

# Configuring Frame Relay

---

This chapter describes the tasks for configuring Frame Relay on a router or access server. For a complete description of the Frame Relay commands mentioned in this chapter, refer to the “Frame Relay Commands” chapter in the *Wide-Area Networking Command Reference*. To locate documentation of specific commands, use the command reference, master index or search online.

Refer to the following chapters in other publications for information about the topics indicated:

To perform this task . . .	See this chapter . . .
	<i>Dial Solutions Configuration Guide</i>
Configure DDR over Frame Relay	“Dial-on-Demand” <i>Configuration Fundamentals Configuration Guide</i>
Install software on a new router or access server by downloading software from a central server over an interface that supports Frame Relay	“Loading and Maintaining System Images and Microcode” <i>Bridging and IBM Networking Configuration Guide</i>
Configure access between SNA devices over a Frame Relay network	“Configuring SNA Frame Relay Access Support”
Configure serial tunnel (STUN) and block serial tunnel encapsulation between SNA devices over a Frame Relay network	“Configuring Serial Tunnel and Block Serial Tunnel”
Configure source-route bridging between SNA devices over a Frame Relay network	“Configuring Source-Route Bridging”

## Cisco Frame Relay MIB

The Cisco Frame Relay MIB adds extensions to the standard Frame Relay MIB (RFC 1315). It provides additional link-level and virtual circuit-level information and statistics that are mostly specific to Cisco Frame Relay implementation. This MIB provides SNMP network management access to most of the information covered by the **show frame-relay** commands, such as, **show frame-relay lmi**, **show frame-relay pvc**, **show frame-relay map**, and **show frame-relay svc**.

## Frame Relay Hardware Configurations

You can create Frame Relay connections using one of the following hardware configurations:

- Connect routers and access servers directly to the Frame Relay switch.
- Connect routers and access servers directly to a channel service unit/digital service unit (CSU/DSU), which then connects to a remote Frame Relay switch.

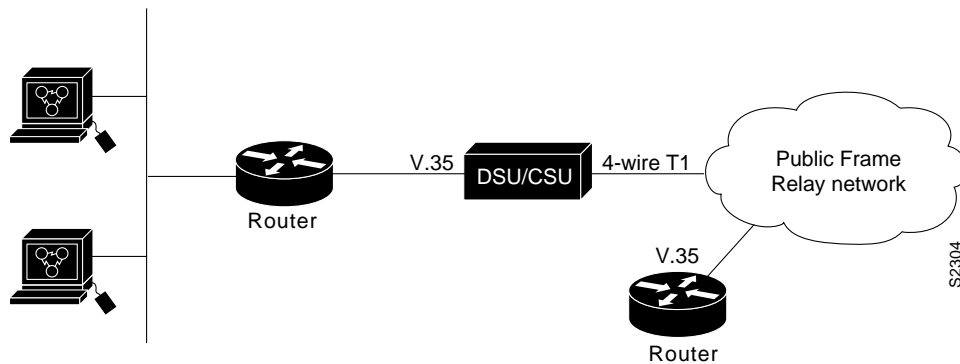
---

**Note** Routers can connect to Frame Relay networks either by direct connection to a Frame Relay switch or through CSU/DSUs. However, a single router interface configured for Frame Relay can only be configured for one of these methods.

---

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 15 illustrates the connections between the different components.

**Figure 15** 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 (WAN).

## Frame Relay Configuration Task List

You must follow certain required, basic steps 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 described in the following sections are required:

- Enable Frame Relay Encapsulation on an Interface
- Configure Dynamic or Static Address Mapping

The tasks described in the following sections are optional and are used to customize Frame Relay:

- Configure the LMI
- Configure Frame Relay Switched Virtual Circuits

- Configure Frame Relay Traffic Shaping
- Customize Frame Relay for Your Network
- Monitor and Maintain the Frame Relay Connections

See the “Frame Relay Configuration Examples” section 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 *Wide-Area Networking Command Reference* for information about the Frame Relay commands listed in the tasks. Use the index or search online for documentation of other commands.

## Enable Frame Relay Encapsulation on an Interface

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

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

Frame Relay supports encapsulation of all supported protocols in conformance with RFC 1490, allowing interoperability between multiple vendors. Use the Internet Engineering Task Force (IETF) form of Frame Relay encapsulation if your router or access server is connected to another vendor’s equipment across a Frame Relay network. IETF encapsulation is supported either at the interface level or on a per-virtual circuit basis.

For an example of how to enable Frame Relay and set the encapsulation method, see the sections “IETF Encapsulation Examples” and “Static Address Mapping Examples” later in this chapter.

We recommend that you shut down the interface prior to changing encapsulation types. Although this is not required, shutting down the interface ensures the interface is reset for the new 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 or access server; 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 that 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

Inverse ARP is enabled by default for all protocols enabled on the physical interface. Packets are not sent out for protocols that are not enabled on the interface.

Because Inverse ARP is enabled by default, 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 dlc</i> [ <b>broadcast</b> ] [ <b>ietf</b> ] [ <b>cisco</b> ]
Define a DLCI used to send International Organization for Standardization (ISO) Connectionless Network Service (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> ] [ <b>ietf</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 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** command description in the *Wide-Area Networking Command Reference* and the examples at the end of this chapter for more information about using the **broadcast** keyword.

For examples of how to establish static address mapping, see the “Static Address Mapping Examples” section later in this chapter.

## Configure the LMI

Beginning with Cisco IOS Release 11.2, the software supports Local Management Interface (LMI) *autosense*, which enables the interface to determine the LMI type supported by the switch. Support for LMI autosense means that you are no longer required to configure the Local Management Interface (LMI) explicitly.

---

For information on using Enhanced Local Management Interface with traffic shaping, see “Configure Frame Relay Traffic Shaping” later in this chapter.

## Allow LMI Autosense to Operate

LMI autosense is active in the following situations:

- The router is powered up or the interface changes state to up.
- The line protocol is down but the line is up.
- The interface is a Frame Relay DTE.
- The LMI type is not explicitly configured.

## Status Request

When LMI autosense is active, it sends out a full status request, in all three LMI flavors, to the switch. The order is ANSI, ITU, cisco but is done in rapid succession. Unlike previous software capability, we can now listen in on both DLCI 1023 (cisco LMI) and DLCI 0 (ANSI and ITU) simultaneously.

## Status Messages

One or more of the status requests will elicit a reply (status message) from the switch. The router will decode the format of the reply and configure itself automatically. If more than one reply is received, the router will configure itself with the type of the last received reply. This is to accommodate intelligent switches that can handle multiple formats simultaneously.

## LMI Autosense

If LMI autosense is unsuccessful, an intelligent retry scheme is built in. Every N391 interval (default is 60 seconds, which is 6 keep exchanges at 10 seconds each), LMI autosense will attempt to ascertain the LMI type. For more information about N391, see the **frame-relay lmi-n391dte** command in the “Frame Relay Commands” chapter of the *Wide-Area Networking Command Reference*.

The only visible indication to the user that LMI autosense is underway is when **debug frame lmi** is turned on. Every N391 interval, the user will now see three rapid status enquiries coming out of the serial interface. One in ANSI, one in ITU and one in cisco LMI-type.

## Configuration Options

No configuration options are provided; this is transparent to the user. You can turn off LMI autosense by explicitly configuring an LMI type. The LMI type must be written into NVRAM so that next time the router powers up, LMI autosense will be inactive. At the end of autoinstall, a **frame-relay lmi-type xxx** statement is included within the interface configuration. This configuration is not automatically written to NVRAM; you must explicitly write the configuration to NVRAM by using the **copy running-config** or **copy startup-config** commands.

### Explicitly Configure the LMI

Our Frame Relay software supports the industry-accepted standards for addressing the Local Management Interface (LMI), including the Cisco specification. If you want to configure the LMI and thus deactivate LMI autosense, complete the tasks in the following sections. The tasks in the first two sections are required if you choose to configure the LMI.

