defined, as follows:

- Serial 1, DLCI 102 to serial 2, DLCI 201 :DCE switching
- Serial 1, DLCI 103 to serial 3, DLCI 301 :DCE/DTE switching
- Serial 2, DLCI 203 to serial 3, DLCI 302 :DCE/DTE switching

DLCI 400 is also defined for locally terminated traffic.

### Configuration for Router B

```

frame-relay switching
!
interface ethernet 0
ip address 131.108.123.231 255.255.255.0
!
    
```

```

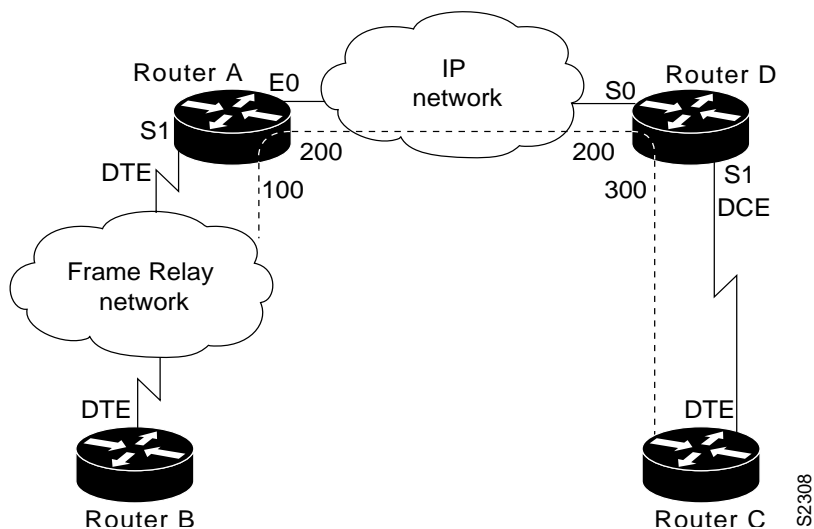
interface ethernet 1
ip address 131.108.5.231 255.255.255.0
!
interface serial 0
no ip address
shutdown
!
interface serial 1
no ip address
encapsulation frame-relay
frame-relay intf-type dce
frame-relay route 102 interface serial 2 201
frame-relay route 103 interface serial 3 301
!
interface serial 2
no ip address
encapsulation frame-relay
frame-relay intf-type dce
frame-relay route 201 interface serial 1 102
frame-relay route 203 interface serial 3 302
!
interface serial 3
ip address 131.108.111.231
encapsulation frame-relay
frame-relay lmi-type ansi
frame-relay route 301 interface serial 1 103
frame-relay route 302 interface serial 1 203
frame-relay map ip 131.108.111.4 400 broadcast

```

### Switching over an IP Tunnel Example

Switching over an IP tunnel is done by creating a point-to-point tunnel across the internetwork over which PVC switching can take place (see Figure 9-7).

**Figure 9-7** Frame Relay Switch over IP Tunnel



The following configurations illustrate how to create the IP network depicted in Figure 9-7.

### Configuration for Router A

```
frame-relay switching
!
interface Ethernet0
ip address 108.131.123.231 255.255.255.0
!
interface Ethernet1
ip address 131.108.5.231 255.255.255.0
!
interface Serial0
no ip address
shutdown
!
interface Serial1
ip address 131.108.222.231 255.255.255.0
encapsulation frame-relay
frame-relay map ip 131.108.222.4 400 broadcast
frame-relay route 100 interface Tunnell 200
!
interface Tunnell
tunnel source Ethernet0
tunnel destination 150.150.150.123
```

### Configuration for Router D

```
frame-relay switching
!
interface Ethernet0
ip address 131.108.231.123 255.255.255.0
!
interface Ethernet1
ip address 131.108.6.123 255.255.255.0
!
interface Serial0
ip address 150.150.150.123 255.255.255.0
encapsulation ppp
!
interface Tunnell
tunnel source Serial0
tunnel destination 108.131.123.231
!
interface Serial1
ip address 131.108.7.123 255.255.255.0
encapsulation frame-relay
frame-relay intf-type dce
frame-relay route 300 interface Tunnell 200
```

## TCP/IP Header Compression Examples

The following examples show various combinations of TCP/IP header compression and encapsulation characteristics on the interface and the effect on the inheritance of those characteristics on a Frame Relay IP map.

