Wide-Area Networking Configuration Guide: Frame Relay, Cisco IOS Release 15S

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

Wide-Area Networking Overview

Cisco IOS software provides a range of wide-area networking capabilities to fit almost every network environment need. Cisco offers cell relay via the Switched Multimegabit Data Service (SMDS), circuit switching via ISDN, packet switching via Frame Relay, and the benefits of both circuit and packet switching via Asynchronous Transfer Mode (ATM). LAN emulation (LANE) provides connectivity between ATM and other LAN types. The *Cisco IOS Wide-Area Networking Configuration Guide* presents a set of general guidelines for configuring the following software components:

This module gives a high-level description of each technology. For specific configuration information, see the appropriate module.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Frame Relay

The Cisco Frame Relay implementation currently supports routing on IP, DECnet, AppleTalk, XNS, Novell IPX, CLNS, Banyan VINES, and transparent bridging.

Although Frame Relay access was originally restricted to leased lines, dialup access is now supported. For more information about dialer profiles or legacy dial-on-demand routing (DDR), see the module Dial-on-Demand Routing Configuration.

To install software on a new router or access server by downloading software from a central server over an interface that supports Frame Relay, see the module Loading and Maintaining System Images.

To configure access between Systems Network Architecture (SNA) devices over a Frame Relay network, see the module Configuring SNA Frame Relay Access Support.

The Frame Relay software provides the following capabilities:

- Support for the three generally implemented specifications of Frame Relay Local Management Interfaces (LMIs):
    - The Frame Relay Interface joint specification produced by Northern Telecom, Digital Equipment Corporation, StrataCom, and Cisco Systems
    - The ANSI-adopted Frame Relay signal specification, T1.617 Annex D
    - The ITU-T-adopted Frame Relay signal specification, Q.933 Annex A
- Conformity to ITU-T I-series (ISDN) recommendation as I122, "Framework for Additional Packet Mode Bearer Services":
    - The ANSI-adopted Frame Relay encapsulation specification, T1.618
    - The ITU-T-adopted Frame Relay encapsulation specification, Q.922 Annex A
- Conformity to Internet Engineering Task Force (IETF) encapsulation in accordance with RFC 2427, except bridging.
- Support for a keepalive mechanism, a multicast group, and a status message, as follows:
    - The keepalive mechanism provides an exchange of information between the network server and the switch to verify that data is flowing.
    - The multicast mechanism provides the network server with a local data-link connection identifier (DLCI) and a multicast DLCI. This feature is specific to our implementation of the Frame Relay joint specification.
    - The status mechanism provides an ongoing status report on the DLCIs known by the switch.
- Support for both PVCs and SVCs in the same sites and routers.
    - 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.
- Support for Frame Relay Traffic Shaping beginning with Cisco IOS Release 11.2. Traffic shaping provides the following:
    - Rate enforcement for individual circuits--The peak rate for outbound traffic can be set to the committed information rate (CIR) or some other user-configurable rate.
    - Dynamic traffic throttling on a per-virtual-circuit basis--When backward explicit congestion notification (BECN) packets indicate congestion on the network, the outbound traffic rate is automatically stepped down; when congestion eases, the outbound traffic rate is stepped up again.

  - Enhanced queueing support on a per-virtual circuit basis--Custom queueing, priority queueing, and weighted fair queueing can be configured for individual virtual circuits.

- Transmission of congestion information from Frame Relay to DECnet Phase IV and CLNS. This mechanism promotes forward explicit congestion notification (FECN) bits from the Frame Relay layer to upper-layer protocols after checking for the FECN bit on the incoming DLCI. Use this Frame Relay congestion information to adjust the sending rates of end hosts. FECN-bit promotion is enabled by default on any interface using Frame Relay encapsulation. No configuration is required.
- Support for Frame Relay Inverse ARP as described in RFC 1293 for the AppleTalk, Banyan VINES, DECnet, IP, and IPX protocols, and for native hello packets for DECnet, CLNP, and Banyan VINES. It allows a router running Frame Relay to discover the protocol address of a device associated with the virtual circuit.
- Support for Frame Relay switching, whereby packets are switched based on the DLCI--a Frame Relay equivalent of a Media Access Control (MAC)-level address. Routers are configured as a hybrid DTE switch or pure Frame Relay DCE access node in the Frame Relay network.

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

  The Cisco implementation of Frame Relay switching provides the following functionality:

- Switching over an IP tunnel
- Switching over Network-to-Network Interfaces (NNI) to other Frame Relay switches
- Local serial-to-serial switching
- Switching over ISDN B channels
- Traffic shaping on switched PVCs
- Congestion management on switched PVCs
- Traffic policing on User-Network Interface (UNI) DCE
- FRF.12 fragmentation on switched PVCs

- Support for *subinterfaces* associated with a physical interface. The software groups one or more PVCs under separate subinterfaces, which in turn are located under a single physical interface. See the Configuring Frame Relay module.
- Support for fast-path transparent bridging, as described in RFC 1490, for Frame Relay encapsulated serial and High-Speed Serial Interfaces (HSSIs) on all platforms.
- Support of the Frame Relay DTE MIB specified in RFC 1315. However, the error table is not implemented. To use the Frame Relay MIB, refer to your MIB publications.
- Support for Frame Relay fragmentation. Cisco has developed the following three types of Frame Relay fragmentation:
  - End-to-End FRF.12 Fragmentation

    FRF.12 fragmentation is defined by the FRF.12 Implementation Agreement. This standard was developed to allow long data frames to be fragmented into smaller pieces (fragments) and interleaved with real-time frames. End-to-end FRF.12 fragmentation is recommended for use on PVCs that share links with other PVCs that are transporting voice and on PVCs transporting Voice over IP (VoIP).

- Frame Relay Fragmentation Using FRF.11 Annex C

  When VoFR (FRF.11) and fragmentation are both configured on a PVC, the Frame Relay fragments are sent in the FRF.11 Annex C format. This fragmentation is used when FRF.11 voice traffic is sent on the PVC, and it uses the FRF.11 Annex C format for data.

  See the module Configuring Voice over Frame Relay in the *Cisco IOS Voice, Video, and Fax Configuration Guide* for configuration tasks and examples for Frame Relay fragmentation using FRF.11 Annex C.

- Cisco Proprietary Fragmentation

  Cisco proprietary fragmentation is used on data packets on a PVC that is also used for voice traffic.

  See the module Configuring Voice over Frame Relay in the *Cisco IOS Voice, Video, and Fax Configuration Guide* for configuration tasks and examples for Cisco proprietary fragmentation.

Frame Relay-ATM Internetworking

Cisco IOS software supports the Frame Relay Forum implementation agreements for Frame Relay-ATM Interworking. Frame Relay-ATM Interworking enables Frame Relay and ATM networks to exchange data, despite differing network protocols. There are two types of Frame Relay-ATM Interworking:

FRF.5 Frame Relay-ATM Network Interworking

FRF.5 provides network interworking functionality that allows Frame Relay end users to communicate over an intermediate ATM network that supports FRF.5. Multiprotocol encapsulation and other higher-layer procedures are transported transparently, just as they would be over leased lines.

FRF.5 describes network interworking requirements between Frame Relay Bearer Services and Broadband ISDN (BISDN) permanent virtual circuit (PVC) services.

The FRF.5 standard is defined by the Frame Relay Forum Document Number FRF.5: *Frame Relay/ATM PVC Network Interworking Implementation Agreement.* For information about which sections of this implementation agreement are supported by Cisco IOS software, see Frame Relay-ATM Interworking Supported Standards.

FRF.8 Frame Relay-ATM Service Interworking

FRF.8 provides service interworking functionality that allows a Frame Relay end user to communicate with an ATM end user. Traffic is translated by a protocol converter that provides communication among dissimilar Frame Relay and ATM equipment.

FRF.8 describes a one-to-one mapping between a Frame Relay PVC and an ATM PVC.

The FRF.8 standard is defined by the Frame Relay Forum Document Number FRF.8: *Frame Relay/ATM PVC Network Service Interworking Implementation Agreement.* For information about which sections of this implementation agreement are supported by Cisco IOS software, see Frame Relay-ATM Interworking Supported Standards.

Switched Multimegabit Data Service

The Cisco implementation of the SMDS protocol is based on cell relay technology as defined in the Bellcore Technical advisories, which are based on the IEEE 802.6 standard. We provide an interface to an SMDS network using DS1 or DS3 high-speed transmission facilities. Connection to the network is made through a device called an SDSU--an SMDS digital service unit (DSU). The SDSU attaches to a router or access server through a serial port. On the other side, the SDSU terminates the line.

The implementation of SMDS supports the IP, DECnet, AppleTalk, XNS, Novell IPX, Banyan VINES, and OSI internetworking protocols, and transparent bridging.

The implementation of SMDS also supports SMDS encapsulation over an ATM interface. For more information and for configuration tasks, see Configuring ATM.

Routing of AppleTalk, DECnet, IP, IPX, and ISO CLNS is fully dynamic; that is, the routing tables are determined and updated dynamically. Routing of the other supported protocols requires that you establish a static routing table of SMDS neighbors in a user group. Once this table is set up, all interconnected routers and access servers provide dynamic routing.

Note When configuring IP routing over SMDS, you may need to make adjustments to accommodate split horizon effects. Refer to the Configuring EIGRP module for information about how Cisco software handles possible split horizon conflicts. By default, split horizon is *disabled* for SMDS networks.

The SMDS implementation includes multiple logical IP subnetworks support as defined by RFC 1209. This RFC describes routing IP over an SMDS cloud in which each connection is considered a host on one specific private network, and points to cases where traffic must transit from network to network.

The implementation of SMDS also provides the Data Exchange Interface (DXI) Version 3.2 with *heartbeat* . The heartbeat mechanism periodically generates a heartbeat poll frame.

When a multicast address is not available to a destination, pseudobroadcasting can be enabled to broadcast packets to those destinations using a unicast address.

Link Access Procedure - Balanced and X.25

X.25 is one of a group of specifications published by the ITU-T. These specifications are international standards that are formally called *Recommendations* . The ITU-T *Recommendation X.25* defines how connections between DTE and DCE are maintained for remote terminal access and computer communications. The X.25 specification defines protocols for two layers of the Open Systems Interconnection (OSI) reference model. The data link layer protocol defined is LAPB. The network layer is sometimes called the packet level protocol (PLP), but is commonly (although less correctly) referred to as the X.25 protocol.

The ITU-T updates its *Recommendations* periodically. The specifications dated 1980 and 1984 are the most common versions currently in use. Additionally, the International Standards Organization (ISO) has published ISO 7776:1986 as an equivalent to the LAPB standard, and ISO 8208:1989 as an equivalent to the ITU-T 1984 *Recommendation X.25* packet layer. The Cisco X.25 software follows the ITU-T 1984 *Recommendation X.25* , except for its Defense Data Network (DDN) and Blacker Front End (BFE) operation, which follow the ITU-T 1980 *Recommendation X.25* .

Note The ITU-T carries out the functions of the former CCITT. The 1988 X.25 standard was the last published as a CCITT *Recommendation* . The first ITU-T *Recommendation* is the 1993 revision.

In addition to providing remote terminal access, The Cisco X.25 software provides transport for LAN protocols--IP, DECnet, XNS, ISO CLNS, AppleTalk, Novell IPX, Banyan VINES, and Apollo Domain--and bridging.

Cisco IOS X.25 software provides the following capabilities:

- LAPB datagram transport--LAPB is a protocol that operates at Level 2 (the data link layer) of the OSI reference model. It offers a reliable connection service for exchanging data (in units called *frames*) with one other host. The LAPB connection is configured to carry a single protocol or multiple protocols. Protocol datagrams (IP, DECnet, AppleTalk, and so forth) are carried over a reliable LAPB connection, or datagrams of several of these protocols are encapsulated in a proprietary protocol and carried over a LAPB connection. Cisco also implements transparent bridging over multiprotocol LAPB encapsulations on serial interfaces.
- X.25 datagram transport-- X.25 can establish connections with multiple hosts; these connections are called virtual circuits. Protocol datagrams (IP, DECnet, AppleTalk, and so forth) are encapsulated inside packets on an X.25 virtual circuit. Mappings between the X.25 address of a host and its datagram protocol addresses enable these datagrams to be routed through an X.25 network, thereby permitting an X.25 PDN to transport LAN protocols.
- X.25 switch--X.25 calls can be routed based on their X.25 addresses either between serial interfaces on the same router (local switching) or across an IP network to another route r, using X.25 over TCP (XOT). XOT encapsulates the X.25 packet level inside a TCP connection, allowing X.25 equipment to be connected via a TCP/IP-based network. The Cisco X.25 switching features provide a convenient way to connect X.25 equipment, but do not provide the specialized features and capabilities of an X.25 PDN.
- ISDN D channel--X.25 traffic over the D channel, using up to 9.6 kbps bandwidth, can be used to support many applications. For example, it may be required as a primary interface where low volume sporadic interactive traffic is the normal mode of operation. For information on how to configure X.25 on ISDN, refer to the modules Configuring X.25 on ISDN and Configuring X.25 on ISDN Using AO/DI.
- PAD--User sessions can be carried across an X.25 network using the packet assembler/disassembler (PAD) protocols defined by the ITU-T Recommendations X.3 and X.29.
- QLLC--The Cisco IOS software can use the Qualified Logical Link Control (QLLC) protocol to carry SNA traffic through an X.25 network.
- Connection-Mode Network Service (CMNS)--CMNS is a mechanism that uses OSI-based network service access point (NSAP) addresses to extend local X.25 switching to nonserial media (for example, Ethernet, FDDI, and Token Ring). This implementation provides the X.25 PLP over Logical Link Control, type 2 (LLC2) to allow connections over nonserial interfaces. The Cisco CMNS implementation supports services defined in ISO Standards 8208 (packet level) and 8802-2 (frame level).
- DDN and BFE X.25--The DDN-specified Standard Service is supported. The DDN X.25 Standard Service is the required protocol for use with DDN Packet-Switched Nodes (PSNs). The Defense Communications Agency (DCA) has certified the Cisco DDN X.25 Standard Service implementation for attachment to the DDN. The Cisco DDN implementation also includes Blacker Front End operation.
- X.25 MIB--Subsets of the specifications in *SNMP MIB Extension for X.25 LAPB* (RFC 1381) and *SNMP MIB Extension for the X.25 Packet Layer* (RFC 1382) are supported. The LAPB XID Table, X.25 Cleared

Circuit Table, and X.25 Call Parameter Table are not implemented. All values are read-only. To use the X.25 MIB, refer to the RFCs.

- Closed User Groups (CUGs)--A CUG is a collection of DTE devices for which the network controls access between two members and between a member and a nonmember. An X.25 network can support up to 10,000 CUGs. CUGs allow various network subscribers (DTE devices) to be segregated into private subnetworks that have limited incoming or outgoing access.

The Cisco X.25 implementation does not support fast switching.

Layer 2 Virtual Private Network

L2VPN services are point-to-point. They provide Layer 2 point-to-point connectivity over either an MPLS or a pure IP (L2TPv3) core.

Layer 2 Tunneling Protocol Version 3

The Layer 2 Tunneling Protocol Version 3 feature expands Cisco's support of Layer 2 VPNs. Layer 2 Tunneling Protocol Version 3 (L2TPv3) is an IETF l2tpext working group draft that provides several enhancements to L2TP to tunnel any Layer 2 payload over L2TP. Specifically, L2TPv3 defines the L2TP protocol for tunneling Layer 2 payloads over an IP core network by using Layer 2 VPNs.

L2VPN Pseudowire Redundancy

L2VPNs can provide pseudowire resiliency through their routing protocols. When connectivity between end-to-end PE routers fails, an alternative path to the directed LDP session and the user data can take over. However, there are some parts of the network where this rerouting mechanism does not protect against interruptions in service. The L2VPN Pseudowire Redundancy feature provides the ability to ensure that the CE2 router in can always maintain network connectivity, even if one or all the failures in the figure occur. The L2VPN Pseudowire Redundancy feature enables you to set up backup pseudowires. You can configure the network with redundant pseudowires (PWs) and redundant network elements.

Layer 2 Virtual Private Network Interworking

Layer 2 transport over MPLS and IP already exists for like-to-like attachment circuits, such as Ethernet-to-Ethernet or PPP-to-PPP. L2VPN Interworking builds on this functionality by allowing disparate attachment circuits to be connected. An interworking function facilitates the translation between the different Layer 2 encapsulations. The L2VPN Interworking feature supports Ethernet, 802.1Q (VLAN), Frame Relay, ATM AAL5, and PPP attachment circuits over MPLS and L2TPv3.

Layer 2 Local Switching

Local switching allows you to switch Layer 2 data between two interfaces of the same type (for example, ATM to ATM, or Frame Relay to Frame Relay) or between interfaces of different types (for example, Frame Relay to ATM) on the same router. The interfaces can be on the same line card or on two different cards.

During these kinds of switching, the Layer 2 address is used, not any Layer 3 address. Same-port local switching allows you to switch Layer 2 data between two circuits on the same interface.

Wide Area Application Services

Cisco's WAAS Express software interoperates with WAN optimization headend applications from Cisco and improves WAN access and use by optimizing applications that require high bandwidth or are bound to a LAN, such as backup.

WAAS Express helps enterprises meet the following objectives:

- Complements the Cisco WAN optimization system by adding the capability to the branch routers.
- Provide branch office employees with LAN-like access to information and applications across a geographically distributed network.
- Minimize unnecessary WAN bandwidth consumption through the use of advanced compression algorithms.
- Virtualize print and other local services to branch office users.
- Improve application performance over the WAN by addressing the following common issues:
    - Low data rates (constrained bandwidth)
    - Slow delivery of frames (high network latency)
    - Higher rates of packet loss (low reliability)

The Network Analysis Module (NAM) Performance Agent (PA) for WAAS Express analyzes and measures network traffic. The PA enables baselining, monitoring, and troubleshooting of application performance. The analysis and measurement of network traffic is done by the Measurement, Aggregation, and Correlation Engine (MACE). MACE performs the required measurements on a subset of traffic and exports the necessary metrics to a target.

CHAPTER 2

Configuring Frame Relay

Feature History

Release	Modification
Cisco IOS	For information about feature support in Cisco IOS software, use Cisco Feature Navigator.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About Frame Relay

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 (VC)-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

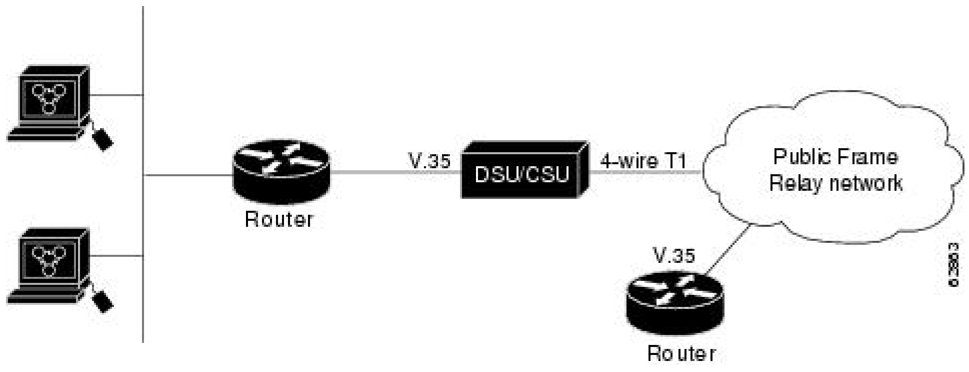
You can create Frame Relay connections using one of the following hardware configurations:

- Routers and access servers connected directly to the Frame Relay switch
- Routers and access servers connected 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 be configured for only 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. The figure below illustrates the connections among the components.

Figure 1: Typical Frame Relay Configuration



The Frame Relay interface actually consists of one physical connection between the network server and the switch that provides the service. This single physical connection provides direct connectivity to each device on a network.

Frame Relay Encapsulation

Frame Relay supports encapsulation of all supported protocols in conformance with RFC 1490, *Multiprotocol Interconnect over Frame Relay*, allowing interoperability among multiple vendors. Use the IETF form of Frame Relay encapsulation if your device 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-VC basis.