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

#### Set the LMI Type

If the router or access server is attached to a public data network (PDN), the LMI type must match the type used on the public network. Otherwise, the LMI type can be set to suit the needs of your private Frame Relay network.

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

Task	Command
Set the LMI type.	<b>frame-relay lmi-type</b> {ansi   cisco   q933a}
Exit configuration mode.	<b>end</b>
Write the LMI type to NVRAM.	<b>copy startup-config destination</b>

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 configure the LMI. By default, this interval is 10 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>keepalive number</b>

To disable keepalives on networks that don't utilize LMI, use the **no 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 Network-to-Network Interface (NNI) error threshold.	<b>frame-relay lmi-n392dce threshold</b>
Set the DCE and NNI monitored events count.	<b>frame-relay lmi-n393dce events</b>

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

See the “Frame Relay Commands” chapter in the *Wide-Area Networking Command Reference* for details about commands used to set the polling and timing intervals.

## Configure Frame Relay Switched Virtual Circuits

Access to Frame Relay networks is made through private leased lines at speeds ranging from 56 kbps to 45 Mbps. Frame Relay is a connection-oriented, packet-transfer mechanism that establishes virtual circuits between endpoints.

### Switched Virtual Circuits

Switched virtual circuits (SVCs) allow access through a Frame Relay network by setting up a path to the destination endpoints only when the need arises and tearing down the path when it is no longer needed.

SVCs can coexist with PVCs in the same sites and routers. For example, routers at remote branch offices might set up PVCs to the central headquarters for frequent communication, but set up SVCs with each other as needed for intermittent communication. As a result, any-to-any communication can be set up without any-to-any PVCs.

On SVCs, quality of service (QOS) elements can be specified on a call-by-call basis to request network resources.

SVC support is offered in the Enterprise image on Cisco platforms that include a serial or HSSI interface (Cisco 7000 series, Cisco 7500 series, Cisco 4500, Cisco 4700, Cisco 4000, Cisco 3000, and Cisco 2500 platforms).

You must have the following services before Frame Relay SVCs can operate:

- Frame Relay SVC support by the service provider—The service provider’s switch must be capable of supporting SVC operation.
- Physical loop connection—A leased line or dedicated line must exist between the router (DTE) and the local Frame Relay switch.

### SVC Operation

SVC operation requires that the Data Link layer (Layer 2) be set up, running ITU-T Q.922 Link Access Procedures to Frame mode bearer services (LAPF), prior to signalling for an SVC. Layer 2 sets itself up as soon as SVC support is enabled on the interface, if both the line and the line protocol are up. When the SVCs are configured and demand for a path occurs, the Q.933 signalling sequence is initiated. Once the SVC is set up, data transfer begins.

Q.922 provides a reliable link layer for Q.933 operation. All Q.933 call control information is transmitted over DLCI 0; this DLCI is also used for the management protocols specified in ANSI T1.617 Annex D or Q.933 Annex A.

You must enable SVC operation at the interface level. Once it is enabled at the interface level, it is enabled on any subinterfaces on that interface. One signalling channel, DLCI 0, is set up for the interface, and all SVCs are controlled from the physical interface.

### Enabling Frame Relay SVC Service

To enable Frame Relay SVC service and set up SVCs, complete the tasks in the following sections. The subinterface tasks are not required, but offer additional flexibility for SVC configuration and operation. The LAPF tasks are not required and not recommended unless you understand thoroughly the impacts on your network.

- Configure SVCs on a Physical Interface
- Configure SVCs on a Subinterface (optional)
- Configure a Map Class
- Configure a Map Group with E.164 or X.121 Addresses
- Associate the Map Class with Static Protocol Address Maps
- Configure LAPF Parameters (optional)

See the “SVC Configuration Examples” section at the end of this chapter.

### Configure SVCs on a Physical Interface

To enable SVC operation on a Frame Relay interface, perform the following tasks beginning in global configuration mode:

Task	Command
Specify the physical interface.	<b>interface</b> <i>type number</i>
Specify the interface IP address, if needed.	<b>ip address</b> <i>ip-address mask</i>
Enable Frame Relay encapsulation on the interface.	<b>encapsulation frame-relay</b>
Assign a map group to the interface.	<b>map-group</b> <i>group-name</i>
Enable Frame Relay SVC support on the interface.	<b>frame-relay svc</b>

Map-group details are specified with the **map-list** command.

### Configure SVCs on a Subinterface

To configure Frame Relay SVCs on a subinterface, perform all the tasks in the previous section, except assigning a the map group. After the physical interface is configured, complete the following tasks beginning in global configuration mode:

Task	Command
Specify a subinterface of the main interface configured for SVC operation.	<b>interface</b> <i>type number.subinterface-number</i> { <b>multipoint</b>   <b>point-to-point</b> }
Specify the subinterface IP address, if needed.	<b>ip address</b> <i>ip-address mask</i>
Assign a map group to the subinterface.	<b>map-group</b> <i>group-name</i>

## Configure a Map Class

To configure a map class, you can perform the following tasks. Only the first task is required.

- Specify the map class name.
- Specify a custom queue list for the map class.
- Specify a priority queue list for the map class.
- Enable BECN feedback to throttle the output rate on the SVC for the map class.
- Set nondefault QOS values for the map class.

You are not required to set the QOS values; default values are provided.

To configure a map class, perform the following tasks beginning in global configuration mode:

Task	Command
Specify the Frame Relay map class name and enter map class configuration mode.	<b>map-class frame-relay</b> <i>map-class-name</i>
Specify a custom queue list to be used for the map class.	<b>frame-relay custom-queue-list</b> <i>list-number</i>
Assign a priority queue to virtual circuits associated with the map class.	<b>frame-relay priority-group</b> <i>list-number</i>
Enable the type of BECN feedback to throttle the frame-transmission rate.	<b>frame-relay adaptive-shaping</b> [ <b>becn</b>   <b>foresight</b> ] <sup>1</sup>
Specify the inbound committed information rate (CIR).	<b>frame-relay cir in</b> <i>bps</i>
Specify the outbound committed information rate (CIR).	<b>frame-relay cir out</b> <i>bps</i>
Set the minimum acceptable incoming CIR.	<b>frame-relay mincir in</b> <i>bps</i> <sup>2</sup>
Set the minimum acceptable outgoing CIR.	<b>frame-relay mincir out</b> <i>bps</i> <sup>2</sup>
Set the incoming committed burst size (Bc).	<b>frame-relay bc in</b> <i>bits</i> <sup>2</sup>
Set the outgoing committed burst size (Bc).	<b>frame-relay bc out</b> <i>bits</i> <sup>2</sup>
Set the incoming excess burst size (Be).	<b>frame-relay be in</b> <i>bits</i> <sup>2</sup>
Set the outgoing excess burst size (Be).	<b>frame-relay be out</b> <i>bits</i> <sup>2</sup>
Set the idle timeout interval.	<b>frame-relay idle-timer</b> <i>duration</i> <sup>2</sup>

1. This command replaces the **frame-relay becn-response-enable** command, which will be removed in a future Cisco IOS release. If you use the **frame-relay becn-response-enable** command in scripts, you should replace it with the **frame-relay adaptive-shaping becn** command.

2. The **in** and **out** keywords are optional. Configuring the command without the in and out keywords will apply that value to both the incoming and outgoing traffic values for the SVC setup. For example, **frame-relay cir 56000** applies 56000 to both incoming and outgoing traffic values for setting up the SVC.

You can define multiple map classes. A map class is associated with a static map, not with the interface or subinterface itself. Because of the flexibility this association allows, you can define different map classes for different destinations.