### IP Map with Inherited TCP/IP Header Compression Example

The following example shows an interface configured for TCP/IP header compression and an IP map that inherits the compression characteristics. Note that the Frame Relay IP map is not explicitly configured for header compression.

```
interface serial 1
encapsulation frame-relay
ip address 131.108.177.178 255.255.255.0
frame-relay map ip 131.108.177.177 177 broadcast
frame-relay ip tcp header-compression passive
```

Use of the **show frame-relay map** command will display the resulting compression and encapsulation characteristics; the IP map has inherited passive TCP/IP header compression:

```
Router> show frame-relay map
Serial 1 (administratively down): ip 131.108.177.177
      dlcil 177 (0xB1,0x2C10), static,
      broadcast,
      CISCO
      TCP/IP Header Compression (inherited), passive (inherited)
```

### Using an IP Map to Override TCP/IP Header Compression Example

The following example shows the use of a Frame Relay IP map to override the compression set on the interface:

```
interface serial 1
encapsulation frame-relay
ip address 131.108.177.178 255.255.255.0
frame-relay map ip 131.108.177.177 177 broadcast nocompress
frame-relay ip tcp header-compression passive
```

Use of the **show frame-relay map** command will display the resulting compression and encapsulation characteristics; the IP map has not inherited TCP header compression:

```
Serial 1 (administratively down): ip 131.108.177.177
      dlcil 177 (0xB1,0x2C10), static,
      broadcast,
      CISCO
```

### Disabling TCP/IP Header Compression Examples

The following examples show the use of two different commands to disable TCP/IP header compression.

#### Disabling Inherited TCP/IP Header Compression Example

In this first example, the initial configuration is the following:

```
interface serial 1
encapsulation frame-relay
ip address 131.108.177.179 255.255.255.0
frame-relay ip tcp header-compression passive
frame-relay map ip 131.108.177.177 177 broadcast
frame-relay map ip 131.108.177.178 178 broadcast tcp header-compression
```

You enter the following commands:

```
serial interface 1
no frame-relay ip tcp header-compression
```

Use of the **show frame-relay map** command will display the resulting compression and encapsulation characteristics:

```
Router> show frame-relay map
Serial 1 (administratively down): ip 131.108.177.177 177
```

```
                dlci 177(0xB1, 0x2C10), static,
                broadcast
                CISCO
Serial 1      (administratively down): ip 131.108.177.178 178
                dlci 178(0xB2,0x2C20), static
                broadcast
                CISCO
                TCP/IP Header Compression (enabled)
```

As a result, header compression is disabled for the first map (with DLCI 177), which inherited its header compression characteristics from the interface, but not disabled for the second map (DLCI 178), which is explicitly configured for header compression.

### Disabling Explicit TCP/IP Header Compression Example

In this second example, the initial configuration is the same as the previous example, but you enter the following commands:

```
serial interface 1
no frame-relay ip tcp header-compression
frame-relay map ip 131.108.177.178 178 nocompress
```

Use of the **show frame-relay map** command will display the resulting compression and encapsulation characteristics:

```
Router> show frame-relay map
Serial 1      (administratively down): ip 131.108.177.177 177
                dlci 177(0xB1,0x2C10), static,
                broadcast
                CISCO
Serial 1      (administratively down): ip 131.108.177.178 178
                dlci 178(0xB2,0x2C20), static
                broadcast
                CISCO
```

The result of the commands is to disable header compression for the first map (with DLCI 177), which inherited its header compression characteristics from the interface, and also explicitly to disable header compression for the second map (with DLCI 178), which had been explicitly configured for header compression.