Shut down the interface prior to changing encapsulation types. Although shutting down the interface is not required, it ensures that the interface is reset for the new encapsulation.

Dynamic or Static Address Mapping

Dynamic Address Mapping

Dynamic address mapping uses Frame Relay Inverse Address Resolution Protocol (ARP) to request the next-hop protocol address for a specific connection, given its known Data link connection identifier (DLCI). Responses to Inverse ARP requests are entered in an address-to-DLCI mapping table on the device or access server. The DLCI mapping 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. However, it 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. For more information, see the Disabling or Reenabling Frame Relay Inverse ARP section.

Note Because Inverse ARP is enabled by default, no additional command is required to configure dynamic mapping on an interface and packets are not sent out for protocols that are not enabled on the interface.

Static Address Mapping

A static map links a specified next-hop protocol address to a specified Data link connection identifier (DLCI). Static mapping removes the need for Inverse Address Resolution Protocol (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 in the any of the following scenarios:

- If the device at the other end does not support Inverse ARP at all
- If the device does not support Inverse ARP for a specific protocol that you want to use over Frame Relay.

You can simplify the configuration for the Open Shortest Path First (OSPF) protocol by adding the optional **broadcast** keyword when doing this task. Refer to the **frame-relay map** command description in the *Cisco IOS Wide-Area Networking Command Reference* and the examples at the end of this chapter for more information about using the **broadcast** keyword.

LMI

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 LMI explicitly.

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.

Activating LMI Autosense

Status Request

When LMI autosense is active, it sends out a full status request, in all three LMI types, to the switch. The order is ANSI, ITU, cisco, but it is done in rapid succession. software provides the ability to 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 prompts a reply (status message) from the switch. The device decodes the format of the reply and configures itself automatically. If more than one reply is received, the device configures itself with the type of the last received reply. This is to accommodate intelligent switches that can handle multiple formats simultaneously.

LMI Autosense

If Local Management Interface (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 attempts to ascertain the LMI type. For more information about N391, see the **frame-relay lmi-n391dte** command in the chapter "Frame Relay Commands " in the *Cisco IOS Wide-Area Networking Command Reference* .

The only visible indication to the user that LMI autosense is in progress is that **debug frame lmi** is enabled. At every N391 interval, the user sees 3 rapid status inquiries from the serial interface one in each of the following LMI-type:

- ANSI
- ITU
- Cisco

Configuration Options

No configuration options are provided; LMI autosense is transparent to the user. You can turn off LMI autosense by explicitly configuring an Local Management Interface (LMI) type. The LMI type must be written into NVRAM so that next time the device 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 system:running-config** or **copy nvram:startup-config** command.

Frame Relay SVCs

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 VCs between endpoints.

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.

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.

Operating SVCs

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.

Frame Relay Traffic Shaping

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

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

Defining VCs for Different Types of Traffic

By defining separate VCs for different types of traffic and specifying queueing and an outbound traffic rate for each VC, you can provide guaranteed bandwidth for each type of traffic. By specifying different traffic rates for different VCs 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 queueing also prevents congestion-caused data loss.

Frame Relay ForeSight

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

ForeSight allows Cisco Frame Relay routers to process and react to ForeSight messages and adjust VC level traffic shaping in a timely manner.

ForeSight must be configured explicitly on both the Cisco router and the Cisco switch. 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 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 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.

When ForeSight is enabled, Frame Relay traffic shaping will adapt to ForeSight messages and BECN messages.

Frame Relay 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 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 VC.

Frame Relay 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.

Enhanced Local Management Interface

Enhanced Local Management Interface (ELMI) allows the router to learn QoS parameters and connectivity information from the Cisco switch and to use this information for traffic shaping, configuration, or management purposes. ELMI simplifies the process of configuring traffic shaping on the router and reduces chances of specifying inconsistent or incorrect values when configuring the router. ELMI works between Cisco routers and Cisco switches (BPX and IGX platforms).

ELMI QoS Autosense

When used in conjunction with traffic shaping, ELMI enables the router to respond to changes in the network dynamically. ELMI enables automated exchange of Frame Relay QoS parameter information between the Cisco router and the Cisco switch. The figure below illustrates a Cisco switch and a Cisco router, both configured with ELMI enabled. The switch sends QoS information to the router, which uses it for traffic rate enforcement.

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 QoS values from the switch and can be configured to use those values in traffic shaping.

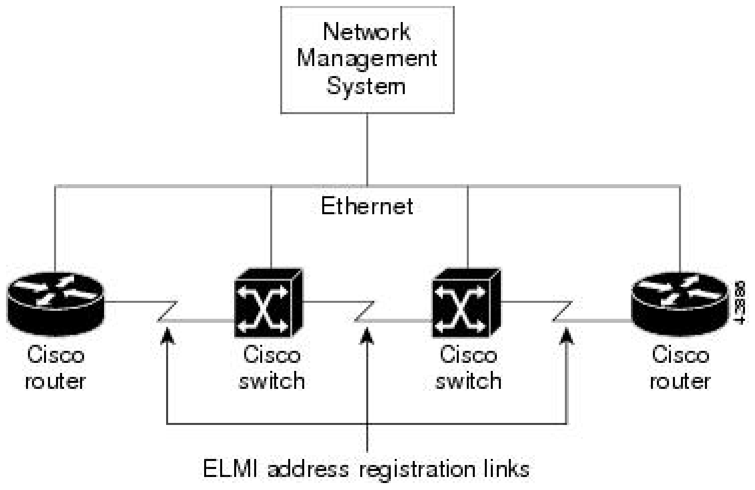
It is not necessary to configure traffic shaping on the interface to enable ELMI, but you may want to do so in order to know the values being used by the switch. If you want the router to respond to the QoS information received from the switch by adjusting the output rate, you must configure traffic shaping on the interface. To configure traffic shaping, use the **frame-relay traffic-shaping** command in interface configuration mode.

ELMI Address Registration

ELMI address registration enables a network management system (NMS) to detect connectivity among Cisco switches and routers in a network using the ELMI protocol. During ELMI version negotiation, neighboring devices exchange their management IP addresses and ifIndex. The NMS polls the devices and uses the Cisco Frame Relay MIB to collect this connectivity information. ELMI address registration allows for autodetection of the complete network topology.

The figure below shows a typical network in which ELMI address registration is in use.

Figure 3: Connectivity Detection Using ELMI Address Registration



ELMI address registration takes place on all interfaces on which ELMI is enabled, even if all the interfaces are connected to the same router or switch. The router periodically sends a version inquiry message with version information, the management IP address, and ifIndex to the switch. The switch sends its management IP address and ifIndex using the version status message. When the management IP address of the switch changes, an asynchronous ELMI version status message is immediately sent to the neighboring device.

Note The ELMI address registration mechanism does not check for duplicate or illegal addresses.

When ELMI is enabled, the router automatically chooses the IP address of one of the interfaces to use for ELMI address registration purposes. The router will choose the IP address of an Ethernet interface first, and then serial and other interfaces. You have the option to use the IP address chosen by the router or to disable the autoaddress mechanism and configure the management IP address yourself. You can also choose to disable ELMI address registration on a specific interface or on all interfaces.

Traffic-Shaping Map Class for the Interface

If you specify a Frame Relay map class for a main interface, all the VCs on its subinterfaces inherit all the traffic-shaping parameters defined for the class. You can override the default for a specific DLCI on a specific subinterface by using the **class** VC configuration command to assign the DLCI explicitly to a different class. See the section Configuring Frame Relay Subinterfaces, on page 53 for information about setting up subinterfaces.

For an example of assigning subinterface DLCIs to the default class and assigning others explicitly to a different class, see the section Example Frame Relay Traffic Shaping, on page 70.

Specifying Map Class with Queueing and Traffic-Shaping Parameters

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

Defining 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.

Defining 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.

Defining Custom Queue Lists for the Map Class

You can define a 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.

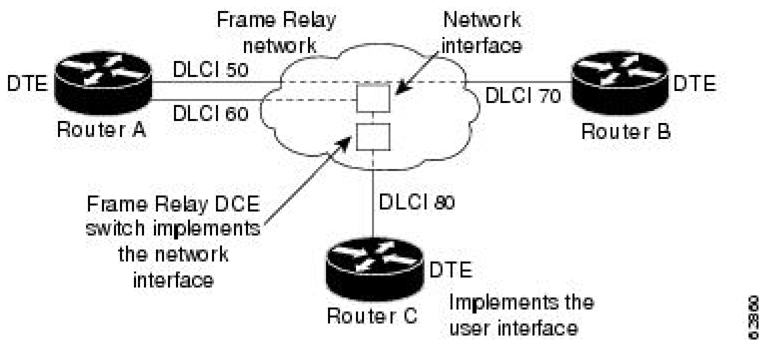
Frame Relay Switching

Frame Relay switching is a means of switching packets based on the DLCI, which can be considered the Frame Relay equivalent of a MAC address. You perform switching by configuring your Cisco router or access server into a Frame Relay network. There are two parts to a Frame Relay network:

- Frame Relay DTE (the router or access server)
- Frame Relay DCE switch

The figure below illustrates Frame Relay switched networks. Routers A, B, and C are Frame Relay DTEs connected to each other via a Frame Relay network.

Figure 4: Frame Relay Switched Network



Frame Relay switching is supported on the following interface types:

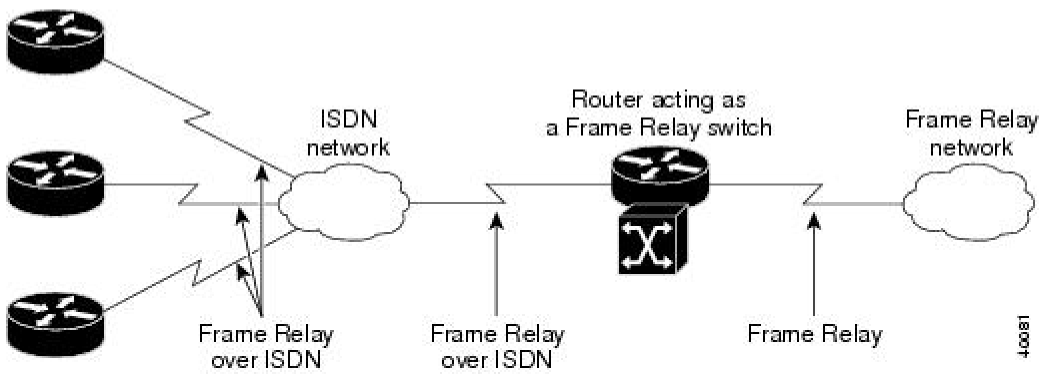
- Serial interfaces
- ISDN interfaces

Note Frame Relay switching is not supported on subinterfaces.

Frame Relay Switching over ISDN B Channels

Frame Relay switching over ISDN B channels enables you to transport Frame Relay data over ISDN. This feature allows small offices to be hubbed out of larger offices rather than being connected directly to the core network. The hub router acts as a Frame Relay switch, switching between ISDN and serial interfaces, as shown in the figure below.

Figure 5: Router Used As a Frame Relay Switch over ISDN



Frame Relay switching over ISDN provides the following functionality:

- LMI is supported on ISDN Frame Relay DCE interfaces.

- A single BRI/PRI interface can use a combination of switched PVCs and terminated Frame Relay PVCs.
- Frame Relay switching supports both leased-line ISDN, on which a B channel is permanently connected, and switched ISDN, on which B channels may be dynamically set up and torn down.

Note the following restrictions for Frame Relay switching over ISDN:

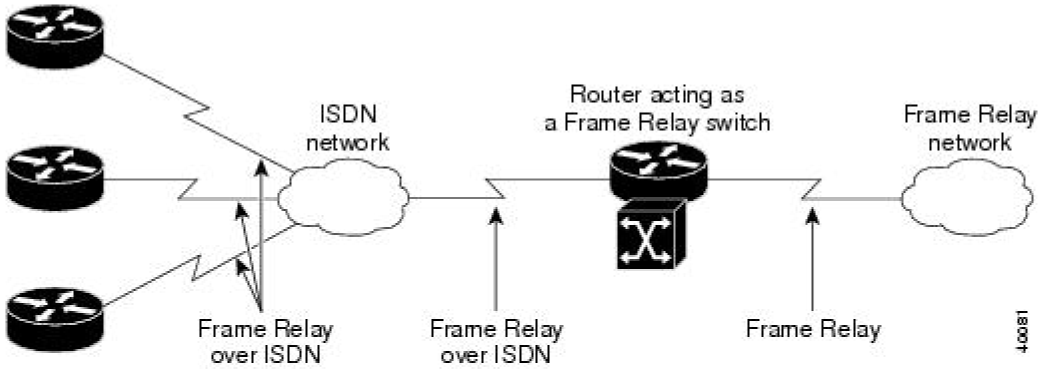
- Frame Relay traffic shaping is not supported on ISDN interfaces.
- The router configured for Frame Relay switching over ISDN cannot initiate the ISDN call.
- PVC-level congestion management is not supported over ISDN. Interface-level congestion management is supported.

When Frame Relay switching is performed by using a dialer profile, encapsulation of the underlying physical (BRI) interface must be configured as high-level data link control (HDLC).

Frame Relay Traffic Shaping on Switched PVCs

Applying Frame Relay traffic shaping to switched PVCs enables a router to be used as a Frame Relay port concentrator in front of a Frame Relay switch. The Frame Relay switch will shape the concentrated traffic before sending it into the network. The figure below shows the network configuration.

Figure 6: Router Used As a Frame Relay Port Concentrator



When you configure traffic shaping, you will define the traffic-shaping parameters in a Frame Relay map class and then attach the map class to the interface or a single switched PVC. All the traffic-shaping map-class parameters are applicable to switched PVCs: namely, Bc, Be, CIR, minimum CIR, average rate, peak rate, and adaptive shaping.

Frame Relay traffic shaping must be enabled on the interface before traffic-shaping map-class parameters will be effective. Note that when you enable Frame Relay traffic shaping, all PVCs, switched and terminated, will be shaped on that interface. Switched PVCs that are not associated with a map class will inherit shaping parameters from the interface or use default values.

Traffic Policing

Traffic policing prevents congestion on incoming PVCs by discarding or setting the DE bit on packets that exceed specified traffic parameters.

You can associate the map class with the interface or individual switched PVCs. Switched PVCs that are not associated with a map class will inherit policing parameters from the interface.

If you use a map class to configure both traffic policing and shaping, use the **in** keyword to specify incoming traffic for policing and the **out** keyword to specify outgoing traffic for shaping. If you configure shaping on one segment of a switched PVC and policing on the other, the shaping parameters will be derived from the policing parameters unless you specifically define shaping parameters in the map class.

Congestion Management on Switched PVCs

Frame Relay congestion management can be used to manage outgoing traffic congestion on switched PVCs. When Frame Relay congestion management is enabled, one way that the router manages congestion is by setting backward explicit congestion notification (BECN) and forward explicit congestion notification (FECN) bits on packets. When a switched PVC or interface is congested, packets experiencing congestion are marked with the FECN bit, and packets traveling in the reverse direction are marked with the BECN bit. When these bits reach a user device at the end of the network, the user device can react to the ECN bits and adjust the flow of traffic.

When the output interface queue reaches or exceeds the ECN excess threshold, Frame Relay bit packets on all PVCs crossing that interface will be marked with FECN or BECN, depending on their direction of travel. When the queue reaches or exceeds the ECN committed threshold, all Frame Relay packets will be marked with FECN or BECN.

A second way the router manages congestion is by discarding Frame Relay packets that are marked with the discard eligible (DE) bit and that exceed a specified level of congestion.

When the queue reaches or exceeds the DE threshold, Frame Relay packets with the DE bit will be discarded rather than queued.

You can define two levels of congestion. The first level applies to individual PVCs transmitting traffic in excess of the committed information rate (CIR). The second level applies to all PVCs at an interface. This scheme allows you to adjust the congestion on PVCs transmitting above the CIR before applying congestion management measures to all PVCs.

Congestion management parameters can be configured on the output interface queue and on traffic-shaping queues.

FRF.12 Fragmentation on Switched PVCs

The FRF.12 Implementation Agreement allows long data frames to be fragmented into smaller pieces. This process allows real-time traffic and non-real-time traffic to be carried together on lower-speed links without causing excessive delay to the real-time traffic. For further information about FRF.12 fragmentation, see the section End-to-End FRF.12 Fragmentation, on page 28 later in this module.

Some Frame Relay access devices do not support the FRF.12 standard for end-to-end fragmentation. Large packets sourced from these devices can cause significant serialization delay across low-speed trunks in switched networks. Using FRF.12 fragmentation can help prevent this delay. An edge router that receives large packets from a Frame Relay access device will fragment those packets before transmitting them across the switched network. The edge router that receives the fragmented packets will reassemble those packets before sending them to a Frame Relay access device that does not support FRF.12. If the receiving Frame Relay access device does support FRF.12, the router will transmit the fragmented packets without reassembling them.

Note the following conditions and restrictions on FRF.12 fragmentation on switched PVCs:

- Frame Relay traffic shaping must be enabled.

- Interface queueing must be dual FIFO queueing or PVC interface priority queueing.
- Switched PVCs must be configured using the **connect** command.
- If the Frame Relay access device does not support FRF.12 fragmentation, the FRF.12 Support on Switched Frame Relay PVCs feature will not benefit the interface between the Frame Relay access device and the edge router. Fragmentation and reassembly occur on the interface between the edge router and the switched Frame Relay network.
- If the Frame Relay access device is sending voice and unfragmented data on the same PVC, voice quality will suffer. The edge router will not reorder packets on switched PVCs.

Frame Relay End-to-End Keepalives

Frame Relay end-to-end keepalives enable monitoring of PVC status for network monitoring or backup applications and are configurable on a per-PVC basis with configurable timers. The Frame Relay switch within the local PVC segment deduces the status of the remote PVC segment through a Network-to-Network Interface (NNI) and reports the status to the local router. If LMI support within the switch is not end-to-end, end-to-end keepalives are the only source of information about the remote router. End-to-end keepalives verify that data is getting through to a remote device via end-to-end communication.

Each PVC connecting two end devices needs two separate keepalive systems, because the upstream path may not be the same as the downstream path. One system sends out requests and handles responses to those requests--the send side--while the other system handles and replies to requests from the device at the other end of the PVC--the receive side. The send side on one device communicates with the receive side on the other device, and vice versa.

The send side sends out a keepalive request and waits for a reply to its request. If a reply is received before the timer expires, a send-side Frame Relay end-to-end keepalive is recorded. If no reply is received before the timer expires, an error event is recorded. A number of the most recently recorded events are examined. If enough error events are accumulated, the keepalive status of the VC is changed from up to down, or if enough consecutive successful replies are received, the keepalive status of the VC is changed from down to up. The number of events that will be examined is called the *event window* .

The receive side is similar to the send side. The receive side waits for requests and sends out replies to those requests. If a request is received before the timer expires, a success event is recorded. If a request is not received, an error event is recorded. If enough error events occur in the event window, the PVC state will be changed from up to down. If enough consecutive success events occur, the state will be changed from down to up.

End-to-end keepalives can be configured in one of four modes: bidirectional, request, reply, or passive-reply.

- In bidirectional mode, both the send side and the receive side are enabled. The send side of the device sends out and waits for replies to keepalive requests from the receive side of the other PVC device. The receive side of the device waits for and replies to keepalive requests from the send side of the other PVC device.
- In request mode, only the send side is enabled, and the device sends out and waits for replies to its keepalive requests.
- In reply mode, only the receive side is enabled, and the device waits for and replies to keepalive requests.
- In passive-reply mode, the device only responds to keepalive requests, but does not set any timers or keep track of any events.

Because end-to-end keepalives allow traffic flow in both directions, they can be used to carry control and configuration information from end to end. Consistency of information between end hosts is critical in applications such as those relating to prioritized traffic and Voice over Frame Relay. Whereas SVCs can convey such information within end-to-end signalling messages, PVCs will benefit from a bidirectional communication mechanism.

End-to-end keepalives are derived from the Frame Relay LMI protocol and work between peer Cisco communications devices. The key difference is that rather than running over the signalling channel, as is the case with LMI, end-to-end keepalives run over individual data channels.