## Configure a Map Group with E.164 or X.121 Addresses

After you have defined a map group for an interface, you can associate the map group with a specific source and destination address to be used. You can specify E.164 addresses or X.121 addresses for the source and destination. To specify the map group to be associated with a specific interface, perform the following task in global configuration mode:

Task	Command
Specify the map group to be associated with specific source address and destination address for the SVC.	<b>map-list</b> <i>group-name</i> <b>source-addr</b> { <b>e164</b>   <b>x121</b> } <i>source-address</i> <b>dest-addr</b> { <b>e164</b>   <b>x121</b> } <i>destination-address</i>

## Associate the Map Class with Static Protocol Address Maps

To define the protocol addresses under a **map-list** command and associate each protocol address with a specified map class, use the **class** command. Use this command for each protocol address to be associated with a map class. To associate a map class with a protocol address, perform the following task in map class configuration mode:

Task	Command
Specify a destination protocol address and a Frame Relay map class name from which to derive QOS information.	<i>protocol protocol-address</i> <b>class</b> <i>class-name</i> [ <b>ietf</b> ] [ <b>broadcast</b> [ <b>trigger</b> ]]

The **ietf** keyword specifies RFC 1490 encapsulation; the **broadcast** keyword specifies that broadcasts must be carried. The **trigger** keyword, which can be configured only if **broadcast** is also configured, enables a broadcast packet to trigger an SVC. If an SVC already exists that uses this map class, the SVC will carry the broadcast.

## Configure LAPF Parameters

Frame Relay Link Access Procedure for Frame Relay (LAPF) commands are used to tune Layer 2 system parameters to work well with the Frame Relay switch. Normally, you do not need to change the default settings.

However, if the Frame Relay network indicates that it does not support the Frame Reject frame (FRMR) at the LAPF Frame Reject procedure, complete the following task in interface configuration mode:

Task	Command
Select not to send FRMR frames at the LAPF Frame Reject procedure.	<b>no frame-relay lapf frmr</b>

By default, the Frame Reject frame is sent at the LAPF Frame Reject procedure.

---

**Note** Manipulation of Layer 2 parameters is not recommended if you do not know well the resulting functional change. For more information, refer to the ITU-T Q.922 specification for LAPF.

---

If you must change Layer 2 parameters for your network environment and you understand the resulting functional change, complete the following tasks as needed:

Task	Command
Set the LAPF window size <i>k</i> .	<b>frame-relay lapf k</b> <i>number</i>
Set the LAPF maximum retransmission count <i>N200</i> .	<b>frame-relay lapf n200</b> <i>retries</i>
Set the maximum length of the Information field of the LAPF I frame <i>N201</i> .	<b>frame-relay lapf n201</b> <i>number</i>
Set the LAPF retransmission timer value <i>T200</i> .	<b>frame-relay lapf t200</b> <i>tenths-of-a-second</i>
Set the LAPF link idle timer value <i>T203</i> of DLCI 0.	<b>frame-relay lapf t203</b> <i>seconds</i>

## Configure Frame Relay Traffic Shaping

The following Frame Relay traffic shaping capabilities were introduced with Cisco IOS Release 11.2:

- Rate Enforcement on a Per-Virtual Circuit Basis—The peak rate for outbound traffic. The value can be set to match CIR or another value.
- Dynamic Traffic Throttling on a Per-Virtual Circuit Basis—When BECN packets indicate congestion on the network, the outbound traffic rate is automatically stepped down; when congestion eases, the outbound traffic rate is increased. This feature is enable by default.
- Enhanced Queuing Support on a Per-Virtual Circuit Basis—Either custom queuing or priority queuing can be configured for individual virtual circuits.

---

**Note** Frame Relay traffic shaping is not effective for Layer 2 PVC switching using the **frame-relay route** command.

---

### Virtual Circuits for Different Types of Traffic

By defining separate virtual circuits for different types of traffic and specifying queuing and an outbound traffic rate for each virtual circuit, you can provide guaranteed bandwidth for each type of traffic. By specifying different traffic rates for different virtual circuits over the same line, you can perform virtual time division multiplexing. By throttling outbound traffic from high-speed lines in central offices to lower-speed lines in remote locations, you can ease congestion and data loss in the network; enhanced queuing also prevents congestion-caused data loss.

### Traffic Shaping Tasks

Traffic shaping applies to both PVCs and SVCs. For information about creating and configuring SVCs, see the “Configure Frame Relay Switched Virtual Circuits” section of this chapter.

To configure Frame Relay traffic shaping, perform the tasks in the following sections:

- Enable Frame Relay Encapsulation on an Interface (earlier in this chapter)
- Enable Frame Relay Traffic Shaping on the Interface
- Enable Enhanced Local Management Interface
- Specify a Traffic Shaping Map Class for the Interface
- Define a Map Class with Queuing and Traffic Shaping Parameters

- Define Access Lists
- Define Priority Queue Lists for the Map Class
- Define Custom Queue Lists for the Map Class

### Enable Frame Relay Traffic Shaping on the Interface

Enabling Frame Relay traffic shaping on an interface enables both traffic shaping and per-virtual circuit queuing on all the interface's PVCs and SVCs. Traffic shaping enables the router to control the circuit's output rate and react to congestion notification information if also configured.

To enable Frame Relay traffic shaping on the specified interface, complete the following task in interface configuration mode:

Task	Command
Enable Frame Relay traffic shaping and per-virtual circuit queuing.	<b>frame-relay traffic-shaping</b>

### Understand Frame Relay Router ForeSight

ForeSight is the network traffic control software used in Cisco StrataCom switches. The Cisco StrataCom Frame Relay switch can extend ForeSight messages over a User-to-Network Interface (UNI), passing the backward congestion notification for virtual circuits.

The Router ForeSight feature allows Cisco Frame Relay routers to process and react to ForeSight messages and adjust virtual circuit level traffic shaping in a timely manner.

Router ForeSight must be configured explicitly on both the Cisco router and the Cisco StrataCom switch. Router ForeSight is enabled on the Cisco router when Frame Relay traffic shaping is configured. However, the router's response to ForeSight is not applied to any VC until the **frame-relay adaptive-shaping foresight** command is added to the VCs map-class. When ForeSight is enabled on the StrataCom switch, the switch will periodically send out a ForeSight message based on the time value configured. The time interval can range from 40 to 5000 milliseconds.

When a Cisco router receives a ForeSight message indicating that certain Data Link Connection Identifiers (DLCIs) are experiencing congestion, the Cisco router reacts by activating its traffic shaping function to slow down the output rate. The router reacts as it would if it were to detect the congestion by receiving a packet with the backward explicit congestion notification (BECN) bit set.

### Congestion Notification Methods

The difference between the BECN and ForeSight congestion notification methods is that BECN requires a user packet to be sent in the direction of the congested DLCI to convey the signal. The sending of user packets is not predictable and, therefore, not reliable as a notification mechanism. Rather than waiting for user packets to provide the congestion notification, timed ForeSight messages guarantee that the router receives notification before congestion becomes a problem. Traffic can be slowed down in the direction of the congested DLCI.

### Router ForeSight Prerequisites

For Router ForeSight to work, the following conditions must exist on the Cisco router:

- Frame Relay traffic shaping must be enabled on the interface.
- The traffic shaping for a circuit is adapted to ForeSight.

The following additional condition must exist on the Cisco StrataCom switch:

- The UNI connecting to the router is Consolidated Link Layer Management (CLLM) enabled, with the proper time interval specified.

Frame Relay Router ForeSight is enabled automatically when you use the **frame-relay traffic-shaping** command. However, you must issue the **map-class frame-relay** command and the **frame-relay adaptive-shaping foresight** command before the router will respond to ForeSight and apply the traffic shaping effect on a specific interface, subinterface, or virtual circuit.

## Enable Enhanced Local Management Interface

When used in conjunction with traffic shaping, the router can respond to changes in the network dynamically. This feature allows the router to learn QOS parameters from the Cisco StrataCom switch and use them for traffic shaping, configuration, or management purposes.

Enhanced Local Management Interface also simplifies the process of configuring traffic shaping on the router. Previously, users had to configure traffic shaping rate enforcement values, possibly for every virtual circuit. Enabling Enhanced Local Management Interface reduces chances of specifying inconsistent or incorrect values when configuring the router.

To enable the Enhanced Local Management Interface feature, you must configure it on the main interface. Perform the following task in interface configuration mode:

Task	Command
Specify the physical interface.	<b>interface</b> <i>type number</i>
Enable Frame Relay encapsulation on the interface.	<b>encapsulation frame-relay</b> [cisco   ietf]
Enable the Enhanced Local Management Interface feature.	<b>frame-relay qos-autosense</b>

**Note** Enhanced Local Management Interface enables automated exchange of Frame Relay QOS parameter information between the Cisco router and the Cisco StrataCom switch. Routers can base congestion management and prioritization decisions on known QOS values, such as the Committed Information Rate (CIR), Committed Burst Size (Bc), and Excess Burst Size (Be). The router senses Quality of Service (QOS) values from the switch and can be configured to use those values in traffic shaping. This enhancement works between Cisco routers and Cisco StrataCom switches (BPX/AXIS and IGX platforms).

It is not necessary to configure traffic shaping on the interface to enable Enhanced Local Management Interface. You might want to enable it to know the values being used by the switch. If you want the router to respond to the QOS information received from the switch by adjusting the output rate, you must configure traffic shaping on the interface. To configure traffic shaping, use the **frame-relay traffic-shaping** command in interface configuration mode

For an example of how to configure a Frame Relay interface with QOS autosense enabled, see the “Enhanced Local Management Interface Example” section later in this chapter.

## Specify a Traffic Shaping Map Class for the Interface

If you specify a Frame Relay map class for a main interface, all the virtual circuits on its subinterfaces inherit all the traffic shaping parameters defined for the class.

To specify a map class for the specified interface, complete the following tasks in beginning interface configuration mode:

Task	Command
Specify a Frame Relay map class for the interface.	<b>frame-relay class</b> <i>map-class-name</i>

You can override the default for a specific DLCI on a specific subinterface by using the **class** virtual circuit configuration command to assign the DLCI explicitly to a different class. See the “Configure Frame Relay Subinterfaces” section for information about setting up subinterfaces. For an example of assigning some subinterface DLCIs to the default class and assigning others explicitly to a different class, see the “Frame Relay Traffic Shaping Examples” section.

### Define a Map Class with Queuing and Traffic Shaping Parameters

When you define a map class for Frame Relay, you can define the average and peak rates (in bits per second) allowed on virtual circuits associated with the map class. You can also, optionally, specify *either* a custom queue-list or a priority queue-group to use on virtual circuits associated with the map class.

To define a map class, complete the following tasks beginning in global configuration mode:

Task	Command
Specify a map class to define.	<b>map-class frame-relay</b> <i>map-class-name</i>
Define the traffic rate for the map class.	<b>frame-relay traffic-rate</b> <i>average</i> [ <i>peak</i> ]
Specify a custom queue-list.	<b>frame-relay custom-queue-list</b> <i>number</i>
Specify a priority queue-list.	<b>frame-relay priority-group</b> <i>number</i>
Select either BECN or ForeSight as the congestion backward-notification mechanism to which traffic shaping will adapt.	<b>frame-relay adaptive-shaping</b> { <b>becn</b>   <b>foresight</b> } <sup>1</sup>

1. This command replaces the **frame-relay becn-response-enable** command, which will be removed in a future Cisco IOS release. If you use the **frame-relay becn-response-enable** command in scripts, you should replace it with the **frame-relay adaptive-shaping** command.

### Define Access Lists

You can specify access lists and associate them with the custom queue-list defined for any map class. The list number specified in the access list and the custom queue list tie them together.

See the appropriate protocol chapters for information about defining access lists for the protocols you want to transmit on the Frame Relay network.

### Define Priority Queue Lists for the Map Class

You can define a priority list for a protocol and you can also define a default priority list. The number used for a specific priority list ties the list to the Frame Relay priority group defined for a specified map class.

For example, if you enter the **frame relay priority-group 2** command for the map class *fast\_vcs* and then you enter the **priority-list 2 protocol decnet high** command, that priority list is used for the *fast\_vcs* map class. The average and peak traffic rates defined for the *fast\_vcs* map class are used for DECnet traffic.

## Define Custom Queue Lists for the Map Class

You can define queue list for a protocol and a default queue list. You can also specify the maximum number of bytes to be transmitted in any cycle. The number used for a specific queue list ties the list to the Frame Relay custom-queue list defined for a specified map class.

For example, if you enter the **frame relay custom-queue-list 1** command for the map class *slow\_vcs* and then you enter the **queue-list 1 protocol ip list 100** command, that queue list is used for the *slow\_vcs* map class; **access-list 100** definition is also used for that map class and queue. The average and peak traffic rates defined for the *slow\_vcs* map class are used for IP traffic that meets the **access list 100** criteria.

## Customize Frame Relay for Your Network

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

- Configure 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 Payload Compression
- Configure Standard-Based FRF.9 Compression
- Configure TCP/IP Header Compression
- Configure Real-Time Header Compression with Frame Relay Encapsulation
- Configure Discard Eligibility
- Configure DLCI Priority Levels

## Configure Frame Relay Subinterfaces

To understand and define Frame Relay Subinterfaces, read “Understand Frame Relay Subinterfaces.” To define the Frame Relay subinterface, perform the required tasks in the following sections:

- Define Frame Relay Subinterfaces
- Define Subinterface Addressing

After the subinterface is defined, you can also perform the optional tasks in the following sections:

- Configure Transparent Bridging for Frame Relay
- Configure a Backup Interface for a Subinterface

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

## Understand Frame Relay Subinterfaces

Frame Relay subinterfaces provide a mechanism for supporting partially meshed Frame Relay networks. 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. Transitivity is true on LANs, but not on Frame Relay networks unless A is directly connected to C.

Additionally, certain protocols such as AppleTalk and transparent bridging cannot 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.

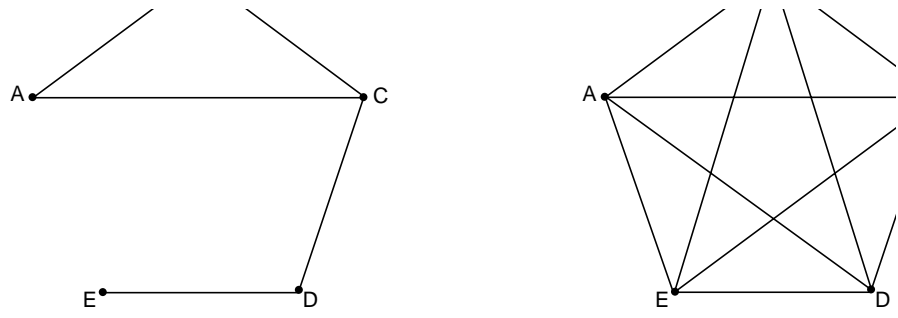
Configuring Frame Relay subinterfaces ensures that a *single physical interface* is treated as *multiple virtual interfaces*. This capability allows us to overcome split horizon rules. Packets received on one virtual interface can now be forwarded out another virtual interface, even if they are configured on the same physical interface.

Subinterfaces address the limitations of Frame Relay networks 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 16) 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 network 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 network work fully is to create a fully meshed network (Network B), but doing so requires a large number of PVCs, which may not be economically feasible.

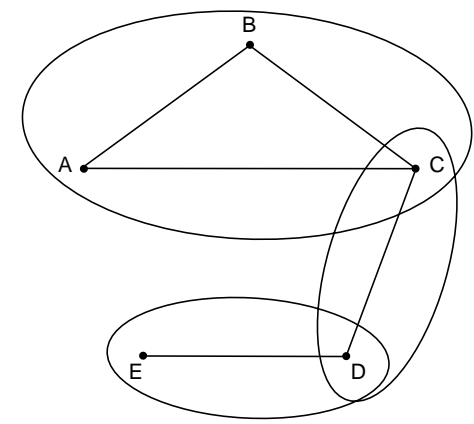
Using subinterfaces, you can subdivide the Frame Relay network into three smaller subnetworks (Network C) with separate network numbers. Nodes A, B, and C are connected to a fully meshed network, and nodes C and D, as well as nodes D and E are connected via point-to-point networks. In this configuration,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

**Figure 16 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 network with full connectivity



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

### Define Frame Relay Subinterfaces

To configure subinterfaces on a Frame Relay network, perform the following tasks beginning in global configuration mode:

Task	Command
<b>Step 1</b> Specify an interface.	<b>interface</b> <i>type number</i>
<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</b> <i>type number.subinterface-number</i> { <b>multipoint</b>   <b>point-to-point</b> }

Subinterfaces can be configured for multipoint or point-to-point communication. (There is no default.)