Encapsulation of keepalive packets is proprietary; therefore, the feature is available only on Cisco devices running a software release that supports the Frame Relay End-to-End Keepalive feature.

You must configure both ends of a VC to send keepalives. If one end is configured as bidirectional, the other end must also be configured as bidirectional. If one end is configured as request, the other end must be configured as reply or passive-reply. If one end is configured as reply or passive-reply, the other end must be configured as request

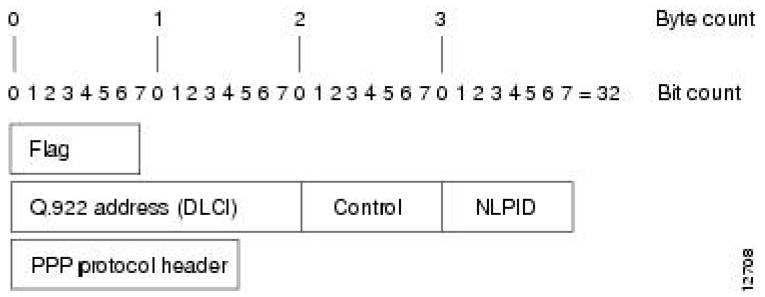
PPP over Frame Relay

Point-to-point protocol (PPP) over Frame Relay allows a router to establish end-to-end PPP sessions over Frame Relay. This is done over a PVC, which is the only circuit currently supported. The PPP session does not occur unless the associated Frame Relay PVC is in an "active" state. The Frame Relay PVC can coexist with other circuits using different Frame Relay encapsulation methods, such as RFC 1490 and the Cisco proprietary method, over the same Frame Relay link. There can be multiple PPP over Frame Relay circuits on one Frame Relay link.

One PPP connection resides on one virtual access interface. This is internally created from a virtual template interface, which contains all necessary PPP and network protocol information and is shared by multiple virtual access interfaces. The virtual access interface is coexistent with the creation of the Frame Relay circuit when the corresponding DLCI is configured. Hardware compression and fancy queueing algorithms, such as weighted fair queueing, custom queueing, and priority queueing, are not applied to virtual access interfaces.

PPP over Frame Relay is only supported on IP. IP datagrams are transported over the PPP link using RFC 1973 compliant Frame Relay framing. The frame format is shown in the figure below.

Figure 7: PPP over Frame Relay Frame Format



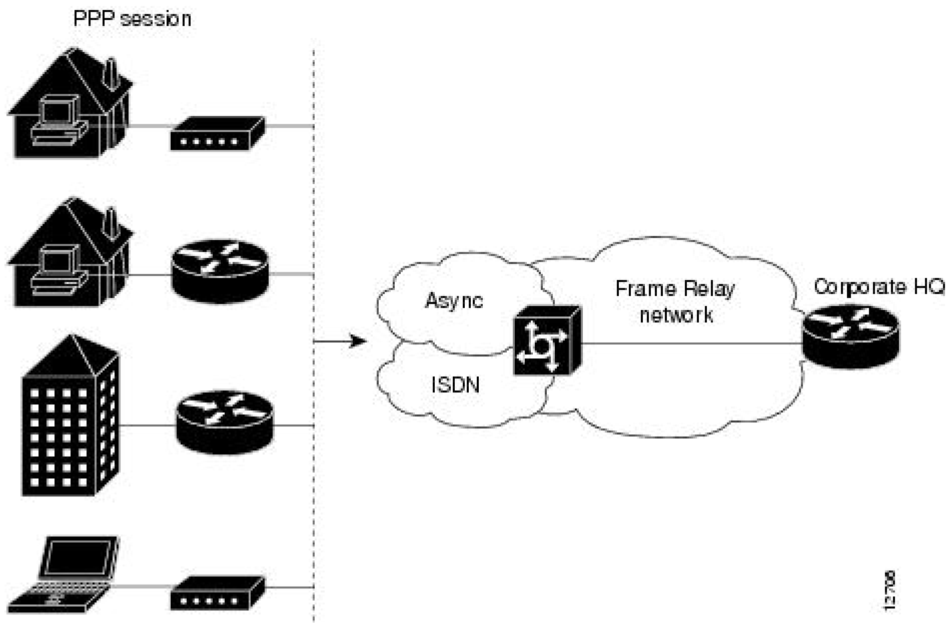
The table below lists the Frame Relay frame format components illustrated in the figure above.

Table 1: PPP Frame Relay Frame Format Descriptions

Field	Description
Flag	A single byte that indicates the beginning or end of a frame.
Address	A two-byte field that indicates the logical connection that maps to the physical channel; the DLCI.
Control	A single byte that calls for transmission of user data. PPP over Frame Relay uses a value of 0X03, which indicates that the frame is an unnumbered information (UI) frame.
NLPID	Network layer protocol ID--a single byte that uniquely identifies a PPP packet to Frame Relay.
PPP protocol	PPP packet type.

The figure below shows remote users running PPP to access their Frame Relay corporate networks.

Figure 8: PPP over Frame Relay Scenario



Before PPP over Frame Relay is configured, Frame Relay must be enabled on the router using the **encapsulation frame-relay**command. The only task required in order to implement PPP over Frame Relay is to configure the interface with the locally terminated PVC and the associated virtual template for PPP and IP, as described in the following section.

After configuring Frame Relay encapsulation on the Cisco router or access server, you must configure the physical interface with the PVC and apply a virtual template with PPP encapsulation to the DLCI.

Understanding 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.

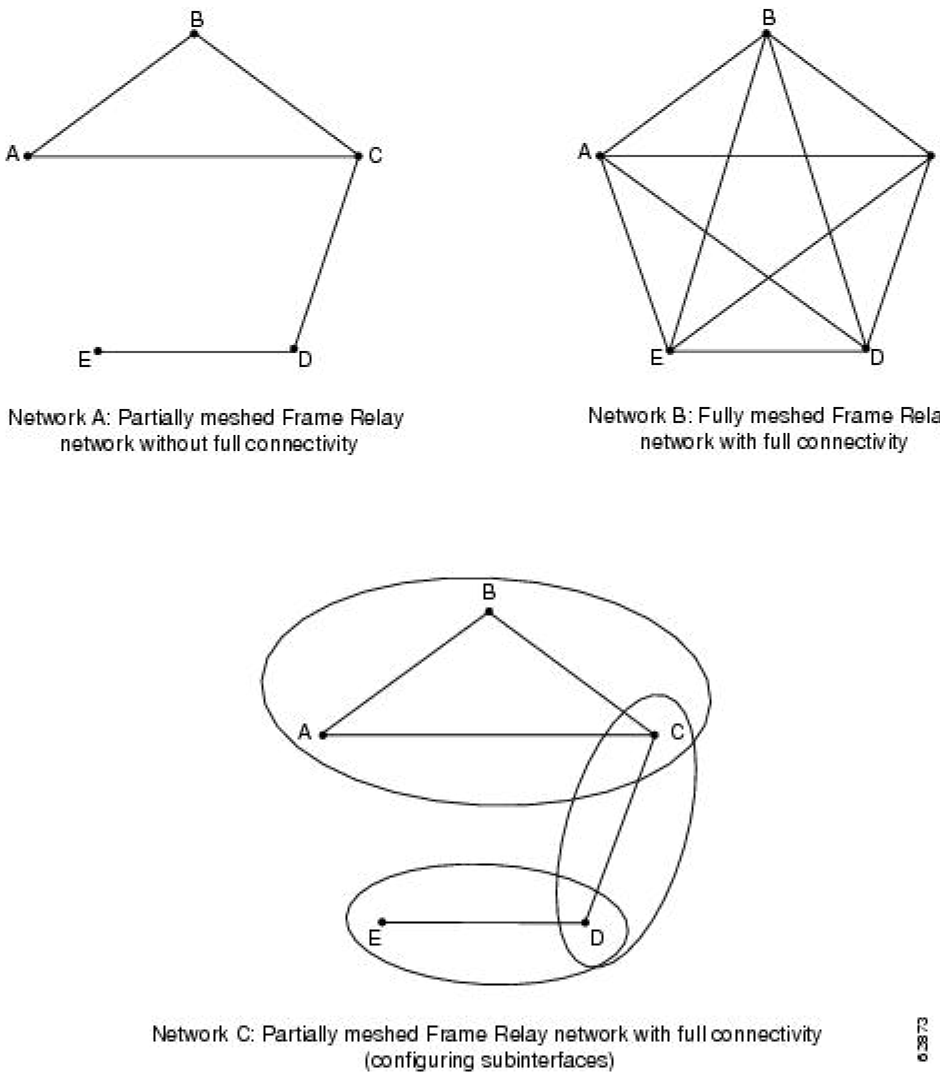
Additionally, certain protocols such as AppleTalk and transparent bridging cannot be supported on partially meshed networks because they require *split horizon* . Split horizon is a routing technique in which a packet received on an interface cannot be sent from the same interface even if received and transmitted on different VCs.

Configuring Frame Relay subinterfaces ensures that a single physical interface is treated as multiple virtual interfaces. This treatment allows you to overcome split horizon rules. Packets received on one virtual interface can be forwarded to 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 were 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.)

The figure below shows a five-node Frame Relay network 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.

Figure 9: Using Subinterfaces to Provide Full Connectivity on a Partially Meshed Frame Relay Network


B
A
C
E
D
Network A: Partially meshed Frame Relay network without full connectivity
B
A
C
E
D
Network B: Fully meshed Frame Relay network with full connectivity
B
A
C
E
D
Network C: Partially meshed Frame Relay network with full connectivity (configuring subinterfaces)
62873

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.

Subinterface Addressing

Point-to-Point Subinterfaces

For point-to-point subinterfaces, the destination is presumed to be known and is identified or implied in the **frame-relay interface-dlci** command. This 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.

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.

Addressing on Multipoint Subinterfaces

- Accepting Inverse ARP for Dynamic Address Mapping on Multipoint Subinterfaces, on page 54
- Configuring a Map Group with E.164 or X.121 Addresses, on page 40

Accepting 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.

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 section "Disabling or Reenabling Frame Relay Inverse ARP, on page 57" 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.

Configuring 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.

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 permanent virtual circuit (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 has a precedence over the backup interface of the subinterface in the case of complete loss of connectivity with the Frame Relay network. As a result, a subinterface backup is activated only in the following cases:

- If the main interface is up
- 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.

Disabling or Reenabling 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 VC.

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.

Inverse ARP is enabled by default but can be disabled explicitly for a given protocol and DLCI pair. Disable or reenable 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 at 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 at the other end of the connection.

Note If you change from a point-to-point subinterface to a multipoint subinterface, 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.

Broadcast Queue for an Interface

Very large Frame Relay networks may 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.

Frame Relay Fragmentation

Cisco has developed three types of Frame Relay fragmentation, which are described in the following sections:

The following provides further information about Frame Relay fragmentation:

End-to-End FRF.12 Fragmentation

The purpose of end-to-end FRF.12 fragmentation is to support real-time and non-real-time data packets on lower-speed links without causing excessive delay to the real-time data. FRF.12 fragmentation is defined by the FRF.12 Implementation Agreement. This standard was developed to allow long data frames to be fragmented into smaller pieces (fragments) and interleaved with real-time frames. In this way, real-time and non-real-time data frames can be carried together on lower-speed links without causing excessive delay to the real-time traffic.

End-to-end FRF.12 fragmentation is recommended for use on permanent virtual circuits (PVCs) that share links with other PVCs that are transporting voice and on PVCs transporting Voice over IP (VoIP). Although VoIP packets should not be fragmented, they can be interleaved with fragmented packets.

FRF.12 is configured on a per-PVC basis using a Frame Relay map class. The map class can be applied to one or many PVCs. Frame Relay traffic shaping must be enabled on the interface in order for fragmentation to work.

Note When Frame Relay fragmentation is configured, WFQ or LLQ is mandatory. If a map class is configured for Frame Relay fragmentation and the queueing type on that map class is not WFQ or LLQ, the configured queueing type is automatically overridden by WFQ with the default values. To configure LLQ for Frame Relay, refer to the *Cisco IOS Quality of Service Solutions Configuration Guide* , Release 12.2.

Setting the Fragment Size

Set the fragment size so that voice packets are not fragmented and do not experience a serialization delay greater than 20 ms.

To set the fragment size, the link speed must be taken into account. The fragment size should be larger than the voice packets, but small enough to minimize latency on the voice packets. Turn on fragmentation for low speed links (less than 768 kbps).

Set the fragment size based on the lowest port speed between the routers. For example, if there is a hub and spoke Frame Relay topology where the hub has a T1 speed and the remote routers have 64 kbps port speeds, the fragment size needs to be set for the 64 kbps speed on both routers. Any other PVCs that share the same physical interface need to configure the fragmentation to the size used by the voice PVC.

If the lowest link speed in the path is 64 kbps, the recommended fragment size (for 10 ms serialization delay) is 80 bytes. If the lowest link speed is 128 kbps, the recommended fragment size is 160 bytes.

For more information, refer to the " Fragmentation (FRF.12)" section in the VoIP over Frame Relay with Quality of Service (Fragmentation, Traffic Shaping, LLQ / IP RTP Priority) document.

Frame Relay Fragmentation Using FRF.11 Annex C

When VoFR (FRF.11) and fragmentation are both configured on a PVC, the Frame Relay fragments are sent in the FRF.11 Annex C format. This fragmentation is used when FRF.11 voice traffic is sent on the PVC, and it uses the FRF.11 Annex C format for data.

With FRF.11, all data packets contain fragmentation headers, regardless of size. This form of fragmentation is not recommended for use with Voice over IP (VoIP).

See the chapter "Configuring Voice over Frame Relay" in the *Cisco IOS Voice, Video, and Fax Configuration Guide* for configuration tasks and examples for Frame Relay fragmentation using FRF.11 Annex C.

Cisco-Proprietary Fragmentation

Cisco-proprietary fragmentation is used on data packets on a PVC that is also used for voice traffic. When the **vofr cisco** command is configured on a DLCI and fragmentation is enabled on a map class, the Cisco 2600 series, 3600 series, and 7200 series routers can interoperate as tandem nodes (but cannot perform call termination) with Cisco MC3810 concentrators running Cisco IOS releases prior to 12.0(3)XG or 12.0(4)T.

To configure Cisco-proprietary voice encapsulation, use the **vofr cisco** command. You must then configure a map class to enable voice traffic on the PVCs.

See the chapter "Configuring Voice over Frame Relay" in the *Cisco IOS Voice, Video, and Fax Configuration Guide* for configuration tasks and examples for Cisco-proprietary fragmentation.

Frame Relay Fragmentation and Hardware Compression Interoperability

FRF.12, FRF.11 Annex C, and Cisco-proprietary fragmentation can be used with FRF.9 or data-stream hardware compression on interfaces and virtual circuits (VCs) using Cisco-proprietary or Internet Engineering Task Force (IETF) encapsulation types.

When payload compression and Frame Relay fragmentation are used at the same time, payload compression is always performed before fragmentation.

Frame Relay fragmentation can be used with the following hardware compression modules:

- Cisco 2600 AIM-COMPR2
- Cisco 3620 and 3640 NM-COMPR
- Cisco 3660 AIM-COMPR4
- Cisco 7200 SA-COMPR

Voice over Frame Relay and Voice over IP packets will not be payload-compressed when Frame Relay fragmentation is configured.

Note On VCs using IETF encapsulation, FRF.9 hardware and software compression will work with Frame Relay fragmentation but will not work with header compression.

Frame Relay Fragmentation Conditions and Restrictions

When Frame Relay fragmentation is configured, the following conditions and restrictions apply:

- WFQ and LLQ at the PVC level are the only queueing strategies that can be used.
- Frame Relay traffic shaping (FRTS) must be configured to enable Frame Relay fragmentation (except on the Cisco 7500 series routers on which Versatile Interface Processor-Based Distributed FRF.11 and FRF.12 is enabled).
- VoFR frames are never fragmented, regardless of size.
- When end-to-end FRF.12 fragmentation is used, the VoIP packets will not include the FRF.12 header, provided the size of the VoIP packet is smaller than the fragment size configured. However, when FRF.11 Annex C or Cisco-proprietary fragmentations are used, VoIP packets will include the fragmentation header.
- If fragments arrive out of sequence, packets are dropped.

Note Fragmentation is performed after frames are removed from the WFQ.

Payload Compression

Packet-by-Packet Payload Compression

You can configure payload compression on point-to-point or multipoint interfaces or subinterfaces. Payload compression uses the Predictor 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 VC consumes approximately 40 kilobytes for dictionary memory.

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. FRF.9 is standard-based and therefore provides multivendor compatibility. FRF.9 compression uses relatively higher compression ratios, allowing more data to be compressed for faster transmission. FRF.9 compression provides the ability to maintain multiple decompression/compression histories on a per-DLCI basis.

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. 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.

Selecting FRF.9 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 main processor of the router (software compression).

Cisco-proprietary Data-Stream Compression

Data-stream compression is a proprietary hardware and software compression protocol that can be used on the same VC or interface and IP header compression. Data-stream compression is functionally equivalent to FRF.9 compression and must be used with Cisco-proprietary encapsulation. Frame Relay fragmentation can also be used with data-stream compression.

TCP IP Header Compression

TCP/IP header compression, as described by RFC 1144, *Compressing TCP/IP Headers for Low-Speed Serial Links* 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, indicates the fields that have 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 queueing changes the order in which packets are transmitted, enabling priority queueing on the interface is not recommended.

Note If you configure an interface with Cisco-proprietary 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.

Specifying an Individual IP Map for TCP IP Header Compression

Note An interface configured to support TCP/IP header compression does not also support priority queuing or custom queuing.

TCP/IP header compression requires Cisco-proprietary 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-proprietary encapsulation and TCP header compression. In addition, 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 a packet had a compressed TCP/IP header when it was received.

Specifying 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.

Note If an interface configured with Cisco-proprietary 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.

If you configure an interface with Cisco-proprietary 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.

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, providing end-to-end network transport functions intended for these real-time traffic applications and multicast or unicast network services. RTP is described in RFC 1889. RTP is not intended for data traffic, which uses TCP or UDP.

Discard Eligibility

Frame Relay packets can be set with low priority or low time sensitivity. These packets 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.

Discard eligibility requires the Frame Relay network to be able to interpret the DE bit. Some networks take no action when the DE bit is set, and others use the DE bit to determine which packets to discard. The best 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 create DE lists that identify the characteristics of packets to be eligible for discarding, and you can also specify DE groups to identify the data link connection identifier (DLCI) that is affected.

You can create DE lists based on the protocol or the interface, and on characteristics such as fragmentation of the packet, a specific TCP or UDP port, an access list number, or a packet size.

DLCI Priority Levels

Data Link Connection Identifier (DLCI) priority levels allow you to separate different types of traffic and provides a traffic management tool for congestion problems caused by the following:

- Mixing batch and interactive traffic over the same DLCI
- Queuing traffic from sites with high-speed access to destination sites with lower-speed access

Before you configure the DLCI priority levels, you must:

- Enable Frame Relay encapsulation.
- Define dynamic or static address mapping.
- Ensure 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.

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 device or access server. In fact, they are independent of the priority queues of the device. However, if you enable queuing and use the same DLCIs for queuing, then high-priority DLCIs can be put into high-priority queues.

How to Configure Frame Relay

Enabling Frame Relay Encapsulation on an Interface

Note Frame Relay encapsulation is a prerequisite for any Frame Relay commands on an interface.

To enable Frame Relay encapsulation on the interface level, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *typenumber*
4. **encapsulation frame-relay**[**ietf**]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *typenumber* **Example:** `Device(config)# int ethernet 0/1`	Specifies the interface, and enters interface configuration mode.
Step 4	**encapsulation frame-relay**[**ietf**] **Example:** `Device(config-if)# encapsulation frame-relay ietf`	Enables and specifies the Frame Relay encapsulation method.

Step 5	**end**
	
	Example:
	
	`Device(config-if)# end`
		Returns to privileged EXEC mode.