### Define Subinterface Addressing

For point-to-point subinterfaces, the destination is presumed to be known and is identified or implied in the **frame-relay interface-dlci** command. For multipoint subinterfaces, the destinations can be dynamically resolved through the use of Frame Relay Inverse ARP or can be statically mapped through the use of the **frame-relay map** command.

### Addressing on Point-to-Point Subinterfaces

If you specified a point-to-point subinterface in Step 3 of the previous procedure, perform the following task in interface configuration mode:

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

---

**Note** This command is typically used on subinterfaces; however, it can also be applied to main interfaces. The **frame-relay interface-dlci** command is used to enable routing protocols on main interfaces that are configured to use Inverse ARP. This command is also helpful for assigning a specific class to a single PVC on a multipoint subinterface.

---

For an explanation of the many available options, refer to this command in the *Wide-Area Networking Command Reference*. For an example of how to associate a DLCI with a subinterface, see the section “Subinterface Examples” later in this chapter.

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 or access server. Instead, you can simply avoid using that subinterface number and use a different subinterface number instead.

### Addressing on Multipoint Subinterfaces

If you specified a multipoint subinterface in Step 3 under “Define Frame Relay Subinterfaces,” perform the tasks in one or both of the following sections:

- Accept Inverse ARP for Dynamic Address Mapping on Multipoint Subinterfaces
- Configure Static Address Mapping on Multipoint Subinterfaces

You can configure some protocols for dynamic address mapping and others for static address mapping.

#### Accept Inverse ARP for Dynamic Address Mapping on Multipoint Subinterfaces

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

Since the physical interface is now configured as multiple subinterfaces, you must provide information that distinguishes a subinterface from the physical interface and associates a specific subinterface with a specific DLCI.

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

Task	Command
Associate a specified multipoint subinterface with a DLCI.	<b>frame-relay interface-dlci</b> <i>dlci</i>

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.

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

For an example of configuring Frame Relay multipoint subinterfaces with dynamic address mapping, see the “Frame Relay Multipoint Subinterface with Dynamic Addressing Example” section.

### Configure Static Address Mapping on Multipoint Subinterfaces

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 that 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>ietf</b> ] <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**

The **broadcast** keyword is required for routing protocols such as OSI protocols and the Open Shortest Path First (OSPF) protocol. See the **frame-relay map** command description in the *Wide-Area Networking Command Reference* 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 Transparent Bridging for Frame Relay

Transparent bridging for Frame Relay encapsulated serial and HSSI interfaces is supported on our routers. Transparent bridging for Frame Relay encapsulated serial interfaces is supported on our access servers.

You can configure transparent bridging for point-to-point or point-to-multipoint subinterfaces.

---

**Note** All PVCs configured on a subinterface belong to the same bridge group.

---

#### Point-to-Point Subinterfaces

To configure transparent bridging for point-to-point subinterfaces, complete the following tasks beginning in global configuration mode:

Task	Command
<b>Step 1</b> Specify an interface.	<b>interface</b> <i>type number</i>
<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</b> <i>type number:subinterface-number</i> <b>point-to-point</b>
<b>Step 4</b> Associate a DLCI with the subinterface.	<b>frame-relay interface-dlci</b> <i>dlci</i> [ <i>option</i> ]
<b>Step 5</b> Associate the subinterface with a bridge group.	<b>bridge-group</b> <i>bridge-group</i>

#### Point-to-Multipoint Interfaces

To configure transparent bridging for point-to-multipoint subinterfaces, complete the following tasks beginning in global configuration mode:

Task	Command
<b>Step 1</b> Specify an interface.	<b>interface</b> <i>type number</i>
<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</b> <i>type number:subinterface-number</i> <b>multipoint</b>
<b>Step 4</b> Define the mapping between a next hop protocol address and the DLCI used to connect to the address.	<b>frame-relay map bridge</b> <i>dlci</i> [ <b>broadcast</b> ] [ <b>ietf</b> ]
<b>Step 5</b> Associate the subinterface with a bridge group.	<b>bridge-group</b> <i>bridge-group</i>

## Configure a Backup Interface for a Subinterface

Both point-to-point and multipoint Frame Relay subinterfaces can be configured with a backup interface. This approach 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 main 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 main interface is up, or if the interface is down and does not have a backup interface defined. If a subinterface fails while its backup interface is in use, and the main interface goes down, the backup subinterface remains connected.

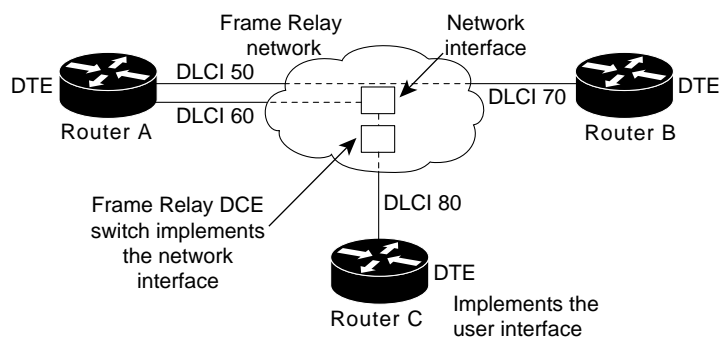
To configure a backup interface for a Frame Relay subinterface, perform the following tasks, beginning in global configuration mode:

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

## 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. You perform the switching by configuring your router or access server as a Frame Relay network. There are two parts to a Frame Relay network: a Frame Relay DTE (the router or access server) and a Frame Relay DCE switch. Figure 17 illustrates this concept.

**Figure 17 Frame Relay Switched Network**



In Figure 17, 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 devices 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

### Enable Frame Relay Switching

You must enable packet switching before you can configure it on a Frame Relay DTE or 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 or a 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 in-dlci out-interface out-dlci</b>

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 or access server 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 under 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 or disable it, perform one of the following tasks 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 many DLCIs terminate in a single router or access server that 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 Payload Compression

You can configure payload compression on point-to-point or multipoint interfaces or subinterfaces. Payload compression uses the stac method to predict what the next character in the frame will be. Because the prediction is done packet-by-packet, the dictionary is not conserved across packet boundaries.

Payload compression on each virtual circuit consumes approximately 40 kilobytes for dictionary memory.

To configure payload compression on a specified multipoint interface or subinterface, complete the following task in interface configuration mode:

Task	Command
Enable payload compression on a multipoint interface.	<b>frame-relay map</b> <i>protocol protocol-address dlci</i> <b>payload-compress packet-by-packet</b>

To configure payload compression on a specified point-to-point interface or subinterface, complete the following task in interface configuration mode:

Task	Command
Enable payload compression on a point-to-point interface.	<b>frame-relay payload-compress packet-by-packet</b>

## Configure Standard-Based FRF.9 Compression

Frame Relay compression can now occur on the VIP board, on the Compression Service Adapter (CSA), or on the main CPU of the router. FRF9 is standard-based and, therefore, provides multivendor compatibility. FRF.9 compression uses higher compression ratios, allowing more data to be compressed for faster transmission.

The CSA hardware has been in use on the Cisco 7200 series and Cisco 7500 series platforms, but it has had no support for Frame Relay compression. FRF.9 compression provides the ability to maintain multiple decompression/compression histories on a per-DLCI basis.

The CSA can be used in the Cisco 7200 series or in the second-generation Versatile Interface Processor (VIP2) in all Cisco 7500 series routers

The specific VIP2 model required for the CSA is VIP2-40, which has 2 MB of SRAM and 32 MB of DRAM.

## How the Router Selects the Compression Method

The router enables compression in the following order:

- 1 If the router contains a compression service adapter, compression is performed in the CSA hardware (hardware compression).
- 2 If the CSA is not available, compression is performed in the software installed on the VIP2 card (distributed compression).
- 3 If the VIP2 card is not available, compression is performed in the router's main processor (software compression).

## Configure FRF.9 Compression Using Map Statements

You can control where you want compression to occur by specifying a specific interface. To enable FRF.9 compression on a specific CSA, VIP CPU, or host CPU, perform the following tasks beginning in global configuration mode:

Task	Command
<b>Step 1</b> Specify the interface.	<b>interface</b> <i>type number</i>
<b>Step 2</b> Specify Frame Relay as the encapsulation type.	<b>encapsulation frame-relay</b>
<b>Step 3</b> Enable FRF.9 compression.	<b>frame-relay map payload-compress frf9 stac</b> [ <i>csa csa_number</i>   <b>distributed</b>   <b>software</b> ]

## Configure FRF.9 Compression on the Subinterface

To configure FRF.9 compression on the subinterface, perform the following tasks beginning in global configuration mode:

Task	Command
<b>Step 1</b> Specify the subinterface type and number.	<b>interface</b> <i>type number</i>
<b>Step 2</b> Specify Frame Relay as the encapsulation type.	<b>encapsulation frame-relay</b>
<b>Step 3</b> Enable FRF.9 compression.	<b>frame-relay payload-compress frf9 stac</b> [ <i>csa csa_number</i>   <b>distributed</b>   <b>software</b> ]

## 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. Reconstructing a smaller header that identifies the connection and indicates the fields that changed and the amount of change reduces the number of bytes 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</b> <i>ip-address</i> <i>dldi</i> [ <b>broadcast</b> ] <b>cisco tcp header-compression</b> { <b>active</b>   <b>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 “FRF.9 Compression Configuration 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 “FRF.9 Compression Configuration 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/IP 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/IP header compression on a specified Frame Relay IP map.	<b>frame-relay map ip <i>ip-address</i> <i>dci</i> nocompress tcp header-compression</b>

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

### Configure Real-Time Header Compression with Frame Relay Encapsulation

Real-time Transport Protocol (RTP) is a protocol used for carrying packetized audio and video traffic over an IP network. RTP is described in RFC 1889. RTP is not intended for data traffic, which uses TCP or UDP. RTP provides end-to-end network transport functions intended for applications with real-time requirements, such as audio, video, or simulation data over multicast or unicast network services.

Although RTP header compression uses Frame Relay encapsulation, this feature is described in the chapter “*Configuring IP Multicast Routing.*” The commands for configuring this feature appear in the chapter “*IP Multicast Routing Commands.*”

### 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</b> <i>list-number</i> { <b>protocol</b> <i>protocol</i>   <b>interface</b> <i>type number</i> } <i>characteristic</i>

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 User Datagram Protocol (UDP) port, an access list number, or a packet size. See the **frame-relay de-list** command in the *Wide-Area Networking Command Reference* 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</i> <i>dlci</i>

### Configure DLCI Priority Levels

DLCI priority levels allow you to separate different types of traffic and can provide a traffic management tool for congestion problems caused by following situations:

- Mixing batch and interactive traffic over the same DLCI.
- Traffic from sites with high-speed access being queued at destination sites with lower speed access.

Before you configure the DLCI priority levels, complete the following tasks:

- Define a global priority list.
- Enable Frame Relay encapsulation, as described earlier in this chapter.
- Define static or dynamic address mapping, as described earlier in this chapter.

Make sure that you define each of the DLCIs to which you intend to apply levels. You can associate priority-level DLCIs with subinterfaces.

- Configure the LMI, as described earlier in this chapter.

---

**Note** DLCI priority levels provide a way to define multiple parallel DLCIs for different types of traffic. DLCI priority levels do not assign priority queues within the router or access server; in fact, they are independent of the device's priority queues. However, if you enable queuing and use the same DLCIs for queuing, then high-priority DLCIs can be put into high-priority queues.

---

To configure DLCI priority levels, perform the following task in interface configuration mode:

Task	Command
Enable multiple parallel DLCIs for different types of Frame Relay traffic, associate specified DLCIs with the same group, and define their levels.	<b>frame-relay priority-dlci-group</b> <i>group-number high-dlci medium-dlci normal-dlci low-dlci</i>

---

**Note** If you do not explicitly specify a DLCI for each of the priority levels, the last DLCI specified in the command line is used as the value of the remaining arguments. At a minimum, you must configure the high-priority and the medium-priority DLCIs.

---

## Monitor and Maintain 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 type number</b>
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>
Display information about the status of LAPF.	<b>show frame-relay lapf</b>
Display all the SVCs under a specified map list.	<b>show frame-relay svc maplist</b>

---

## 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
- SVC Configuration Examples
- Frame Relay Traffic Shaping Examples
- Configuration Providing Backward Compatibility Example

- Booting from a Network Server over Frame Relay Examples
- Frame Relay Switching Examples
- FRF.9 Compression Configuration 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 example illustrates 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 2

```
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
```

### Frame Relay Multipoint Subinterface with Dynamic Addressing Example

The following example configures two multipoint subinterfaces for dynamic address resolution. Each subinterface is provided with an individual protocol address and subnet mask, and the **frame-relay interface-dlci** command associates the subinterface with a specified DLCI. Addresses of remote destinations for each multipoint subinterface will be resolved dynamically.

```
interface Serial0
  no ip address
  encapsulation frame-relay
  frame-relay lmi-type ansi
!
interface Serial0.103 multipoint
  ip address 172.21.177.18 255.255.255.0
  frame-relay interface-dlci 300
!
interface Serial0.104 multipoint
  ip address 172.21.178.18 255.255.255.0
  frame-relay interface-dlci 400
```

### IPX Routes over Frame Relay Subinterfaces Examples

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
  interface serial 0.2 multipoint
  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
 encapsulation frame-relay
interface serial 0.1 point-to-point
 bridge-group 1
 frame-relay interface-dlci 42
interface serial 0.2 point-to-point
 bridge-group 1
 frame-relay interface-dlci 64
```

```
interface serial 0.3 point-to-point
  bridge-group 1
  frame-relay interface-dlci 73
```

## SVC Configuration Examples

The following examples provide SVC configuration examples for interfaces and subinterfaces.

### Interface Example

The following example configures a physical interface, applies a map-group to the physical interface, and then defines the map-group.

```
interface serial 0
  ip address 172.10.8.6
  encapsulation frame-relay
  map-group bermuda
  frame-relay lmi-type q933a
  frame-relay svc
!
map-list bermuda source-addr E164 123456 dest-addr E164 654321
  ip 131.108.177.100 class hawaii
  appletalk 1000.2 class rainbow
!
map-class frame-relay rainbow
  frame-relay idle-timer 60
!
map-class frame-relay hawaii
  frame-relay cir in 64000
  frame-relay cir out 64000
```

### Subinterface Example

The following example configures a point-to-point interface for SVC operation. This example assumes that the main serial 0 interface has been configured for signalling, and that SVC operation has been enabled on the main interface.