Configuring Static Address Mapping

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

Command	Purpose
`Router(config-if)#` **frame-relay map** *protocol protocol-address dlci* [**broadcast**] [**ietf**] [**cisco**]	Maps between a next-hop protocol address and DLCI destination address. 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**
`Router(config-if)#` **frame-relay map clns** *dlci* [**broadcast**]	Defines a DLCI used to send ISO CLNS frames.
`Router(config-if)#` **frame-relay map bridge** *dlci* [**broadcast**] [**ietf**]	Defines a DLCI destination bridge.

Explicitly Configuring the LMI

Setting the LMI Type

If the device 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 requirements of your private Frame Relay network. You can set one of the following three types of LMIs on Cisco devices:

- ANSI T1.617 Annex D
- Cisco
- ITU-T Q.933 Annex A

To do so, use the following commands beginning in interface configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *typenumber*
4. **frame-relay lmi-type** {**ansi** | **cisco** | **q933a**}
5. **end**
6. **copy nvram:startup-config** *destination*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *typenumber* **Example:** `Device(config)# int ethernet 0/1`	Specifies the interface, and enters interface configuration mode.

Step 4	**frame-relay lmi-type** {**ansi** | **cisco** | **q933a**}
	
	Example:
	`Device(config-if)#`
		Sets the LMI type.

Step 5	**end**
	
	Example:
	`Device(config-if)# end`
		Returns to privileged EXEC mode.

Step 6	**copy nvram:startup-config** *destination* **Example:** `Device#`	Writes the LMI type to NVRAM.

Setting the LMI Keepalive Interval

A keepalive interval must be set to configure the LMI. By default, this interval is 10 seconds and, according to the LMI protocol, must be less than the corresponding interval on the switch. To set the keepalive interval, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)#` **keepalive** *number*	Sets the LMI keepalive interval.

Setting the LMI Polling and Timer Intervals

You can set various optional counters, intervals, and thresholds to fine-tune the operation of your Local Management Interface data terminal equipment (LMI DTE) and data communications equipment (DCE) devices. Set these attributes by using one or more of the following commands in interface configuration mode:

Command	Purpose
frame-relay lmi-n392dce *threshold*	Sets the DCE and Network-to-Network Interface (NNI) error threshold.
frame-relay lmi-n393dce *events*	Sets the DCE and NNI monitored events count.
frame-relay lmi-t392dce *seconds*	Sets the polling verification timer on a DCE or NNI interface.
frame-relay lmi-n391dte *keep-exchanges*	Sets a full status polling interval on a DTE or NNI interface.
frame-relay lmi-n392dte *threshold*	Sets the DTE or NNI error threshold.
frame-relay lmi-n393dte *events*	Sets the DTE and NNI monitored events count.

Enabling Frame Relay SVC Service

Configuring SVCs on a Physical Interface

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

SUMMARY STEPS

1. Router(config)# **interface** *type number*
2. Router(config-if)# **ip address** *ip-address mask*
3. Router(config-if)# **encapsulation frame-relay**
4. Router(config-if)# **map-group** *group-name*
5. Router(config-if)# **frame-relay svc**

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number*	Specifies the physical interface.
Step 2	Router(config-if)# **ip address** *ip-address mask*	Specifies the interface IP address, if needed.
Step 3	Router(config-if)# **encapsulation frame-relay**	Enables Frame Relay encapsulation on the interface.
Step 4	Router(config-if)# **map-group** *group-name*	Assigns a map group to the interface. Map group details are specified with the **map-list** command.
Step 5	Router(config-if)# **frame-relay svc**	Enables Frame Relay SVC support on the interface.

Configuring SVCs on a Subinterface

Note This task offers additional flexibility for SVC configuration and operation.

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

SUMMARY STEPS

1. Router(config)# **interface** *type number* **.** *subinterface-number* {**multipoint** | **point-to-point**}
2. Router(config-subif)# **ip address** *ip-address mask*
3. Router(config-subif)# **map-group** *group-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number* **.** *subinterface-number* {**multipoint** | **point-to-point**}	Specifies a subinterface configured for SVC operation.

	Command or Action	Purpose
Step 2	Router(config-subif)# **ip address** *ip-address mask*	Specifies the subinterface IP address, if needed.
Step 3	Router(config-subif)# **map-group** *group-name*	Assigns a map group to the subinterface.

Configuring a Map Class

Perform the following tasks to configure a map class:

- Specify the map class name. (Required)
- Specify a custom queue list for the map class. (Optional)
- Specify a priority queue list for the map class. (Optional)
- Enable BECN feedback to throttle the output rate on the SVC for the map class. (Optional)
- Set nondefault QoS values for the map class (no need to set the QoS values; default values are provided). (Optional)

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

To configure a map class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **map-class frame-relay** *map-class-name*
2. Router(config-map-class)# **frame-relay custom-queue-list** *list-number*
3. Router(config-map-class)# **frame-relay priority-group** *list-number*
4. Router(config-map-class)# **frame-relay adaptive-shaping**[**becn** | **foresight**][1]
5. Router(config-map-class)# **frame-relay cir in** *bps*
6. Router(config-map-class)# **frame-relay cir out** *bps*
7. Router(config-map-class)# **frame-relay mincir in** *bps*[2]
8. Router(config-map-class)# **frame-relay mincir out** *bps*Configuring a Map Class, on page 39
9. Router(config-map-class)# **frame-relay bc in** *bits*Configuring a Map Class, on page 39
10. Router(config-map-class)# **frame-relay bc out** *bits*Configuring a Map Class, on page 39
11. Router(config-map-class)# **frame-relay be in** *bits*Configuring a Map Class, on page 39
12. Router(config-map-class)# **frame-relay be out** *bits*Configuring a Map Class, on page 39
13. Router(config-map-class)# **frame-relay idle-timer** *seconds*Configuring a Map Class, on page 39

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **map-class frame-relay** *map-class-name*	Specifies Frame Relay map class name and enters map class configuration mode.
Step 2	Router(config-map-class)# **frame-relay custom-queue-list** *list-number*	Specifies a custom queue list to be used for the map class.
Step 3	Router(config-map-class)# **frame-relay priority-group** *list-number*	Assigns a priority queue to VCs associated with the map class.
Step 4	Router(config-map-class)# **frame-relay adaptive-shaping**[**becn** | **foresight**][1]	Enables the type of BECN feedback to throttle the frame-transmission rate.
Step 5	Router(config-map-class)# **frame-relay cir in** *bps*	Specifies the inbound committed information rate (CIR), in bits per second.
Step 6	Router(config-map-class)# **frame-relay cir out** *bps*	Specifies the outbound CIR, in bits per second.
Step 7	Router(config-map-class)# **frame-relay mincir in** *bps*[2]	Sets the minimum acceptable incoming CIR, in bits per second.
Step 8	Router(config-map-class)# **frame-relay mincir out** *bps*Configuring a Map Class, on page 39	Sets the minimum acceptable outgoing CIR, in bits per second.
Step 9	Router(config-map-class)# **frame-relay bc in** *bits*Configuring a Map Class, on page 39	Sets the incoming committed burst size (Bc), in bits.
Step 10	Router(config-map-class)# **frame-relay bc out** *bits*Configuring a Map Class, on page 39	Sets the outgoing Bc, in bits.
Step 11	Router(config-map-class)# **frame-relay be in** *bits*Configuring a Map Class, on page 39	Sets the incoming excess burst size (Be), in bits.
Step 12	Router(config-map-class)# **frame-relay be out** *bits*Configuring a Map Class, on page 39	Sets the outgoing Be, in bits.
Step 13	Router(config-map-class)# **frame-relay idle-timer** *seconds*Configuring a Map Class, on page 39	Sets the idle timeout interval, in seconds.

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 the 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.

Configuring 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, use the following command in global configuration mode:

Command	Purpose
`Router(config)#` **map-list** *map-group-name* **source-addr** {**e164** | **x121**} `source-address` **dest-addr** {**e164** | **x121**} `destination-address`	Specifies the map group associated with specific source and destination addresses for the SVC.

Associating 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, use the following command in map list configuration mode:

Command	Purpose
`Router(config-map-list)# protocol protocol-address` **class** *class-name* [**ietf**] [**broadcast** [**trigger**]]	Specifies a destination protocol address and a Frame Relay map class name from which to derive QoS information. • 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.

Configuring LAPF Parameters

Note The LAPF tasks are not required and not recommended unless you understand thoroughly the impacts on your network.

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

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, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)#` **no frame-relay lapf frmr**	Selects not to send FRMR frames at the LAPF Frame Reject procedure.

Changing Layer 2 Parameters for Your Network

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, use the following commands as needed:

Command	Purpose
`Router(config-if)#` **frame-relay lapf k** *number*	Sets the LAPF window size k.
`Router(config-if)#` **frame-relay lapf n200** *retries*	Sets the LAPF maximum retransmission count N200.
`Router(config-if)#` **frame-relay lapf n201** *bytes*	Sets maximum length of the Information field of the LAPF I frame N201, in bytes.
`Router(config-if)#` **frame-relay lapf t200** *tenths-of-a-second*	Sets the LAPF retransmission timer value T200, in tenths of a second.
`Router(config-if)#` **frame-relay lapf t203** *`seconds`*	Sets the LAPF link idle timer value T203 of DLCI 0, in seconds.

Configuring Frame Relay Traffic Shaping

Enabling Frame Relay Traffic Shaping on the Interface

To configure a map class with traffic-shaping and per-VC queueing parameters, see the sections Specifying a Traffic-Shaping Map Class for the Interface and Defining a Map Class with Queueing and Traffic-Shaping Parameters.

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

Command	Purpose
`Router(config-if)#` **frame-relay traffic-shaping**	Enables Frame Relay traffic shaping and per-VC queueing. **Note** The default committed information rate (CIR) of 56K will apply in the following situations: When traffic shaping is enabled (by using the **frame-relay traffic-shaping** command), but a map class is not assigned to the VC and when traffic shaping is enabled (by using the **frame-relay traffic-shaping**command) and a map class is assigned to the VC, but traffic-shaping parameters have not been defined in the map class.

Configuring Enhanced Local Management Interface

Enabling ELMI

To enable ELMI, use the following commands beginning in interface configuration mode:

SUMMARY STEPS

1. Router(config)# **interface** *type number*
2. Router(config-if)# **encapsulation frame-relay**[**cisco** | **ietf**]
3. Router(config-if)# **frame-relay QoS-autosense**

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number*	Specifies the physical interface.
Step 2	Router(config-if)# **encapsulation frame-relay**[**cisco** | **ietf**]	Enables Frame Relay encapsulation on the interface.
Step 3	Router(config-if)# **frame-relay QoS-autosense**	Enables ELMI.

Disabling Automatic IP Address Selection

Automatic IP address selection is enabled by default when ELMI is enabled. To disable the automatic selection of the IP address to be used for ELMI address registration, use the following global configuration command:

Command	Purpose
`Router(config)#` **no frame-relay address registration auto-address**	Disables the automatic selection of the IP address to be used for ELMI address registration. **Note** When automatic IP address selection is disabled and an IP address has not been configured using the **frame-relay address registration ip** global configuration command, the IP address for ELMI address registration will be set to 0.0.0.0.

Configuring the IP Address to Be Used for ELMI Address Registration

To configure the IP address for ELMI address registration, use the following global configuration command:

Command	Purpose
`Router(config)#` **frame-relay address registration ip** *address*	Configures the IP address to be used for ELMI address registration. **Note** Automatic IP address selection is disabled when you configure the management IP address using the **frame-relay address registration ip** global configuration command.

Enabling ELMI Address Registration on an Interface

To enable ELMI address registration on an interface, use the following interface configuration command:

Command	Purpose
`Router(config-if)#` **frame-relay address-reg enable**	Enables ELMI address registration on an interface. To disable ELMI address registration on an interface, use the **no** form of the command.

Verifying ELMI Address Registration

To verify that ELMI address registration is configured correctly, use the following privileged EXEC configuration command:

Command	Purpose
`Router#` **show frame-relay qos-autosense** [**interface** *interface*]	Displays the QoS values and ELMI address registration information sensed from the switch.

Specifying a Traffic-Shaping Map Class for the Interface

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

SUMMARY STEPS

1. Router(config-if)# **frame-relay class** *map-class-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config-if)# **frame-relay class** *map-class-name*	Specifies a Frame Relay map class for the interface.

Defining a Map Class with Queueing and Traffic-Shaping Parameters

To define a map class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **map-class frame-relay** *map-class-name*
2. Router(config-map-class)# **frame-relay traffic-rate** *average* [*peak*]
3. Router(config-map-class)# **frame-relay custom-queue-list** *list-number*
4. Router(config-map-class)# **frame-relay priority-group** *list-number*
5. Router(config-map-class)# **frame-relay adaptive-shaping**{**becn** | **foresight**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **map-class frame-relay** *map-class-name*	Specifies a map class to define.

	Command or Action	Purpose
Step 2	Router(config-map-class)# **frame-relay traffic-rate** *average* [*peak*]	Defines the traffic rate for the map class.
Step 3	Router(config-map-class)# **frame-relay custom-queue-list** *list-number*	Specifies a custom queue list.
Step 4	Router(config-map-class)# **frame-relay priority-group** *list-number*	Specifies a priority queue list.
Step 5	Router(config-map-class)# **frame-relay adaptive-shaping**{**becn** | **foresight**}	Selects BECN or ForeSight as congestion backward-notification mechanism to which traffic shaping adapts. **Note** 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** software command.

Configuring Frame Relay Switching

Enabling 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 using the following command in global configuration mode before configuring the switch type:

Command	Purpose
`Router(config)#` **frame-relay switching**	Enables Frame Relay switching.

Configuring a Frame Relay DTE Device or 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. (DTE is the default.) To do so, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)#` **frame-relay intf-type** [**dce** | **dte** | **nni**]	Configures a Frame Relay DTE device or DCE switch.

Creating Switched PVC over ISDN

To create a switched PVC over ISDN, or to create a switched PVC on which traffic shaping, traffic policing, and congestion management can be configured, use the following command in global configuration mode:

Command	Purpose
`Router(config)#` **connect** *connection-name interface dlci interface dlci*	Defines connections between Frame Relay PVCs.

Creating a Switched PVC with Static Route

Note Static routes cannot be configured over tunnel interfaces on the Cisco 800 series, 1600 series, and 1700 series platforms. Static routes can only be configured over tunnel interfaces on platforms that have the Enterprise feature set.

To create a switched PVC with a static route, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)#` **frame-relay route** *in-dlci* **interface** *out-interface-type out-interface-number out-dlci*	Specifies a static route for PVC switching.

Identifying a PVC As Switched

Before you can associate a map class with a switched PVC, you must identify the PVC as being switched. To identify a PVC as switched, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)#` **frame-relay interface-dlci** *dlci* **switched**	Identifies a PVC as switched.

Configuring Traffic Policing on UNI DCE Devices

Enabling Frame Relay Policing

To enable Frame Relay policing on a interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)#` **frame-relay policing**	Enables Frame Relay policing on all switched PVCs on the interface.

Configuring Frame Relay Policing Parameters

To configure policing parameters in a Frame Relay map class, use one or more of the following commands in map-class configuration mode:

Command	Purpose
`Router(config-map-class)#` **frame-relay cir** {**in** | **out**} *bps*	Sets the CIR for a Frame Relay PVC, in bits per second.
`Router(config-map-class)#` **frame-relay bc** {**in** | **out**} *bits*	Sets the committed burst size for a Frame Relay PVC, in bits.
`Router(config-map-class)#` **frame-relay be** {**in** | **out**} *bits*	Sets the excess burst size for a Frame Relay PVC, in bits.
`Router(config-map-class)#` **frame-relay tc** *milliseconds*	Sets the measurement interval for policing incoming traffic on a PVC when the CIR is zero, in milliseconds.

Configuring Congestion Management on Switched PVCs

Configuring Frame Relay Congestion Management on the Interface

To configure Frame Relay congestion management on all switched PVCs on an interface, use the following commands beginning in interface configuration mode:

SUMMARY STEPS

1. Router(config-if)# **frame-relay congestion management**
2. Router(config-fr-congest)# **threshold de** *percentage*
3. Router(config-fr-congest)# **threshold ecn** {**bc** | **be**} *percentage*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config-if)# **frame-relay congestion management**	Enables Frame Relay congestion management on all switched PVCs on an interface and enters Frame Relay congestion management configuration mode.
Step 2	Router(config-fr-congest)# **threshold de** *percentage*	Configures the threshold at which DE-marked packets will be discarded from switched PVCs on the output interface.
Step 3	Router(config-fr-congest)# **threshold ecn** {**bc** | **be**} *percentage*	Configures the threshold at which ECN bits will be set on packets in switched PVCs on the output interface.

Configuring Frame Relay Congestion Management on Traffic-Shaping Queues

To configure Frame Relay congestion management on the traffic-shaping queues of switched PVCs, use one or more of the following commands in map-class configuration mode:

Command	Purpose
`Router(config-map-class)#` **frame-relay congestion threshold de** *percentage*	Configures the threshold at which DE-marked packets will be discarded from the traffic-shaping queue of a switched PVC.
`Router(config-map-class)#` **frame-relay congestion threshold ecn** *percentage*	Configures the threshold at which ECN bits will be set on packets in the traffic-shaping queue of a switched PVC.
`Router(config-map-class)#` **frame-relay holdq** *queue-size*	Configures the maximum size of a traffic-shaping queue on a switched PVC.

Configuring FRF.12 Fragmentation on Switched PVCs

To configure FRF.12 on switched PVCs, use the following map-class configuration command. The map class can be associated with one or more switched PVCs.

Command	Purpose
`Router(config-map-class)#` **frame-relay fragment** *fragment_size* **switched**	Enables FRF.12 fragmentation on switched Frame Relay PVCs for a Frame Relay map class.

Verifying Frame Relay Switching

To verify the correct configuration of Frame Relay switching, use one or more of the following commands:

Command	Purpose
`Router#` **show frame-relay fragment** [**interface** *interface*] [*dlci*]	Displays statistics about Frame Relay fragmentation.
`Router#` **show frame-relay pvc** [**interface** *interface*] [*dlci*]	Displays statistics about Frame Relay PVCs including detailed reasons for packet drops on switched PVCs and complete status information for switched NNI PVCs.
`Router#` **show interfaces** [*type number*]	Displays information about the configuration and queue at the interface.

Troubleshooting Frame Relay Switching

To diagnose problems in switched Frame Relay networks, use the following EXEC commands:

Command	Purpose
`Router#` **debug frame-relay switching** [**interface** *interface*] [*dlci*] [**interval** *seconds*]	Displays debug messages for switched Frame Relay PVCs. The **interval** keyword and *seconds* argument sets the interval at which the debug messages will be displayed.
`Router#` **show frame-relay pvc** [**interface** *interface*] [*dlci*]	Displays statistics about Frame Relay PVCs, including detailed reasons for packet drops on switched PVCs and complete status information for switched NNI PVCs.

Customizing Frame Relay for Your Network

Configuring Frame Relay End-to-End Keepalives

Configuring End-to-End Keepalives

To configure Frame Relay end-to-end keepalives, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **map-class frame-relay** *map-class-name*
2. Router(config-map-class)# **frame-relay end-to-end keepalive mode** {**bidirectional** | **request** | **reply** | **passive-reply**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **map-class frame-relay** *map-class-name*	Specifies a map class for the VC.
Step 2	Router(config-map-class)# **frame-relay end-to-end keepalive mode** {**bidirectional** | **request** | **reply** | **passive-reply**}	Specifies Frame Relay end-to-end keepalive mode. • **bidirectional** --The device sends keepalive requests to the other end of the VC and responds to keepalive requests from the other end of the VC. • **request** --The device sends keepalive requests to the other end of the VC. • **reply** --The device responds to keepalive requests from the other end of the VC. • **passive-reply** --The device responds to keepalive requests from the other end of the VC, but will not track errors or successes.

Modifying the Default Parameters

You can modify the end-to-end keepalives default parameter values by using any of the following map-class configuration commands:

Command	Purpose
`Router(config-map-class)#` **frame-relay end-to-end keepalive error-threshold** {**send** | **receive**} *count*	Modifies the number of errors needed to change the keepalive state from up to down.
`Router(config-map-class)#` **frame-relay end-to-end keepalive event-window** {**send** | **receive**} *count*	Modifies the number of recent events to be checked for errors.
`Router(config-map-class)#` **frame-relay end-to-end keepalive success-events** {**send** | **receive**} *count*	Modifies the number of consecutive success events required to change the keepalive state from down to up.
`Router(config-map-class)#` **frame-relay end-to-end keepalive timer** {**send** | **receive**} *interval*	Modifies the timer interval.