```
int s 0.1 point-point
! Define the map-group; details are specified under the map-list holiday command.
map-group holiday
!
! Associate the map-group with a specific source and destination.
map-list holiday local-addr X121 <X121-addr> dest-addr E164 <E164-addr>
! Specify destination protocol addresses for a map-class.
  ip 131.108.177.100 class hawaii IETF
  appletalk 1000.2 class rainbow IETF broadcast
!
! Define a map class and its QOS settings.
map-class hawaii
  frame-relay cir in 2000000
  frame-relay cir out 56000
  frame-relay be 9000
!
! Define another map class and its QOS settings.
map-class rainbow
  frame-relay cir in 64000
  frame-relay idle-timer 2000
```

## Frame Relay Traffic Shaping Examples

The following sections provide examples of Frame Relay traffic shaping.

### Traffic Shaping with Three Point-to-Point Subinterfaces Example

In this example, the virtual circuits on subinterfaces Serial0.1 and Serial0.2 inherit class parameters from the main interface, namely those defined in *slow\_vcs*, but the virtual circuit defined on subinterface Serial0.2 (DLCI 102) is specifically configured to use map class *fast\_vcs*.

Map class *slow\_vcs* uses a peak rate of 9600 and average rate of 4800 bps. Because BECN feedback is enabled by default, the output rate will be cut back as low as 4800 bps in response to received BECNs. This map class is configured to use custom queuing using queue-list 1. In this example, queue-list 1 has 3 queues, with the first two being controlled by access lists 100 and 115.

Map class *fast\_vcs* uses a peak rate of 64000 and average rate of 16000 bps. Because BECN feedback is enabled by default, the output rate will be cut back as low as 4800 bps in response to received BECNs. This map class is configured to use priority-queuing using priority-group 2.

```

interface Serial0
  no ip address
  encapsulation frame-relay
  frame-relay lmi-type ansi
  frame-relay traffic-shaping
  frame-relay class slow_vcs
!
interface Serial0.1 point-to-point
  ip address 10.128.30.1 255.255.255.248
  ip ospf cost 200
  bandwidth 10
  frame-relay interface-dlci 101
!
interface Serial0.2 point-to-point
  ip address 10.128.30.9 255.255.255.248
  ip ospf cost 400
  bandwidth 10
  frame-relay interface-dlci 102
    class fast_vcs
!
interface Serial0.3 point-to-point
  ip address 10.128.30.17 255.255.255.248
  ip ospf cost 200
  bandwidth 10
  frame-relay interface-dlci 103
!
map-class frame-relay slow_vcs
  frame-relay traffic-rate 4800 9600
  frame-relay custom-queue-list 1
!
map-class frame-relay fast_vcs
  frame-relay traffic-rate 16000 64000
  frame-relay priority-group 2
!
access-list 100 permit tcp any any eq 2065
access-list 115 permit tcp any any eq 256
!
priority-list 2 protocol decnet high
priority-list 2 ip normal
priority-list 2 default medium
!
queue-list 1 protocol ip 1 list 100
queue-list 1 protocol ip 2 list 115
queue-list 1 default 3

```

```
queue-list 1 queue 1 byte-count 1600 limit 200
queue-list 1 queue 2 byte-count 600 limit 200
queue-list 1 queue 3 byte-count 500 limit 200
```

### Traffic Shaping with Router ForeSight Example

The following example illustrates a router configuration with traffic shaping enabled. DLCIs 100 and 101 on subinterfaces Serial 13.2 and Serial 13.3 inherit class parameters from the main interface. The traffic shaping for these two VCs will be adaptive to the ForeSight notification.

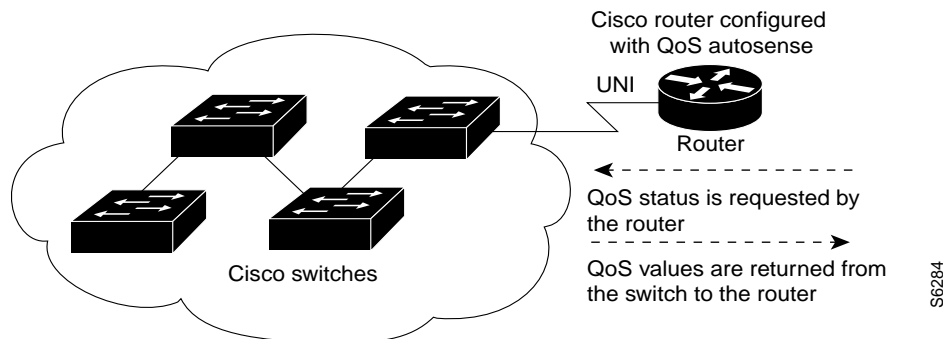
For Serial 0, the output rate for DLCI 103 will not be affected by the Router ForeSight function.

```
interface Serial0
  no ip address
  encapsulation frame-relay
  frame-relay lmi-type ansi
  frame-relay traffic-shaping
!
interface Serial0.2 point-to-point
  ip address 10.128.30.17 255.255.255.248
  frame-relay interface-dlci 102
  class fast_vcs
!
interface Serial0.3 point-to-point
  ip address 10.128.30.5 255.255.255.248
  ip ospf cost 200
  frame-relay interface-dlci 103
  class slow_vcs
!
interface serial 3
  no ip address
  encapsulation frame-relay
  frame-relay traffic-shaping
  frame-relay class fast_vcs
!
interface Serial3.2 multipoint
  ip address 100.120.20.13 255.255.255.248
  frame-relay map ip 100.120.20.6 16 ietf broadcast
!
interface Serial3.3 point-to-point
  ip address 100.120.10.13 255.255.255.248
  frame-relay interface-dlci 101
!
map-class frame-relay slow_vcs
  frame-relay adaptive-shaping becn
  frame-relay traffic-rate 4800 9600
!
map-class frame-relay fast_vcs
  frame-relay adaptive-shaping foresight
  frame-relay traffic-rate 16000 64000
  frame-relay cir 56000
  frame-relay bc 64000
```

### Enhanced Local Management Interface Example

Figure 18 illustrates a Cisco StrataCom switch and a Cisco router, both configured with the Enhanced Local Management Interface feature enabled. The switch sends QOS information to the router, which uses it for traffic rate enforcement.

**Figure 18 Enhanced Local Management Interface—Sent between the Cisco StrataCom Switch and the Cisco Router**



This configuration example shows a Frame-Relay interface enabled with QoS autosense. The router receives messages from the Cisco StrataCom switch, which is also configured with QoS autosense enabled. When Enhanced Local Management Interface is configured in conjunction with traffic shaping, the router will receive congestion information through BECN or Router ForeSight congestion signaling and reduce its output rate to the value specified in the traffic shaping configuration.

```
interface serial0
  no ip address
  encapsulation frame-relay
  frame-relay lmi-type ansi
  frame-relay traffic-shaping
  frame-relay qos-autosense
!
interface serial0.1 point-to-point
  no ip address
  frame-relay interface-dlci 101
```

## 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 Trivial File Transfer Protocol (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. To boot over Frame Relay, you must explicitly give the address of the network server to boot from, 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 must 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 network server) to the router booting over Frame Relay. Here, 101 is a DLCI of the router being booted.

## Frame Relay Switching Examples

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

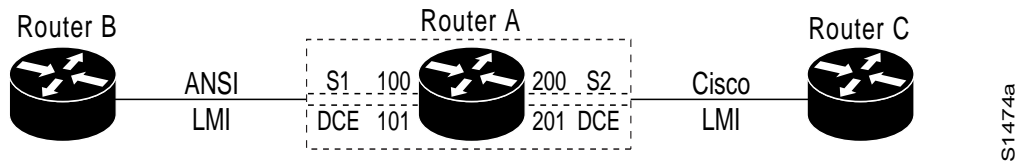
- **PVC Switching Configuration Example**—In this example, one router has two interfaces configured as DCEs; the router switches frames from the incoming interface to the outgoing interface on the basis of the DLCI alone.
- **Pure Frame Relay DCE Example**—In this example, a Frame Relay network is set 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 19).

Figure 19 PVC Switching Configuration



### Configuration for Router A

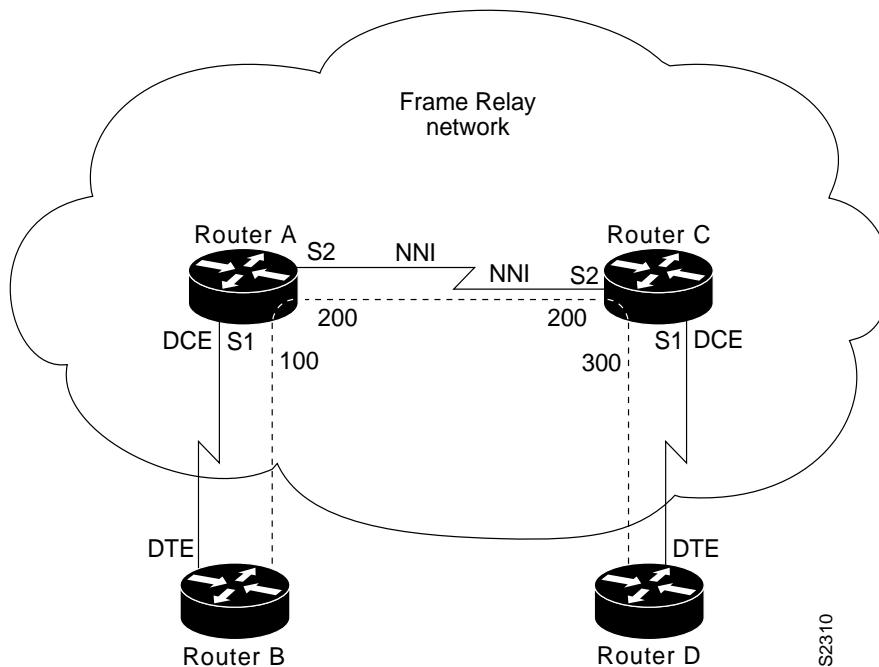
```