Verifying Frame Relay End-to-End Keepalives

To monitor the status of Frame Relay end-to-end keepalives, use the following command in EXEC configuration mode:

Command	Purpose
`Router#` **show frame-relay end-to-end keepalive** *interface*	Shows the status of Frame Relay end-to-end keepalives.

Enabling PPP over Frame Relay

To configure the physical interface that will carry the PPP session and link it to the appropriate virtual template interface, perform the following task in interface configuration mode:

Command	Purpose
`Router(config-if)#` **frame-relay interface-dlci** *dlci* [**ppp** *virtual-template-name*]	Defines the PVC and maps it to the virtual template.

For an example of configuring PPP over Frame Relay, see the section Example PPPoverFrameRelay, on page 83 or Example PPP over Frame Relay DCE, on page 83 later in this chapter.

Configuring Frame Relay Subinterfaces

Configuring Subinterfaces

Subinterfaces can be configured for multipoint or point-to-point communication. (There is no default.) To configure subinterfaces on a Frame Relay network, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **interface type** *number* **.** *subinterface-number* {**multipoint** | **point-to-point**}
2. Router(config-subif)# **encapsulation frame-relay**

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface type** *number* **.** *subinterface-number* {**multipoint** | **point-to-point**}	Creates a point-to-point or multipoint subinterface.
Step 2	Router(config-subif)# **encapsulation frame-relay**	Configures Frame Relay encapsulation on the serial interface.

Defining Subinterface Addressing on Point-to-Point Subinterfaces

If you specified a point-to-point subinterface in the preceding procedure, use the following command in subinterface configuration mode:

SUMMARY STEPS

1. Router(config-subif)# **frame-relay interface-dlci** *dlci*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config-subif)# **frame-relay interface-dlci** *dlci*	Associates the selected point-to-point subinterface with a DLCI.

Accepting Inverse ARP for Dynamic Address Mapping on Multipoint Subinterfaces

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

Command	Purpose
`Router(config-if)#` **frame-relay interface-dlci** *dlci*	Associates a specified multipoint subinterface with a DLCI.

Configuring Static Address Mapping on Multipoint Subinterfaces

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

Command	Purpose
`Router(config-if)#` **frame-relay map** *protocol protocol-address dlci* [**broadcast**] [**ietf**] [**cisco**]	Maps between a next-hop protocol address and DLCI destination address. 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**
`Router(config-if)#` **frame-relay map clns** *dlci* [**broadcast**]	Defines a DLCI used to send ISO CLNS frames. The **broadcast** keyword is required for routing protocols such as OSI protocols and the Open Shortest Path First (OSPF) protocol.
`Router(config-if)#` **frame-relay map bridge** *dlci* [**broadcast**] [**ietf**]	Defines a DLCI destination bridge.

Configuring Transparent Bridging for Point-to-Point Subinterfaces

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

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

SUMMARY STEPS

1. Router(config)# **interface type** *number*
2. Router(config-if)# **encapsulation frame-relay**
3. Router(config)# **interface type** *number* **:** *subinterface-number* **point-to-point**
4. Router(config-subif)# **frame-relay interface-dlci** *dlci*
5. Router(config-subif)# **bridge-group** *bridge-group*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface type** *number*	Specifies an interface.
Step 2	Router(config-if)# **encapsulation frame-relay**	Configures Frame Relay encapsulation on the interface.
Step 3	Router(config)# **interface type** *number* **:** *subinterface-number* **point-to-point**	Specifies a subinterface.
Step 4	Router(config-subif)# **frame-relay interface-dlci** *dlci*	Associates a DLCI with the subinterface.
Step 5	Router(config-subif)# **bridge-group** *bridge-group*	Associates the subinterface with a bridge group.

Configuring Transparent Bridging for Point-to-Multipoint Interfaces

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

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

SUMMARY STEPS

1. Router(config)# **interface type** *number*
2. Router(config-if)# **encapsulation frame-relay**
3. Router(config)# **interface type***number***:subinterface-number multipoint**
4. Router(config-subif)# **frame-relay map bridge** *dlci* [**broadcast**] [**ietf**]
5. Router(config-subif)# **bridge-group** *bridge-group*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface type** *number*	Specifies an interface.
Step 2	Router(config-if)# **encapsulation frame-relay**	Configures Frame Relay encapsulation.
Step 3	Router(config)# **interface type***number***:subinterface-number multipoint**	Specifies a subinterface.
Step 4	Router(config-subif)# **frame-relay map bridge** *dlci* [**broadcast**] [**ietf**]	Defines a DLCI destination bridge.
Step 5	Router(config-subif)# **bridge-group** *bridge-group*	Associates the subinterface with a bridge group.

Configuring a Backup Interface for a Subinterface

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

SUMMARY STEPS

1. Router(config)# **interface type** *number*
2. Router(config-if)# **encapsulation frame-relay**
3. Router(config)# **interface type** *number* **.** *subinterface-number* **point-to-point**
4. Router(config-subif)# **frame-relay interface-dlci** *dlci*
5. Router(config-subif)# **backup interface type** *number*
6. Router(config-subif)# **backup delay** *enable-delay disable-delay*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface type** *number*	Specifies the interface.
Step 2	Router(config-if)# **encapsulation frame-relay**	Configures Frame Relay encapsulation.

	Command or Action	Purpose
Step 3	Router(config)# **interface type** *number* **.** *subinterface-number* **point-to-point**	Configures the subinterface.
Step 4	Router(config-subif)# **frame-relay interface-dlci** *dlci*	Specifies DLCI for the subinterface.
Step 5	Router(config-subif)# **backup interface type** *number*	Configures backup interface for the subinterface.
Step 6	Router(config-subif)# **backup delay** *enable-delay disable-delay*	Specifies backup enable and disable delay.

Disabling or Reenabling Frame Relay Inverse ARP

To select or disable Inverse ARP, use one of the following commands in interface configuration mode:

Command	Purpose
frame-relay inverse-arp *protocol dlci*	Enables Frame Relay Inverse ARP for a specific protocol and DLCI pair, only if it was previously disabled.
no frame relay inverse-arp *protocol dlci*	Disables Frame Relay Inverse ARP for a specific protocol and DLCI pair.

Creating a Broadcast Queue for an Interface

To create a broadcast queue, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)#` **frame-relay broadcast-queue** *size byte-rate packet-rate*	Creates a broadcast queue for an interface.

Configuring Frame Relay Fragmentation

Configuring End-to-End FRF.12 Fragmentation

To configure FRF.12 fragmentation in a Frame Relay map class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **map-class frame-relay** *map-class-name*
2. Router(config-map-class)# **frame-relay fragment** *fragment_size*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **map-class frame-relay** *map-class-name*	Specifies a map class to define QoS values for a Frame Relay SVC or PVC. The map class can be applied to one or many PVCs.
Step 2	Router(config-map-class)# **frame-relay fragment** *fragment_size*	Configures Frame Relay fragmentation for the map class. The *fragment_size* argument defines the payload size of a fragment; it excludes the Frame Relay headers and any Frame Relay fragmentation header. The valid range is from 16 to 1600 bytes, and the default is 53.

Verifying the Configuration of End-to-End FRF.12 Fragmentation

To verify FRF.12 fragmentation, use one or more of the following EXEC commands:

Command	Purpose
show frame-relay fragment [**interface** *interface*] [*dlci*]	Displays Frame Relay fragmentation information.
show frame-relay pvc [**interface** *interface*] [*dlci*]	Displays statistics about PVCs for Frame Relay interfaces.

Configuring Payload Compression

Configuring Payload Compression On a Multipoint Interface or Subinterface

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

Command	Purpose
`Router(config-if)#` **frame-relay map** *protocol* *protocol-address* *dlci* **payload-compression packet-by-packet**	Enables payload compression on a multipoint interface.

Configuring Payload Compression On a Point-to-Point Interface or Subinterface

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

Command	Purpose
`Router(config-if)#` **frame-relay payload-compression packet-by-packet**	Enables payload compression on a point-to-point interface.

Configuring FRF.9 Compression Using Map Statements

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

SUMMARY STEPS

1. Router(config)# **interface** *type number*
2. Router(config-if)# **encapsulation frame-relay**
3. Router(config-if)# **frame-relay map payload-compression frf9 stac**[*hardware-options*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number*	Specifies the interface.
Step 2	Router(config-if)# **encapsulation frame-relay**	Specifies Frame Relay as encapsulation type.
Step 3	Router(config-if)# **frame-relay map payload-compression frf9 stac**[*hardware-options*]	Enables FRF.9 compression.

Configuring FRF.9 Compression on the Subinterface

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

SUMMARY STEPS

1. Router(config)# **interface** *type number*
2. Router(config-subif)# **encapsulation frame-relay**
3. Router(config-subif)# **frame-relay payload-compression frf9 stac**[*hardware-options*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number*	Specifies the subinterface type and number.
Step 2	Router(config-subif)# **encapsulation frame-relay**	Specifies Frame Relay as encapsulation type.
Step 3	Router(config-subif)# **frame-relay payload-compression frf9 stac**[*hardware-options*]	Enables FRF.9 compression.

Configuring Data-Stream Hardware Compression and IP Header Compression on a Point-to-Point Subinterface

To configure data-stream hardware compression and TCP or Real-Time Transport Protocol (RTP) header compression on a point-to-point subinterface, use the following commands beginning in global configuration mode. Note that when you specify data-stream hardware compression, Cisco-proprietary encapsulation is automatically enabled.

SUMMARY STEPS

1. Router(config)# **interface** *type number* **point-to-point**
2. Router(config-subif)# **ip address** *address mask*
3. Router(config-subif)# **frame-relay interface-dlci** *dlci*
4. Router(config-subif)# **frame-relay payload-compression data-stream stac** [*hardware-options*
5. Do one of the following:

   - Router(config-subif)# **frame-relay ip tcp header-compression** [**passive**]
   -
   -
   -
   -

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number* **point-to-point**	Configures a subinterface type and enters subinterface configuration mode.
Step 2	Router(config-subif)# **ip address** *address mask*	Sets the IP address for an interface.
Step 3	Router(config-subif)# **frame-relay interface-dlci** *dlci*	Assigns a DLCI to a specified Frame Relay subinterface on the router or access server.

	Command or Action	Purpose
Step 4	Router(config-subif)# **frame-relay payload-compression data-stream stac** [*hardware-options*	Enables hardware compression on an interface or subinterface that uses Cisco-proprietary encapsulation.
Step 5	Do one of the following: • Router(config-subif)# **frame-relay ip tcp header-compression** [**passive**] • • • • **Example:** `Router(config-subif)#` **frame-relay ip rtp header-compression** [**passive**]	Configures an interface to ensure that the associated PVCs carry outgoing TCP headers in compressed form. Enables RTP header compression on the physical interface.

Configuring Data-Stream Hardware Compression and IP Header Compression on a Multipoint Subinterface

To configure data-stream hardware compression and TCP or RTP header compression on a multipoint subinterface, use the following commands beginning in global configuration mode. Note that when you specify data-stream hardware compression, Cisco-proprietary encapsulation is automatically enabled.

SUMMARY STEPS

1. Router(config)# **interface** *type number* **multipoint**
2. Router(config-subif)# **frame-relay interface-dlci** dlci
3. Router(config-subif)# **frame-relay map** protocol protocol-address dlci [**payload-compression data-stream stac** [*hardware-options*]]
4. Do one of the following:
   - Router(config-subif)# **frame-relay ip tcp header-compression** [**passive**]
   -
   -
   -
   -

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number* **multipoint**	Configures a subinterface type and enters subinterface configuration mode.
Step 2	Router(config-subif)# **frame-relay interface-dlci** dlci	Assigns a DLCI to a specified Frame Relay subinterface on the router or access server.
Step 3	Router(config-subif)# **frame-relay map** protocol protocol-address dlci [**payload-compression data-stream stac** [*hardware-options*]]	Defines the mapping between a destination protocol address and the DLCI used to connect to the destination address on an interface that uses Cisco-proprietary encapsulation.
Step 4	Do one of the following: • Router(config-subif)# **frame-relay ip tcp header-compression** [**passive**] • • • • **Example:** `Router(config-subif)#` **frame-relay ip rtp header-compression** `[`**passive**`]`	Configures an interface to ensure that the associated PVCs carry outgoing TCP headers in compressed form. Enables RTP header compression on the physical interface.

Verifying Payload Compression

To verify that payload compression is working correctly, use the following privileged EXEC commands:

Command	Purpose
`Router#` **show compress**	Displays compression statistics.
`Router#` **show frame-relay pvc** `dlci`	Displays statistics about PVCs for Frame Relay interfaces, including the number of packets in the post-hardware-compression queue.
`Router#` **show traffic-shape queue**	Displays information about the elements queued at a particular time at the DLCI level, including the number of packets in the post-hardware- compression queue.

Configuring TCP IP Header Compression

Configuring an Individual IP Map for TCP IP Header Compression

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

Command	Purpose
frame-relay map ip *ip-address dlci* [**broadcast**] **tcp header-compression** [**active** | **passive**] [**connections** *number*]	Configures an IP map to use TCP/IP header compression. Cisco-proprietary encapsulation is enabled by default.

Configuring an Interface for TCP IP Header Compression

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

SUMMARY STEPS

1. Router(config-if)# **encapsulation frame-relay**
2. Router(config-if)# **frame-relay ip tcp header-compression** [**passive**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config-if)# **encapsulation frame-relay**	Configures Cisco-proprietary encapsulation on the interface.
Step 2	Router(config-if)# **frame-relay ip tcp header-compression** [**passive**]	Enables TCP/IP header compression.

Disabling 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, use one of the following commands in interface configuration mode:

Command	Purpose
no frame-relay ip tcp header-compression	Disables TCP/IP header compression on all Frame Relay IP maps that are not explicitly configured for TCP header compression.
frame-relay map ip *ip-address dlci* **nocompress**	Disables RTP and TCP/IP header compression on a specified Frame Relay IP map.

Configuring Discard Eligibility

Defining a DE List

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

SUMMARY STEPS

1. Router(config)# **frame-relay de-list** *list-number* {**protocol** *protocol* | **interface** *type number*} *characteristic*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **frame-relay de-list** *list-number* {**protocol** *protocol* | **interface** *type number*} *characteristic*	Defines a DE list.

Defining a DE Group

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

Command	Purpose
frame-relay de-group *group-number* `dlci`	Defines a DE group.

Configuring DLCI Priority Levels

To configure DLCI priority levels, use the following command in interface configuration mode:

Command	Purpose
frame-relay priority-dlci-group *group-number high-dlci medium-dlci normal-dlci low-dlci*	Enables multiple parallel DLCIs for different Frame Relay traffic types; associates and sets level of specified DLCIs with same group. **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.

Monitoring and Maintaining the Frame Relay Connections

To monitor Frame Relay connections, use any of the following commands in EXEC mode:

Command	Purpose
clear frame-relay-inarp	Clears dynamically created Frame Relay maps, which are created by the use of Inverse ARP.
show interfaces serial *type number*	Displays information about Frame Relay DLCIs and the LMI.
show frame-relay lmi [*type number*]	Displays LMI statistics.
show frame-relay map	Displays the current Frame Relay map entries.
show frame-relay pvc [*type number* [*dlci*]]	Displays PVC statistics.
show frame-relay route	Displays configured static routes.
show frame-relay traffic	Displays Frame Relay traffic statistics.
show frame-relay lapf	Displays information about the status of LAPF.
show frame-relay svc maplist	Displays all the SVCs under a specified map list.

Configuration Examples for Frame Relay

Example IETF Encapsulation

Example IETF Encapsulation on the Interface

The following example sets IETF encapsulation at the interface level. 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
```

Example IETF Encapsulation on a Per-DLCI Basis

The following example configures IETF encapsulation 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
```

Example Static Address Mapping

Example Two Routers in Static Mode

The following example shows how to configure two routers for static mode:

Configuration for Router 1

```
interface serial0
 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 serial1
 ip address 131.108.64.1 255.255.255.0
 encapsulation frame-relay
 keepalive 10
 frame-relay map ip 131.108.64.2 43
```

Example AppleTalk Routing

The following example shows 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
```

Example DECnet Routing

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
```

Example IPX Routing

The following example shows 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
```

Example Subinterface

Example Basic Subinterface

In the following example, subinterface 1 is configured as a point-to-point subnet and subinterface 2 is configured as a multipoint subnet.

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

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

Example Frame Relay Multipoint Subinterface with Dynamic Addressing

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
```

Example IPX Routes over Frame Relay Subinterfaces

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
```

Example Unnumbered IP over a Point-to-Point Subinterface

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

```
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
```

Example Transparent Bridging Using Subinterfaces

The following example shows Frame Relay DLCIs 42, 64, and 73 as separate point-to-point links with transparent bridging running over them. The bridging spanning tree views each PVC as a separate bridge port, and a frame arriving on the PVC can be relayed back out on a separate PVC.

```
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
```

Example SVC Configuration

Example SVC Interface

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
```

Example SVC Subinterface

The following example configures a point-to-point interface for SVC operation. It 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
```

Example Frame Relay Traffic Shaping

Example Traffic Shaping with Three Point-to-Point Subinterfaces

In the following example, VCs on subinterfaces Serial0.1 and Serial0.2 inherit class parameters from the main interface--namely, those defined in the map class "slow_vcs"--but the VC 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, the output rate will be cut back to as low as 2400 bps in response to received BECNs. This map class is configured to use custom queueing 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, the output rate will be cut back to as low as 8000 bps in response to received BECNs. This map class is configured to use priority-queueing 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
 frame-relay adaptive-shaping becn
!
map-class frame-relay fast_vcs
 frame-relay traffic-rate 16000 64000
 frame-relay priority-group 2
 frame-relay adaptive-shaping becn
!
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
```

Example Traffic Shaping with ForeSight

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
```

Example LMI Configuration

Example ELMI and Frame Relay Traffic Shaping

The following configuration shows a Frame Relay interface enabled with QoS autosense. The router receives messages from the Cisco switch, which is also configured with QoS autosense enabled. When ELMI is configured in conjunction with traffic shaping, the router will receive congestion information through BECN or router ForeSight congestion signalling 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
```

Example Configuring the IP Address for ELMI Address Registration

The following example shows how to configure the IP address to be used for ELMI address registration. Automatic IP address selection is automatically disabled when the IP address is configured. ELMI is enabled on serial interface 0.

```
interface Serial 0
 no ip address
 encapsulation frame-relay
  frame-relay lmi-type ansi
  frame-relay qos-autosense
!
frame-relay address registration ip address 139.85.242.195
!
```

Example Disabling ELMI Address Registration on an Interface

In the following example, ELMI address registration is disabled on serial interface 0. This interface will share an IP address of 0.0.0.0 and an ifIndex of 0. Automatic IP address selection is enabled by default when ELMI is enabled, so the management IP address of other interfaces on this router will be chosen automatically.

```
interface Serial 0
 no ip address
 encapsulation frame-relay
  frame-relay lmi-type ansi
  frame-relay qos-autosense
  no frame-relay address-reg-enable
!
```

Example Backward Compatibility

The following configuration provides backward compatibility and interoperability with versions 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
```

Example Booting from a Network Server over Frame Relay

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

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 be configured with the following command:

```
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.

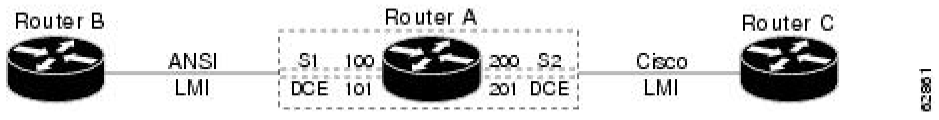
Example Frame Relay Switching

Example PVC Switching Configuration

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 figure below, the router switches two PVCs between serial interfaces 1 and 2. Frames with DLCI 100 received on serial 1 will be transmitted with DLCI 200 on serial 2.

Figure 10: PVC Switching Configuration



The following example shows one router with two interfaces configured as DCEs. The router switches frames from the incoming interface to the outgoing interface on the basis of the DLCI alone.

Configuration for Router A

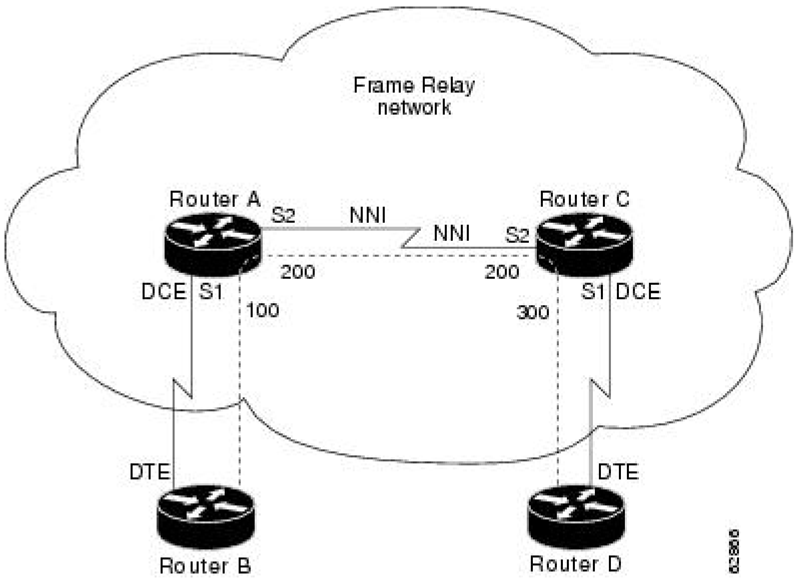
```
frame-relay switching
interface Serial1
 no ip address
 encapsulation frame-relay
 keepalive 15
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 100 interface Serial2 200
 frame-relay route 101 interface Serial2 201
 clockrate 2000000
!
interface Serial2
 encapsulation frame-relay
 keepalive 15
 frame-relay intf-type dce
 frame-relay route 200 interface Serial1 100
 frame-relay route 201 interface Serial1 101
 clockrate 64000
```

Example Pure Frame Relay DCE

Using the PVC switching feature, it is possible to build an entire Frame Relay network using routers. In the figure below, router A and router C act as Frame Relay switches implementing a two-node network. The standard Network-to-Network Interface (NNI) signalling protocol is used between router A and router C.

The following example shows a Frame Relay network with two routers functioning as switches and standard NNI signalling used between them.

Figure 11: Frame Relay DCE Configuration



Configuration for Router A

```
frame-relay switching
!
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
!
```

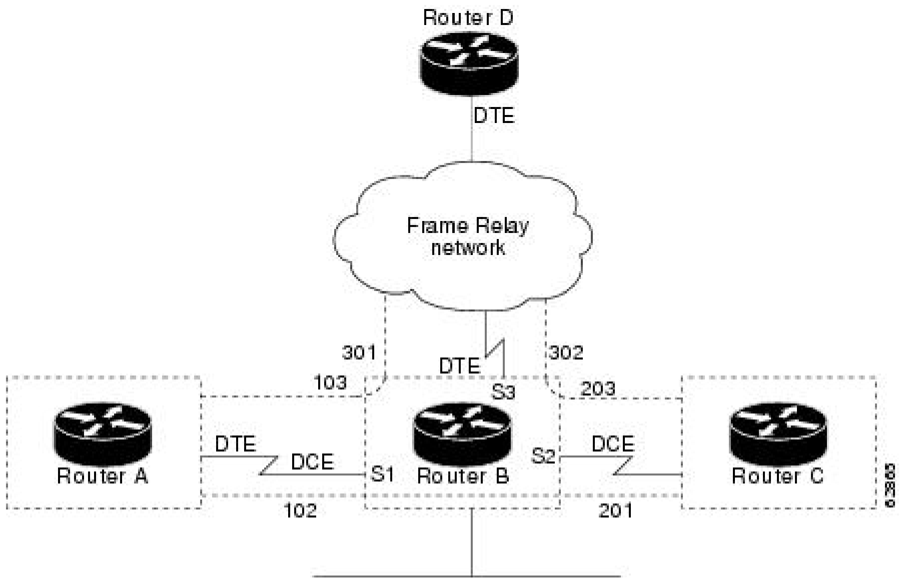
Configuration for Router C

```
frame-relay switching
!
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
!
```

Example Hybrid DTE DCE PVC Switching

Routers can be configured as hybrid DTE/DCE Frame Relay switches, as shown in the figure below.

Figure 12: Hybrid DTE/DCE PVC Switching



The following example shows one router configured with both DCE and DTE interfaces (router B acts as a hybrid DTE/DCE Frame Relay switch). It can switch frames between 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
```

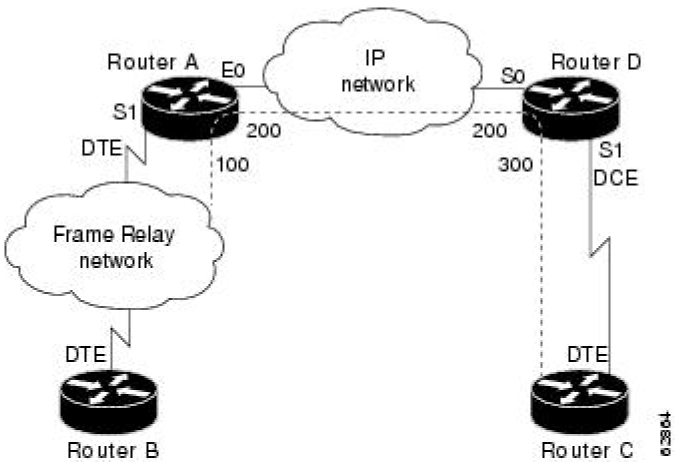
```
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
 frame-relay route 102 interface serial 2 201
 frame-relay route 103 interface serial 3 301
!
interface serial 2
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
 frame-relay route 201 interface serial 1 102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

Example Switching over an IP Tunnel

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, as shown in the figure below.

Note Static routes cannot be configured over tunnel interfaces on the Cisco 800 series, 1600 series, and 1700 series platforms. Static routes can only be configured over tunnel interfaces on platforms that have the Enterprise feature set.

Figure 13: Frame Relay Switch over IP Tunnel



The following example shows two routers configured to switch Frame Relay PVCs over a point-to-point IP tunnel, which is the IP network configuration depicted in the figure above.

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 Tunnel1 200
!
interface tunnel1
 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 tunnel1
 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 Tunnel1 200
```

Example Frame Relay Switching over ISDN B Channels

The following example illustrates Frame Relay switching over an ISDN dialer interface:

```
frame-relay switching
  !
  interface BRI0
    isdn switch-type basic-5ess
    dialer pool-member 1
    dialer pool-member 2
  !
  interface dialer1
    encapsulation frame-relay
    dialer pool 1
    dialer-group 1
    dialer caller 60038
    dialer string 60038
    frame-relay intf-type dce
  !
  interface dialer2
    encapsulation frame-relay
    dialer pool 2
    dialer-group 1
    dialer caller 60039
    dialer string 60039
    frame-relay intf-type dce
  !
  interface serial0
    encapsulation frame-relay
    frame-relay intf-type dce
  !
  connect one serial0 16 dialer1 100
  connect two serial0 17 dialer2 100
  dialer-list 1 protocol ip permit
```

Note Note that when Frame Relay switching is performed by using a dialer profile, encapsulation of the underlying physical (BRI) interface must be configured as high-level data link control (HDLC).

Example Traffic Shaping on Switched PVCs

In the example that follows, traffic on serial interface 0 is being shaped prior to entry to the Frame Relay network. PVC 100/16 is shaped according to the "shape256K" class. PVC 200/17 is shaped using the "shape64K" class inherited from the interface.

```
frame-relay switching
  !
  interface serial0
    encapsulation frame-relay
    frame-relay intf-type dce
    frame-relay traffic-shaping
    frame-relay class shape64K
    frame-relay interface-dlci 16 switched
      class shape256K
  !
  interface serial1
    encapsulation frame-relay
    frame-relay intf-type dce
  !
  connect one serial0 16 serial1 100
  connect two serial0 17 serial1 200
  !
  map-class frame-relay shape256K
    frame-relay traffic-rate 256000 512000
  !
  map-class frame-relay shape64K
    frame-relay traffic-rate 64000 64000
```

Example Traffic Policing on a UNI DCE

In the following example, incoming traffic is being policed on serial interface 1. The interface uses policing parameters configured in map class "police256K". PVC 100/16 inherits policing parameters from the interface. PVC 200/17 uses policing parameters configured in "police64K".

```
frame-relay switching
  !
  interface serial0
    encapsulation frame-relay
    frame-relay intf-type dce
  !
  interface serial1
    encapsulation frame-relay
    frame-relay policing
    frame-relay class police256K
    frame-relay intf-type dce
    frame-relay interface-dlci 200 switched
```

```
      class police64K
  !
  connect one serial0 16 serial1 100
  connect two serial0 17 serial1 200
  !
  map-class frame-relay police256K
    frame-relay cir 256000
    frame-relay bc 256000
    frame-relay be 0
  !
  map-class frame-relay police64K
    frame-relay cir 64000
    frame-relay bc 64000
    frame-relay be 64000
```

Example Congestion Management on Switched PVCs

The following example illustrates the configuration of congestion management and DE discard levels for all switched PVCs on serial interface 1. Policing is configured on PVC 16.

```
frame-relay switching
  !
  interface serial0
    encapsulation frame-relay
    frame-relay intf-type dce
    frame-relay policing
    frame-relay interface-dlci 16 switched
      class 256K
  !
  interface serial1
    encapsulation frame-relay
    frame-relay intf-type dce
    frame-relay congestion-management
      threshold ecn be 0
      threshold ecn bc 20
      threshold de 40
  !
  connect one serial1 100 serial0 16
  !
  map-class frame-relay 256K
    frame-relay cir 256000
    frame-relay bc 256000
    frame-relay be 256000
```

Example Congestion Management on the Traffic-Shaping Queue of a Switched PVC

The following example illustrates the configuration of congestion management in a class called "perpvc_congestion". The class is associated with the traffic-shaping queue of DLCI 200 on serial interface 3.

```
  map-class frame-relay perpvc_congestion
    frame-relay holdq 100
    frame-relay congestion threshold ecn 50
  interface Serial3
    frame-relay traffic-shaping
    frame-relay interface-dlci 200 switched
      class perpvc_congestion
```

Example FRF.12 Fragmentation on a Switched PVC Configuration

In the following example, FRF.12 fragmentation is configured in a map class called "data". The "data" map class is assigned to switched pvc 20 on serial interface 3/3.

```
frame-relay switching
!
interface Serial3/2
 encapsulation frame-relay
 frame-relay intf-type dce
!
interface Serial3/3
 encapsulation frame-relay
 frame-relay traffic-shaping
 frame-relay interface-dlci 20 switched
  class data
 frame-relay intf-type dce
!
map-class frame-relay data
 frame-relay fragment 80 switched
 frame-relay cir 64000
 frame-relay bc 640
!
connect data Serial3/2 16 Serial3/3 20
```

Example Frame Relay End-to-End Keepalive

Example End-to-End Keepalive Bidirectional Mode with Default Configuration

In the following example, the devices at each end of a VC are configured so that a DLCI is assigned to a Frame Relay serial interface, a map class is associated with the interface, and Frame Relay end-to-end keepalive is configured in bidirectional mode using default values:

```
! router1
router1(config) interface serial 0/0.1 point-to-point
router1(config-if) ip address 10.1.1.1 255.255.255.0
router1(config-if) frame-relay interface-dlci 16
router1(config-if) frame-relay class vcgrp1
router1(config-if) exit
!
router1(config)# map-class frame-relay vcgrp1
router1(config-map-class)# frame-relay end-to-end keepalive mode bidirectional
! router2
router2(config) interface serial 1/1.1 point-to-point
router2(config-if) ip address 10.1.1.2 255.255.255.0
router2(config-if) frame-relay interface-dlci 16
router2(config-if) frame-relay class vceek
router1(config-if) exit
!
router2(config)# map-class frame-relay vceek
router2(config-map-class)# frame-relay end-to-end keepalive mode bidirectional
```

Example End-to-End Keepalive Request Mode with Default Configuration

In the following example, the devices at each end of a VC are configured so that a DLCI is assigned to a Frame Relay serial interface and a map class is associated with the interface. One device is configured in request mode while the other end of the VC is configured in reply mode.

```
! router1
router1(config) interface serial 0/0.1 point-to-point
router1(config-if) ip address 10.1.1.1 255.255.255.0
router1(config-if) frame-relay interface-dlci 16
router1(config-if) frame-relay class eek
router1(config-if) exit
!
router1(config)# map-class frame-relay eek
router1(config-map-class)# frame-relay end-to-end keepalive mode request

! router2
router2(config) interface serial 1/1.1 point-to-point
router2(config-if) ip address 10.1.1.2 255.255.255.0
router2(config-if) frame-relay interface-dlci 16
router2(config-if) frame-relay class group_3
router1(config-if) exit
!
router2(config)# map-class frame-relay group_3
router2(config-map-class)# frame-relay end-to-end keepalive mode reply
```

Example End-to-End Keepalive Request Mode with Modified Configuration

In the following example, the devices at each end of a VC are configured so that a DLCI is assigned to a Frame Relay serial interface and a map class is associated with the interface. One device is configured in request mode while the other end of the VC is configured in reply mode. The event window, error threshold, and success events values are changed so that the interface will change state less frequently:

```
! router1
router1(config) interface serial 0/0.1 point-to-point
router1(config-if) ip address 10.1.1.1 255.255.255.0
router1(config-if) frame-relay interface-dlci 16
router1(config-if) frame-relay class eek
router1(config-if) exit
!
router1(config)# map-class frame-relay eek
router1(config-map-class)# frame-relay end-to-end keepalive mode request
router1(config-map-class)# frame-relay end-to-end keepalive event-window send 5
router1(config-map-class)# frame-relay end-to-end keepalive error-threshold send 3
router1(config-map-class)# frame-relay end-to-end keepalive success-events send 3
! router2
router2(config) interface serial 1/1.1 point-to-point
router2(config-if) ip address 10.1.1.2 255.255.255.0
router2(config-if) frame-relay interface-dlci 16
router2(config-if) frame-relay class group_3
router1(config-if) exit
!
router2(config)# map-class frame-relay group_3
router2(config-map-class)# frame-relay end-to-end keepalive mode reply
```

Example PPPoverFrameRelay

Example PPP over Frame Relay DTE

The following example configures a router as a DTE device for PPP over Frame Relay. Subinterface 2.1 contains the necessary DLCI and virtual template information. Interface Virtual-Template 1 contains the PPP information that is applied to the PPP session associated with DLCI 32 on serial subinterface 2.1.

```
interface serial 2
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
interface serial 2.1 point-to-point
 frame-relay interface-dlci 32 ppp virtual-template1
!
interface Virtual-Template1
 ip unnumbered ethernet 0
 ppp authentication chap pap
```

Note By default, the encapsulation type for a virtual template interface is PPP encapsulation; therefore, **encapsulation ppp**will not appear when you view the configuration of the router.

Example PPP over Frame Relay DCE

The following example configures a router to act as a DCE device. Typically, a router is configured as a DCE if it is connecting directly to another router or if connected to a 90i D4 channel unit, which is connected to a telco channel bank. The three commands required for this type of configuration are the **frame-relay switching, frame-relay intf-type dce,** and **frame-relay route** commands:

```
frame-relay switching
!
interface Serial2/0:0
 no ip address
 encapsulation frame-relay IETF
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 31 interface Serial1/2 100
 frame-relay interface-dlci 32 ppp Virtual-Template1
!
interface Serial2/0:0.2 point-to-point
 no ip address
 frame-relay interface-dlci 40 ppp Virtual-Template2
!
interface Virtual-Template1
 ip unnumbered Ethernet0/0
 peer default ip address pool default
 ppp authentication chap pap
 !
interface Virtual-Template2
 ip address 100.1.1.2 255.255.255.0
 ppp authentication chap pap
```

Note By default, the encapsulation type for a virtual template interface is PPP encapsulation; therefore, **encapsulation ppp**will not appear when you view the configuration of the router.

Example Frame Relay Fragmentation Configuration

Example FRF.12 Fragmentation

The following example shows the configuration of pure end-to-end FRF.12 fragmentation and weighted fair queueing in the map class called "frag". The fragment payload size is set to 40 bytes. The "frag" map class is associated with DLCI 100 on serial interface 1.

```
router(config)#
interface serial 1

router(config-if)# frame-relay interface-dlci 100
router(config-fr-dlci)# class frag
router(config-fr-dlci)# exit
router(config)# map-class frame-relay frag
router(config-map-class)# frame-relay fragment 40
```

Example Frame Relay Fragmentation with Hardware Compression

In the following example, FRF.12 fragmentation and FRF.9 hardware compression are configured on multipoint interface 3/1 and point-to-point interface 3/1.1:

```
interface serial3/1
 ip address 10.1.0.1 255.255.255.0
 encapsulation frame-relay
 frame-relay traffic-shaping
 frame-relay class frag
 frame-relay map ip 10.1.0.2 110 broadcast ietf payload-compression frf9 stac
!
interface serial3/1.1 point-to-point
 ip address 10.2.0.1 255.255.255.0
 frame-relay interface-dlci 120 ietf
 frame-relay payload-compression frf9 stac
!
map-class frame-relay frag
 frame-relay cir 64000
 frame-relay bc 640
 frame-relay fragment 100
```

Example Payload Compression Configuration

Note Shut down the interface or subinterface prior to adding or changing compression techniques. Although shutdown is not required, shutting down the interface ensures that it is reset for the new data structures.

Example FRF.9 Compression for Subinterfaces Using the frame-relaymap Command

The following example shows a subinterface being configured for FRF.9 compression using the **frame-relay map** command:

```
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
```

Example FRF.9 Compression for Subinterfaces

The following example shows a subinterface being configured for FRF.9 compression:

```
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
```

Example Data-Stream Hardware Compression with TCP IP Header Compression on a Point-to-Point Subinterface

The following example shows the configuration of data-stream hardware compression and TCP header compression on point-to-point interface 1/0.1:

```
interface serial1/0
  encapsulation frame-relay
  frame-relay traffic-shaping
 !
 interface serial1/0.1 point-to-point
  ip address 10.0.0.1 255.0.0.0
  frame-relay interface-dlci 100
  frame-relay payload-compression data-stream stac
  frame-relay ip tcp header-compression
```

Example Data-Stream Hardware Compression with TCP IP Header Compression on a Multipoint Subinterface

The following example shows the configuration of data-stream hardware compression and TCP header compression on multipoint interface 3/1:

```
interface serial3/1
 ip address 10.1.0.1 255.255.255.0
 encapsulation frame-relay
 frame-relay traffic-shaping
 frame-relay map ip 10.1.0.2 110 broadcast cisco payload-compression data-stream stac
 frame-relay ip tcp header-compression
```

Example Data-Stream Hardware Compression with RTP Header Compression and Frame Relay Fragmentation

The following example shows the configuration of data-stream hardware compression, RTP header compression, and FRF.12 fragmentation on point-to-point interface 1/0.1:

```
interface serial1/0
  encapsulation frame-relay
  frame-relay traffic-shaping
 !
 interface serial1/0.1 point-to-point
  ip address 10.0.0.1 255.0.0.0
  frame-relay interface-dlci 100
  frame-relay class frag
  frame-relay payload-compression data-stream stac
  frame-relay ip rtp header-compression
 !
 map-class frame-relay frag
  frame-relay cir 64000
  frame-relay bc 640
  frame-relay be 0
  frame-relay fragment 100
  frame-relay ip rtp priority 16000 16000 20
```

Example TCP IP Header Compression

Example IP Map with Inherited TCP IP Header Compression

Note Shut down the interface or subinterface prior to adding or changing compression techniques. Although shutdown is not required, shutting down the interface ensures that it is reset for the new data structures.

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.

Example Using an IP Map to Override TCP IP Header Compression

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:

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

Example Disabling Inherited TCP IP Header Compression

In this example, following is the initial configuration:

```
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
```

Enter the following commands to enable inherited TCP/IP header compression:

```
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.