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 20).

Figure 20 Frame Relay DCE Configuration



Configuration for Router A

```

frame-relay switching
!
interface ethernet 0
no ip address
shutdown :Interfaces not in use may be shut down; shut down is not required.
!
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

```

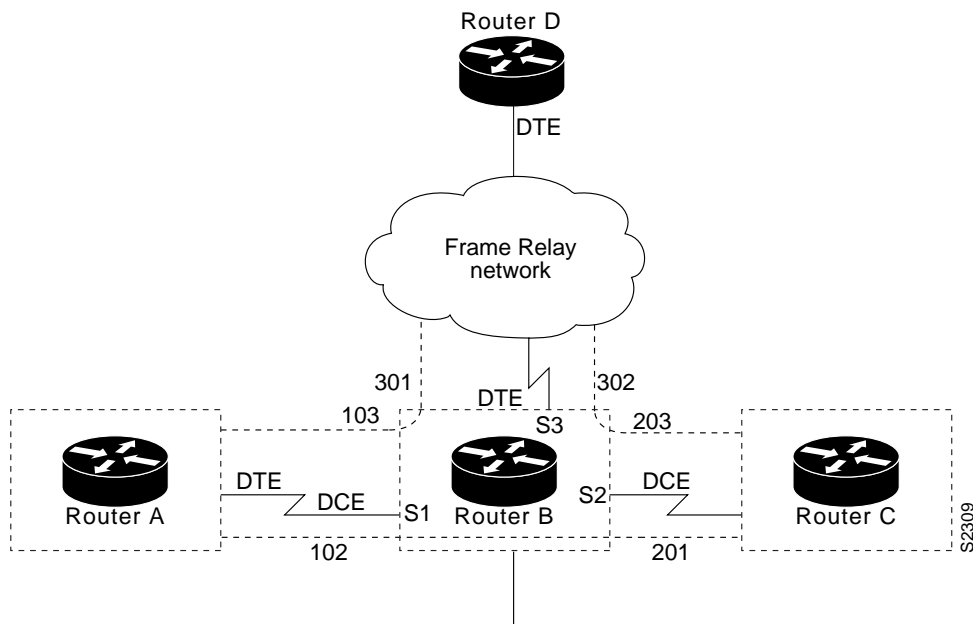
frame-relay switching
!
interface ethernet 0
no ip address
shutdown :Interfaces not in use may be shut down; shut down is not required.
!
interface ethernet1
no ip address
shutdown
!
interface ethernet 2
no ip address
shutdown
!
interface ethernet 3
no ip address
shutdown
!
interface serial 0
ip address 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 21).

**Figure 21 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 :Interfaces not in use may be shut down; shut down is not required.
!
interface serial 1
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
    
```

```

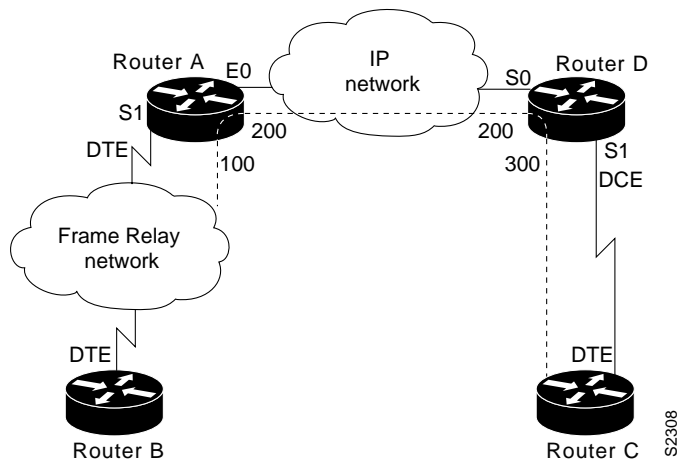
frame-relay route 102 interface serial 2 201
frame-relay route 103 interface serial 3 301
!
interface serial 2
no ip address
encapsulation frame-relay
frame-relay intf-type dce
frame-relay route 201 interface serial 1 102
frame-relay route 203 interface serial 3 302
!
interface serial 3
ip address 131.108.111.231
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

You can achieve switching over an IP tunnel by creating a point-to-point tunnel across the internetwork over which PVC switching can take place (see Figure 22).

**Figure 22** Frame Relay Switch over IP Tunnel



The following configurations illustrate how to create the IP network depicted in Figure 22.

#### 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 : Interfaces not in use may be shut down; shutdown is not required.
!
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
```

## FRF.9 Compression Configuration Examples

These examples show how to configure FRF.9 compression. The first example shows FRF.9 compression configuration using the **frame-relay map** command.

We recommend that you shut down the interface or subinterface prior to adding or changing compression techniques. Although this is not required, shutting down the interface ensures the interface is reset for the new data structures.

```
interface Serial2/0/1
  ip address 172.16.1.4 255.255.255.0
  no ip route-cache
  encapsulation frame-relay IETF
  no keepalive
  frame-relay map ip 172.16.1.1 105 IETF payload-compression FRF9 stac
```

This second example shows FRF.9 compression configuration for subinterfaces.

```
interface Serial2/0/0
  no ip address
  no ip route-cache
  encapsulation frame-relay
  ip route-cache distributed
  no keepalive
!
```

```
interface Serial2/0/0.500 point-to-point
 ip address 172.16.1.4 255.255.255.0
 no cdp enable
 frame-relay interface-dlci 500 IETF
 frame-relay payload-compression FRF9 stac
```

## 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.

We recommend that you shut down the interface or subinterface prior to adding or changing compression techniques. Although this is not required, shutting down the interface ensures the interface is reset for the new data structures.

### 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
           dlci 177 (0xB1,0x2C10), static,
           broadcast,
           CISCO
           TCP/IP Header Compression (inherited), passive (inherited)
```

This example also applies to dynamic mappings achieved with the use of inverse-arp on point-to-point subinterfaces where no Frame Relay maps are configured.

### 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
           dlci 177 (0xB1,0x2C10), static,
           broadcast,
           CISCO
```

We recommend that you shut down the interface or subinterface prior to adding or changing compression techniques. Although this is not required, shutting down the interface ensures the interface is reset for the new data structures.

### Disabling TCP/IP Header Compression Examples

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

We recommend that you shut down the interface or subinterface prior to adding or changing compression techniques. Although this is not required, shutting down the interface ensures the interface is reset for the new data structures.

### 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. However, header compression is 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 was explicitly configured for header compression.