Example Disabling Explicit TCP IP Header Compression

In this example, the initial configuration is the same as in the preceding example, but you must enter the following set of commands to enable explicit TCP/IP header compression:

```
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.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS Wide-Area Networking configuration tasks	*Cisco IOS XE Wide-Area Networking Configuration Guide*
Wide-Area networking commands	*Cisco IOS Wide-Area Networking Command Reference*
Sending DDR traffic over Frame Relay	• Configuring Legacy DDR Spokes • Configuring Legacy DDR Hubs
Installing software on a new router or access server by downloading from a central server over an interface that supports Frame Relay	Loading and Maintaining System Images
Using AutoInstall over Frame Relay	Overview - Basic Configuration of a Cisco Networking Device
Configuring transparent bridging between devices over a Frame Relay network	Configuring Transparent Bridging
Configuring source-route bridging between SNA devices over a Frame Relay network	Configuring Source-Route Bridging
Configuring serial tunnel (STUN) and block serial tunnel encapsulation between devices over a Frame Relay network	Configuring Serial Tunnel and Block Serial Tunnel
Configuring access between SNA devices over a Frame Relay network	Configuring SNA Frame Relay Access Support
Configuring Voice over Frame Relay Using FRF.11 and FRF.12	Configuring Voice over Frame Relay

Related Topic	Document Title
Configuring low latency queueing, PVC interface priority queueing, and link fragmentation and interleaving using multilink PPP for Frame Relay	*Cisco IOS Quality of Service Solutions Configuration Guide*

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

CHAPTER 3

Frame Relay 64-Bit Counters

Feature History

Release	Modification
12.0(17)S	This feature was introduced on the Cisco 12000 series.
12.2(4)T	This feature was integrated into Cisco IOS Release 12.2(4)T.
12.2(4)T3	Support for the Cisco 7500 series routers was added.
12.0(21)S	The **frame-relay ifmib-counter64** command was introduced.
12.3(10)	The **frame-relay ifmib-counter64** command was integrated into Cisco IOS Release 12.3(10).
12.3(11)T	The **frame-relay ifmib-counter64** command was integrated into Cisco IOS Release 12.3(11)T.
12.2(18)SXE	The **frame-relay ifmib-counter64** command was integrated into Cisco IOS Release 12.2(18)SXE.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Feature Overview

The Frame Relay 64-Bit Counters feature provides 64-bit counter support on Frame Relay interfaces and subinterfaces. This feature enables the gathering of statistics through Simple Network Management Protocol (SNMP) for faster interfaces operating at OC-3, OC-12, and OC-48 speeds.

The following counters are supported by this feature: Bytes In, Bytes Out, Packets In, and Packets Out.

The **show frame-relay pvc** command has been modified to display the 64-bit counters.

Benefits

The values in 32-bit counters sometime wrap because the field is too small. Wrapping causes the values in these fields to become meaningless. The 64-bit counters support the reliable gathering of statistics by SNMP by preventing the wrapping of counter values.

Restrictions

SNMP cannot retrieve 64-bit virtual-circuit (VC) counters.

Related Documents

For information on configuring Frame Relay using Cisco IOS software, refer to the following documents:

- The chapter " Configuring Frame Relay " in the *Cisco IOS Wide-Area Networking Configuration Guide* , Release 12.2
- The chapter " Frame Relay Commands " in the *Cisco IOS Wide-Area Networking Command Reference* , Release 12.2

For information on configuring SNMP using Cisco IOS software, see the following documents:

- The chapter " Configuring Simple Network Management Protocol " in the *Cisco IOS Configuration Fundamentals Configuration Guide* , Release 12.2

- The chapter " SNMP Commands " in the *Cisco IOS Configuration Fundamentals Command Reference* , Release 12.2

Supported Platforms

- Cisco 7200 series
- Cisco 7500 series (Cisco IOS Release 12.2(4)T3 and later)

Platform Support Through Feature Navigator

Cisco IOS software is packaged in feature sets that support specific platforms. To get updated information regarding platform support for this feature, access Feature Navigator. Feature Navigator dynamically updates the list of supported platforms as new platform support is added for the feature.

Feature Navigator is a web-based tool that enables you to quickly determine which Cisco IOS software images support a specific set of features and which features are supported in a specific Cisco IOS image.

To access Feature Navigator, you must have an account on Cisco.com. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you. Qualified users can establish an account on Cisco.com by following the directions at http://www.cisco.com/register.

Feature Navigator is updated when major Cisco IOS software releases and technology releases occur. As of May 2001, Feature Navigator supports M, T, E, S, and ST releases. You can access Feature Navigator at the following URL:

http://www.cisco.com/go/fn

Supported Standards and MIBs and RFCs

Standards

No new or modified standards are supported by this feature.

MIBs

The **frame-relay ifmib-counter64**command modifies the interface MIB (IF-MIB) by allowing slower Frame Relay interfaces and subinterfaces to be included in the 64-bit interface MIB counters.

To obtain lists of supported MIBs by platform and Cisco IOS release, and to download MIB modules, go to the Cisco MIB website on Cisco.com at the following URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

No new or modified RFCs are supported by this feature.

Prerequisites

This document assumes that you know how to configure Frame Relay and SNMP support using Cisco IOS software.

Configuration Tasks

Enabling Frame Relay Interfaces for 64-Bit Interface MIB Counters

Note This task is supported in Cisco IOS releases 12.0(21)S, 12.3(10), 12.3(11)T, 12.2(18)SXE, and later releases.

Frame Relay interfaces and subinterfaces that have a line speed greater than 20 Mbps are included in the 64-bit interface MIB counters by default. Perform this task to enable Frame Relay interfaces and subinterfaces that have a line speed of less than 20 Mbps to be included in the 64-bit interface MIB counters.

SUMMARY STEPS

1. Router> **enable**
2. Router# **configure terminal**
3. Router(config)# **interface serial** *interface-number*
4. Router(config-if)# **encapsulation frame-relay**
5. Router(config-if)# **frame-relay ifmib-counter64** [**if** | **subif**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router> **enable**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	Router# **configure terminal**	Enters global configuration mode.
Step 3	Router(config)# **interface serial** *interface-number*	Specifies an interface to be configured and enters interface configuration mode.
Step 4	Router(config-if)# **encapsulation frame-relay**	Enables Frame Relay encapsulation.
Step 5	Router(config-if)# **frame-relay ifmib-counter64** [**if** | **subif**]	Enables Frame Relay interfaces and subinterfaces that have a line speed of less than 20 Mbps to be included in 64-bit interface MIB counters.

	Command or Action	Purpose
		• This command allows Frame Relay interfaces and subinterfaces that have a line speed of less than 20 Mbps to be included in the following 64-bit interface MIB counters: • ifHCinOctets • ifHCOutOctets • ifHCinUcastPkts • ifHCOutUcastPkts

Monitoring and Maintaining Frame Relay 64-Bit Counters

To view the values of the Frame Relay 64-bit counters, use the following command in EXEC mode:

Command	Purpose
`Router# show frame-relay pvc 64-bit [interface interface ] [ dlci ]`	Displays statistics about permanent virtual circuits (PVCs) for Frame Relay interfaces.

Configuration Examples

Enabling Slower Frame Relay Interfaces and Subinterfaces for 64-Bit Interface MIB Counters Example

In the following example, the **frame-relay ifmib-counter64** command is used with the **if** keyword to enable serial interfaces 6/0/1:0, 6/0/2:0, and 6/0/3:0 and related subinterfaces to be included in the 64-bit interface MIB counters. The example also shows corresponding output for the **show frame-relay pvc**command and the corresponding statistics for the 64-bit interface MIB counters.

```
interface Serial6/0/1:0
 ip address 1.1.1.1 255.255.255.0
 encapsulation frame-relay
 no ip route-cache cef
 no ip route-cache
 frame-relay interface-dlci 101
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 if
interface Serial6/0/2:0
 no ip address
 encapsulation frame-relay
 no ip route-cache cef
 no ip route-cache
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 if
!
interface Serial6/0/2:0.1 point-to-point
 ip address 2.1.1.1 255.255.255.0
 no ip route-cache
 frame-relay interface-dlci 201
!
interface Serial6/0/3:0
 ip address 3.1.1.1 255.255.255.0
 encapsulation frame-relay
 frame-relay interface-dlci 301
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 if
interface Serial6/0/3:0.1 point-to-point
 ip address 3.1.2.1 255.255.255.0
 frame-relay interface-dlci 302
```

The following example shows corresponding sample output for the **show frame-relay pvc** command with the **64-bit** keyword. Note that the **frame-relay ifmib-counter64**command does not affect the output of the **show frame-relay pvc**command.

```
Router# show frame-relay pvc 101 64-bit
DLCI = 101, INTERFACE = Serial6/0/1:0
  input pkts 231                    output pkts 228
  in bytes 23604                    out bytes 23502
Router#
Router# show frame-relay pvc 201 64-bit

DLCI = 201, INTERFACE = Serial6/0/2:0.1
  input pkts 1453                   output pkts 1408
  in bytes 335024                   out bytes 327272
Router#
Router# show frame-relay pvc 301 64-bit
DLCI = 301, INTERFACE = Serial6/0/3:0
  input pkts 510                    output pkts 508
  in bytes 52690                    out bytes 52622
Router#
Router# show frame-relay pvc 302 64-bit
DLCI = 302, INTERFACE = Serial6/0/3:0.1
  input pkts 957                    output pkts 912
  in bytes 283246                   out bytes 275493
Router#
```

The following output from an SNMP inquiry shows that the 64-bit interface MIB counters include the interfaces configured above:

```
ifHCInOctets.5 = 0x000000000
ifHCInOctets.16 = 0x000000000
ifHCInOctets.17 = 0x003360d33
ifHCInOctets.18 = 0x000000000
ifHCInOctets.19 = 0x000000000
ifHCInOctets.20 = 0x000000000
ifHCInOctets.24 = 0x000000000
ifHCInOctets.25 = 0x000000000
ifHCInOctets.26 = 0x0001a7afc !! This is serial interface 6/0/1:0
ifHCInOctets.28 = 0x0001a7370 !! This is serial interface 6/0/2:0
ifHCInOctets.34 = 0x00006a45a !! This is serial interface 6/0/3:0
ifHCInOctets.36 = 0x000051cb0 !! This is serial subinterface 6/0/2:0.1
ifHCInOctets.37 = 0x00004526e !! This is serial subinterface 6/0/3:0.1
```

Enabling Only Slower Frame Relay Subinterfaces for 64-Bit Interface MIB Counters Example

In the following example, the **frame-relay ifmib-counter64** command is used with the **subif** keyword to enable subinterfaces that are associated with serial interfaces 6/0/1:0, 6/0/2:0, and 6/0/3:0 to be included in the 64-bit interface MIB counters. Slower main interfaces are not included. The example also shows the corresponding statistics for the 64-bit interface MIB counters.

```
interface Serial6/0/1:0
 ip address 1.1.1.1 255.255.255.0
 encapsulation frame-relay
 no ip route-cache cef
 no ip route-cache
 frame-relay interface-dlci 101
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 subif
interface Serial6/0/2:0
 no ip address
 encapsulation frame-relay
 no ip route-cache cef
 no ip route-cache
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 subif
interface Serial6/0/2:0.1 point-to-point
 ip address 2.1.1.1 255.255.255.0
 no ip route-cache
 frame-relay interface-dlci 201
!
interface Serial6/0/3:0
 ip address 3.1.1.1 255.255.255.0
 encapsulation frame-relay
 frame-relay interface-dlci 301
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 subif
interface Serial6/0/3:0.1 point-to-point
 ip address 3.1.2.1 255.255.255.0
 frame-relay interface-dlci 302
```

The following example shows corresponding sample output for the **show frame-relay pvc** command with the **64-bit** keyword. Note that the **frame-relay ifmib-counter64**command does not affect the output of the **show frame-relay pvc** command.

```
Router# show frame-relay pvc 101 64-bit
DLCI = 101, INTERFACE = Serial6/0/1:0
  input pkts 231                    output pkts 228
  in bytes 23604                    out bytes 23502
Router#
Router# show frame-relay pvc 201 64-bit
DLCI = 201, INTERFACE = Serial6/0/2:0.1
  input pkts 1453                   output pkts 1408
  in bytes 335024                   out bytes 327272
Router#
Router# show frame-relay pvc 301 64-bit
DLCI = 301, INTERFACE = Serial6/0/3:0
  input pkts 510                    output pkts 508
  in bytes 52690                    out bytes 52622
Router#
Router# show frame-relay pvc 302 64-bit
DLCI = 302, INTERFACE = Serial6/0/3:0.1
  input pkts 957                    output pkts 912
  in bytes 283246                   out bytes 275493
```

The following output from an SNMP inquiry shows that the 64-bit interface MIB counters include the subinterfaces configured above:

```
ifHCInOctets.5 = 0x000000000
ifHCInOctets.16 = 0x000000000
ifHCInOctets.17 = 0x00337a158
ifHCInOctets.18 = 0x000000000
ifHCInOctets.19 = 0x000000000
ifHCInOctets.20 = 0x000000000
ifHCInOctets.24 = 0x000000000
ifHCInOctets.25 = 0x000000000
ifHCInOctets.36 = 0x000051cb0 !! This is serial subinterface 6/0/2:0.1
ifHCInOctets.37 = 0x00004526e !! This is serial subinterface 6/0/3:0.1
```

CHAPTER 4

Frame Relay Queueing and Fragmentation at the Interface

The Frame Relay Queueing and Fragmentation at the Interface feature introduces support for low-latency queueing (LLQ) and FRF.12 end-to-end fragmentation on a Frame Relay interface.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Frame Relay Queueing and Fragmentation at the Interface

The tasks in this document assume that you know how to configure low-latency queueing and shaping service policies.

The following prerequisites are specific to the Cisco 7500 series:

- The Frame Relay Queueing and Fragmentation at the Interface feature is supported on VIP-based interfaces with VIP2-50 or higher.
- Distributed Cisco Express Forwarding (dCEF) must be enabled both globally and on the Frame Relay interface.

Restrictions for Frame Relay Queueing and Fragmentation at the Interface

- Interface fragmentation and Frame Relay traffic shaping cannot be configured at the same time.
- Interface fragmentation and class-based fragmentation cannot be configured at the same time.
- Frame Relay switched virtual circuits (SVCs) are not supported.
- Hierarchical shaping and multiple shapers are not supported.

Information About Frame Relay Queueing and Fragmentation at the Interface

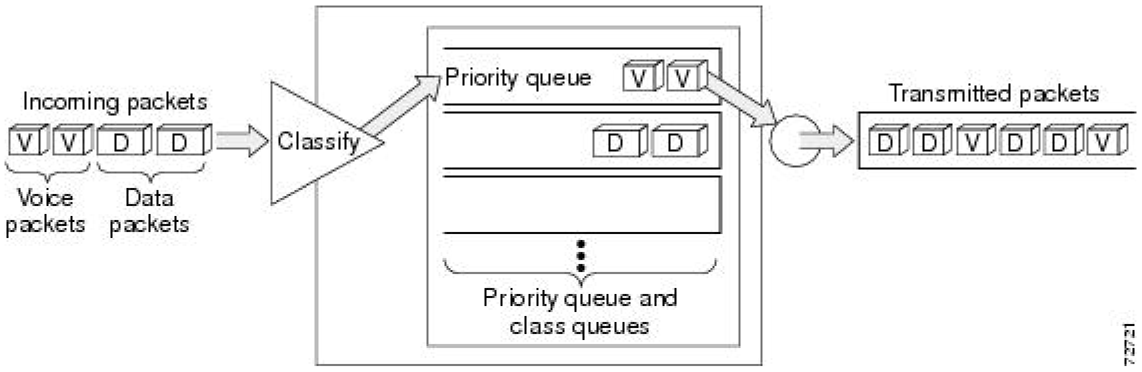
The Frame Relay Queueing and Fragmentation at the Interface feature simplifies the configuration of low-latency, low-jitter quality of service (QoS) by enabling the queueing policy and fragmentation configured on the main interface to apply to all permanent virtual circuits (PVCs) and subinterfaces under that interface. Before the introduction of this feature, queueing and fragmentation had to be configured on each individual PVC. Subrate shaping can also be configured on the interface.

How Frame Relay Queueing and Fragmentation at the Interface Works

When FRF.12 end-to-end fragmentation is enabled on an interface, all PVCs on the main interface and its subinterfaces will have fragmentation enabled with the same configured fragment size. To maintain low latency and low jitter for high-priority traffic, the configured fragment size must be greater than the largest high-priority frames. This configuration will prevent high-priority traffic from being fragmented and queued behind lower-priority fragmented frames. If the size of a high-priority frame is larger than the configured fragment size, the high-priority frame will be fragmented. Local Management Interface (LMI) traffic will not be fragmented and is guaranteed its required bandwidth.

When a low-latency queueing policy map is applied to the interface, traffic through the interface is identified using class maps and is directed to the appropriate queue. Time-sensitive traffic such as voice should be classified as high priority and will be queued on the priority queue. Traffic that does not fall into one of the defined classes will be queued on the class-default queue. Frames from the priority queue and class queues are subject to fragmentation and interleaving. As long as the configured fragment size is larger than the high-priority frames, the priority queue traffic will not be fragmented and will be interleaved with fragmented frames from other class queues. This approach provides the highest QoS transmission for priority queue traffic. The figure below illustrates the interface queueing and fragmentation process.

Figure 14: Frame Relay Queueing and Fragmentation at the Interface



Subrate shaping can also be applied to the interface, but interleaving of high-priority frames will not work when shaping is configured. If shaping is not configured, each PVC will be allowed to send bursts of traffic up to the physical line rate.

When shaping is configured and traffic exceeds the rate at which the shaper can send frames, the traffic is queued at the shaping layer using fair queueing. After a frame passes through the shaper, the frame is queued at the interface using whatever queueing method is configured. If shaping is not configured, then queueing occurs only at the interface.

Note For interleaving to work, both fragmentation and the low-latency queueing policy must be configured with shaping disabled.

The Frame Relay Queueing and Fragmentation at the Interface feature supports the following functionality:

- Voice over Frame Relay
- Weighted Random Early Detection
- Frame Relay payload compression

Note When payload compression and Frame Relay fragmentation are used at the same time, payload compression is always performed before fragmentation.

- IP header compression

Benefits of Frame Relay Queueing and Fragmentation at the Interface

Simple Configuration

The Frame Relay Queueing and Fragmentation at the Interface feature allows fragmentation, low-latency queueing, and subrate shaping to be configured on a Frame Relay interface queue. The fragmentation and queueing and shaping policy will apply to all PVCs and subinterfaces under the main interface, eliminating the need to configure QoS on each PVC individually.

Flexible Bandwidth

This feature allows PVCs to preserve the logical separation of traffic from different services while reducing bandwidth partitioning between PVCs. Each PVC can send bursts of traffic up to the interface shaping rate or, if shaping is not configured, the physical interface line rate.

How to Configure Frame Relay Queueing and Fragmentation at the Interface

Configuring Class Policy for the Priority Queue

To configure a policy map for the priority class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map*
4. **class** *class-name*
5. Router(config-pmap-c)# **priority** *bandwidth-kbps*
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map* **Example:** `Router(config) policy-map policy1`	Specifies the name of the policy map to be created or modified. • Use this command to define the queueing policy for the priority queue.

Step 4	**class** *class-name*
	
	Example:
	`Router(config-pmap)# class c1`
		Specifies the name of a class to be created and included in the service policy.
		• The class name that you specify in the policy map defines the characteristics for that class and its match criteria as configured using the **class-map** command.

Step 5	Router(config-pmap-c)# **priority** *bandwidth-kbps* **Example:** `Router(config-pmap-c)# priority 30`	Creates a strict priority class and specifies the amount of bandwidth, in kbps, to be assigned to the class.
Step 6	**exit** **Example:** `Router(config-pmap-c)# exit`	Exits the current configuration mode.

Configuring Class Policy for the Bandwidth Queues

To configure a policy map and create class policies that make up the service policy, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map*
4. **class** *class-name*
5. Router(config-pmap-c)# **bandwidth** *bandwidth-kbps*
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map* **Example:** `Router(config)# policy-map policy1`	Specifies the name of the policy map to be created or modified. • Use this command to define the queueing policy for the priority queue. • The bandwidth queues and the priority queue use the same policy map.

Step 4	**class** *class-name*
	
	Example:
	`Router(config-pmap)# class c1`
		Specifies the name of a class to be created and included in the service policy.
		• The class name that you specify in the policy map defines the characteristics for that class and its match criteria as configured using the **class-map**command.

Step 5	Router(config-pmap-c)# **bandwidth** *bandwidth-kbps* **Example:** `Router(config-pmap-c)# bandwidth 10`	Specifies the amount of bandwidth to be assigned to the class, in kbps, or as a percentage of the available bandwidth. Bandwidth must be specified in kbps or as a percentage consistently across classes. (Bandwidth of the priority queue must be specified in kbps.) • The sum of all bandwidth allocation on an interface cannot exceed 75 percent of the total available interface bandwidth. However, if you need to configure more than 75 percent of the interface bandwidth to classes, you can override the 75 percent maximum by using the **max-reserved-bandwidth** command.
Step 6	**exit** **Example:** `Router(config-pmap-c)# exit`	Exits the current configuration mode.

Configuring the Shaping Policy Using the Class-Default Class

In general, the class-default class is used to classify traffic that does not fall into one of the defined classes. Even though the class-default class is predefined when you create the policy map, you still have to configure it. If a default class is not configured, traffic that does not match any of the configured classes is given best-effort treatment, which means that the network will deliver the traffic if it can, without any assurance of reliability, delay prevention, or throughput.

If you configure shaping in addition to queueing on the interface, use the class-default class to configure the shaping policy. The shaping policy will serve as the parent in a hierarchical traffic policy. The queueing policy will serve as the child policy. The class-default class is used for the shaping policy so that all traffic for the entire interface is shaped and a bandwidth-limited stream can be created.

To configure the shaping policy in the class-default class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map*
4. **class class-default**
5. **shape** [**average** | **peak**] *mean-rate* [[*burst-size*] [*excess-burst-size*]]
6. **service-policy** *policy-map-name*
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map* **Example:** `Router(config)# policy-map policy1`	Specifies the name of the policy map to be created or modified. • Use this command to define the shaping policy.

Step 4	**class class-default**
	
	Example:
	`Router(config-pmap)# class class-default`
		Specifies the default class so that you can configure or modify its policy.

Step 5	**shape** [**average** | **peak**] *mean-rate* [[*burst-size*] [*excess-burst-size*]] **Example:** `Router(config-pmap-c)# shape peak 10`	(Optional) Shapes traffic to the indicated bit rate according to the algorithm specified.

	Command or Action	Purpose
Step 6	**service-policy** *policy-map-name* **Example:** `Router(config-pmap-c)# service-policy policy1`	Specifies the name of a policy map to be used as a matching criterion (for nesting traffic policies [hierarchical traffic policies] within one another). • Use this command to attach the policy map for the priority queue (the child policy) to the shaping policy (the parent policy).
Step 7	**exit** **Example:** `Router(config-pmap-c)# exit`	Exits the current configuration mode.

Configuring Queueing and Fragmentation on the Frame Relay Interface

To configure low-latency queueing and FRF.12 end-to-end fragmentation on a Frame Relay interface, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type* / *number*
4. **encapsulation frame-relay**
5. **service-policy output** *policy-map-name*
6. **frame-relay fragment** *fragment-size* **end-to-end**
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *type* / *number* **Example:** `Router(config)# interface fe 0/0`	Configures an interface type and enters interface configuration mode.
Step 4	**encapsulation frame-relay** **Example:** `Router(config-if)#` **encapsulation frame-relay**	Enables Frame Relay encapsulation.
Step 5	**service-policy output** *policy-map-name* **Example:** `Router(config-if)#` **service-policy output** `policy1`	Attaches a policy map to an output interface, to be used as the service policy for that interface. • If shaping is being used, use this command to attach the shaping policy (which includes the nested queueing policy) to the interface. • Interleaving of high-priority frames will not work if shaping is configured on the interface. • If shaping is not being used, use this command to attach the queueing policy to the interface.
Step 6	**frame-relay fragment** *fragment-size* **end-to-end** **Example:** `Router(config-if)# frame-relay fragment 100 end-to-end`	Enables fragmentation of Frame Relay frames. • To maintain low latency and low jitter for priority queue traffic, configure the fragment size to be greater than the largest high-priority frame that would be expected.

Step 7	**exit**
	
	Example:
	
	`Router(config-if)#`
	
	`exit`
		Exits the current configuration mode.

Verifying Frame Relay Queueing and Fragmentation at the Interface

To verify the configuration and performance of Frame Relay queueing and fragmentation at the interface, perform the following steps:

SUMMARY STEPS

1. Enter the **show running-config** command to verify the configuration.
2. Enter the **show policy-map interface** command to display low-latency queueing information, packet counters, and statistics for the policy map applied to the interface. Compare the values in the "packets" and the "pkts matched" counters; under normal circumstances, the "packets" counter is much larger than the "pkts matched" counter. If the values of the two counters are nearly equal, then the interface is receiving a large number of process-switched packets or is heavily congested.
3. Enter the **show interfaces serial**command to display information about the queueing strategy, priority queue interleaving, and type of fragmentation configured on the interface. You can determine whether the interface has reached a congestion condition and packets have been queued by looking at the "Conversations" fields. A nonzero value for "max active" counter shows whether any queues have been active. If the "active" counter is a nonzero value, you can use the **show queue** command to view the contents of the queues.

DETAILED STEPS

Step 1 Enter the **show running-config** command to verify the configuration.

Example:

```
Router# show running-config
Building configuration...
.
.
.
class-map match-all voice
  match ip precedence 5
!
!policy-map llq
  class voice
    priority 64
policy-map shaper
  class class-default
   shape peak 96000
   service-policy llq
!
!interface Serial1/1
 ip address 16.0.0.1 255.255.255.0
 encapsulation frame-relay
 service-policy output shaper
 frame-relay fragment 80 end-to-end
!
```

Step 2 Enter the **show policy-map interface** command to display low-latency queueing information, packet counters, and statistics for the policy map applied to the interface. Compare the values in the "packets" and the "pkts matched" counters; under normal circumstances, the "packets" counter is much larger than the "pkts matched" counter. If the values of the two counters are nearly equal, then the interface is receiving a large number of process-switched packets or is heavily congested.
The following sample output for the **show policy-map interface command** is based on the configuration in Step 1:

Example:

```
Router# show policy-map interface serial 1/1
 Serial1/1
  Service-policy output:shaper
```

```
    Class-map:class-default (match-any)
      12617 packets, 1321846 bytes
      5 minute offered rate 33000 bps, drop rate 0 bps
      Match:any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           192000/96000     1992   7968      7968      83        1992
        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         12586     1321540   0         0         no
      Service-policy :llq
        Class-map:voice (match-all)
          3146 packets, 283140 bytes
          5 minute offered rate 7000 bps, drop rate 0 bps
          Match:ip precedence 1
          Weighted Fair Queueing
            Strict Priority
            Output Queue:Conversation 24
            Bandwidth 64 (kbps) Burst 1600 (Bytes)
            (pkts matched/bytes matched) 0/0
            (total drops/bytes drops) 0/0
        Class-map:class-default (match-any)
          9471 packets, 1038706 bytes
          5 minute offered rate 26000 bps
          Match:any
```

Step 3 Enter the **show interfaces serial**command to display information about the queueing strategy, priority queue interleaving, and type of fragmentation configured on the interface. You can determine whether the interface has reached a congestion condition and packets have been queued by looking at the "Conversations" fields. A nonzero value for "max active" counter shows whether any queues have been active. If the "active" counter is a nonzero value, you can use the **show queue** command to view the contents of the queues.

The following sample output for the **show interfaces serial**command is based on the configuration in Step 1:

Example:

```
Router# show interfaces serial 1/1
Serial1/1 is up, line protocol is up
  Hardware is M4T
  Internet address is 16.0.0.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 5/255, rxload 1/255
  Encapsulation FRAME-RELAY, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
  LMI enq sent  40, LMI stat recvd 40, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  Fragmentation type:end-to-end, size 80, PQ interleaves 0
  Broadcast queue 0/64, broadcasts sent/dropped 0/0, interface broadcasts 0
  Last input 00:00:03, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:06:34
  Input queue:0/75/0/0 (size/max/drops/flushes); Total output drops:0
  Queueing strategy:weighted fair
  Output queue:0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/1/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 33000 bits/sec, 40 packets/sec
     40 packets input, 576 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     15929 packets output, 1668870 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
```

```
0 output buffer failures, 0 output buffers swapped out
0 carrier transitions     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Monitoring and Maintaining Frame Relay Queueing and Fragmentation at the Interface

To monitor and maintain Frame Relay queueing and fragmentation at the interface, use the following commands in privileged EXEC mode:

SUMMARY STEPS

1. **debug frame-relay fragment** [**event** | **interface** *type* / *number dlci*]
2. **show frame-relay fragment** [**interface** *type* / *number* [*dlci*]]
3. **show interfaces serial** *number*
4. **show queue** *interface-type interface-number*
5. **show policy-map interface** *number* [**input** | **output**]

DETAILED STEPS

Step 1 **debug frame-relay fragment** [**event** | **interface** *type* / *number dlci*]
Displays information related to Frame Relay fragmentation on a PVC.

Step 2 **show frame-relay fragment** [**interface** *type* / *number* [*dlci*]]
Displays information about Frame Relay fragmentation.

Step 3 **show interfaces serial** *number*
Displays information about a serial interface.

Step 4 **show queue** *interface-type interface-number*
Displays the contents of packets inside a queue for a particular interface.

Step 5 **show policy-map interface** *number* [**input** | **output**]
Displays the packet statistics of all classes that are configured for all service policies on the specified interface.

Configuration Examples for Frame Relay Queueing and Fragmentation at the Interface

Example Frame Relay Queueing Shaping and Fragmentation at the Interface

The following example shows the configuration of a hierarchical policy for low-latency queueing, FRF.12 fragmentation, and shaping on serial interface 3/2. Note that traffic from the priority queue will not be interleaved with fragments from the class-default queue because shaping is configured.

```
class-map voice
 match access-group 101

policy-map llq
 class voice
  priority 64

policy-map shaper
 class class-default
  shape average 96000
  service-policy llq
interface serial 3/2
 ip address 10.0.0.1 255.0.0.0
 encapsulation frame-relay
 bandwidth 128
 clock rate 128000
 service-policy output shaper
 frame-relay fragment 80 end-to-end

 access-list 101 match ip any host 10.0.0.2
```

Example Frame Relay Queueing and Fragmentation at the Interface

The following example shows the configuration of low-latency queueing and FRF.12 fragmentation on serial interface 3/2. Because shaping is not being used, a hierarchical traffic policy is not needed and traffic from the priority queue will be interleaved with fragments from the other queues. Without shaping, the output rate of the interface is equal to the line rate or configured clock rate. In this example, the clock rate is 128,000 bps.

```
class-map voice
 match access-group 101

policy-map llq
 class voice
  priority 64
 class video
  bandwidth 32
interface serial 3/2
 ip address 10.0.0.1 255.0.0.0
 encapsulation frame-relay
 bandwidth 128
 clock rate 128000
 service-policy output llq
 frame-relay fragment 80 end-to-end
 access-list 101 match ip any host 10.0.0.2
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
WAN commands	*Cisco IOS Wide-Area Networking Command Reference*
Configuring Frame Relay	Configuring Frame Relay

Standards

Standard	Title
FRF.12	*Frame Relay Fragmentation Implementation Agreement* , December 1997

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Frame Relay Queueing and Fragmentation at the Interface

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 2: Feature Information for Frame Relay Queueing and Fragmentation at the Interface

Feature Name	Releases	Feature Information
Frame Relay Queuing and Fragmentation at the Interface	12.2(11)S 12.2(13)T 15.0(1)S	The Frame Relay Queueing and Fragmentation at the Interface feature introduces support for low-latency queueing (LLQ) and FRF.12 end-to-end fragmentation on a Frame Relay interface. The following commands were introduced or modified: **frame-relay fragment end-to-end**, **show interfaces serial**.

CHAPTER 5

Frame Relay PVC Bundles with QoS Support for IP and MPLS

Frame Relay permanent virtual circuit (PVC) bundle functionality allows you to associate a group of Frame Relay PVCs with a single next-hop address. When Frame Relay PVC bundles are used with IP, packets are mapped to specific PVCs in the bundle on the basis of the precedence value or differentiated services code point (DSCP) settings in the type of service (ToS) field of the IP header. Each packet is treated differently according to the QoS configured for each PVC.

MPLS QoS support for Frame Relay PVC bundles extends Frame Relay PVC bundle functionality to support the mapping of Multiprotocol Label Switching (MPLS) packets to specific PVCs in the bundle. MPLS packets are mapped to PVCs according to the settings of the experimental (EXP) bits in the MPLS packet header.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Frame Relay PVC Bundles with QoS Support for IP and MPLS

To implement Frame Relay PVC bundles between two routers, you must enable IP Cisco Express Forwarding switching on the routers.

To configure MPLS EXP levels on bundle member PVCs, you must have tag-switching enabled on the interface.

It is recommended (but not required) that you implement PVC Interface Priority Queueing (PIPQ) in conjunction with Frame Relay PVC bundles. This will ensure that if the interface becomes congested, higher-priority traffic can exit the interface ahead of lower-priority traffic.

Restrictions for Frame Relay PVC Bundles with QoS Support for IP and MPLS

- A PVC can be a part of one and only one PVC bundle.
- A PVC bundle may contain no more than eight PVCs.
- A PVC that is a bundle member cannot be used in any other capacity, For example a PVC bundle member cannot be configured in a map statement.
- A PVC bundle cannot perform precedence and DSCP matching at the same time. If the wrong matching scheme is configured, unpredictable behavior will result.
- A PVC bundle will not come up unless all the precedence, DSCP, or EXP levels are configured in the bundle.
- Voice over Frame Relay (VoFR) is not supported on PVC-bundle members.
- Fast switching over Frame Relay PVC bundles is not supported.

Information About Frame Relay PVC Bundles with QoS Support for IP and MPLS

Benefits of Frame Relay PVC Bundles with QoS Support for IP and MPLS

- IP or MPLS packets carrying different types of traffic can be transported on different PVCs within the same PVC bundle.
- Precedence-based PVC bundles can be converted to EXP-based PVC bundles by enabling tag-switching. EXP-based PVC bundles can be converted to precedence-based PVC bundles by disabling tag-switching.

- This feature provides flexible PVC management within a PVC bundle by allowing traffic assigned to a failed PVC to be redirected to an alternate PVC within the bundle. This feature also allows you to configure the bundle to go down when certain PVCs go down.

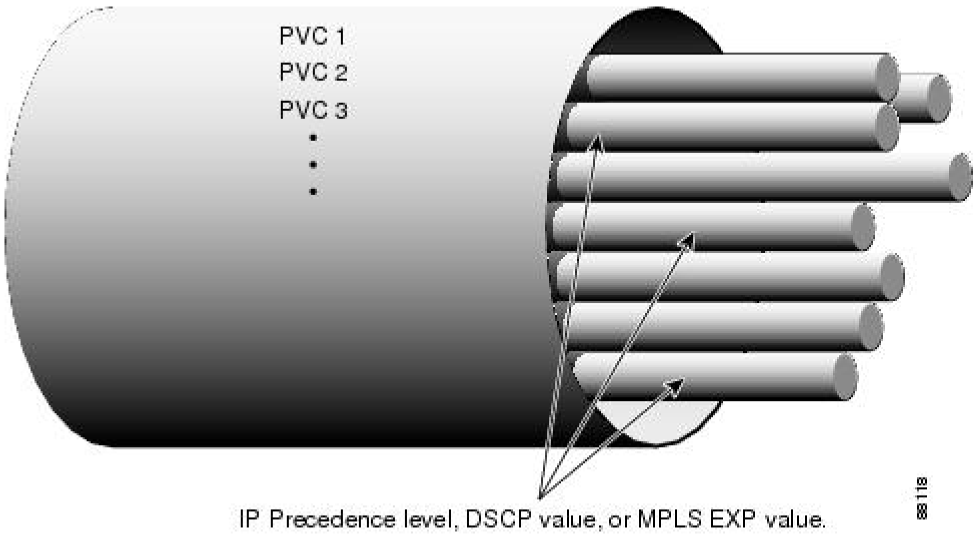
Frame Relay PVC Bundle Support

The use of Frame Relay PVC bundles allows you to configure multiple PVCs with different QoS characteristics between any pair of Frame Relay-connected routers. As shown in the figure below, a PVC bundle may contain up to eight PVCs. The individual PVCs within a bundle are called *bundle members* .

To determine which PVC in a bundle will be used to forward a specific type of traffic, the router maps the IP precedence level or DSCP value in an IPv4 packet header to a PVC configured with the same value. In the case of MPLS, packets are mapped to specific PVCs in a bundle based on the settings of the EXP bits in the MPLS packet headers.

Once you define a Frame Relay bundle and add PVCs to it, you can configure attributes and characteristics to discrete PVC bundle members, or you can apply them collectively at the bundle level. Frame Relay traffic shaping may be applied to every PVC within a bundle. As with individual PVCs, you can enable rate adaptation to occur in response to incoming backward explicit congestion notifications (BECN) from the network.

Figure 15: Frame Relay PVC bundle



You can create differentiated service using PVC bundles by distributing IP precedence levels or DSCP values over the various bundle members. You can map either a single precedence level or a range of precedence levels to each PVC in the bundle. Thus, either you can limit an individual PVC to carry only packets marked with a specific precedence level or you can enable a PVC to carry packets marked with different precedence levels.

Service Levels and PVC Selection Criteria

The DSCP and Precedence bits classify IP packet service levels. The Precedence field consists of the first three bits of the ToS octet in the IPv4 header. These bits define eight precedence levels. When DSCP mapping is used, the DSCP octet replaces the ToS octet in the IPv4 header. Currently the first six bits are used, defining 64 service levels.

Using precedence-based or DSCP-based mapping, each IPv4 packet is mapped to a specific PVC in the bundle, according to the value of the ToS or DSCP octet in the IP header. There is no special treatment for broadcast or multicast or IP routing packets; the only differentiation in treatment is a result of the ToS or DSCP octet settings.

The MPLS EXP bits make up a three-bit experimental field in the MPLS packet header. They are a bit-by-bit copy of the IP Precedence bits and provide the same eight QoS levels. Under MPLS EXP-based mapping, each MPLS packet is mapped to a specific PVC in the bundle, according the setting of the EXP bits.

Frame Relay PVC Bundle Management

In addition to mapping specific traffic types to specific PVCs according to QoS parameters designated by the ToS or DSCP values in the IPv4 headers or EXP values in the MPLS headers, PVC bundle management takes care of handling non-IP traffic and determining what happens if a PVC goes down.

By default, Inverse Address Resolution Protocol (ARP) traffic and other critical non-IP traffic is carried by the PVC configured for carrying IP Precedence or EXP level 6 or DSCP level 63. You can select a PVC with a different QoS to carry Inverse ARP traffic if required. Noncritical non-IP traffic is carried by the PVC that configured for carrying IP precedence, EXP, or DSCP level 0.

It is important during configuration to account for every precedence, EXP, or DSCP level in the configuration of the PVC bundle members. If all the packet service levels are not accounted for, the PVC bundle will never come up.

Once a PVC bundle is up, if an individual bundle member goes down, an attempt is made to identify an alternate PVC to handle the packet service level or levels that were carried by the downed PVC. If no alternate PVC is found, the entire PVC bundle is brought down.

Traffic Bumping

You can configure each PVC bundle member to bump traffic to another PVC in the bundle in the event that the bundle member goes down. You can specify whether the bumping will be implicit or explicit bumping. You can also specify that a particular PVC will never accept bumped traffic from another PVC. The default conditions are to perform implicit traffic bumping and to accept bumped traffic.

Implicit bumping diverts the traffic from a failed PVC to the PVC having the next-lower service level. Explicit bumping forces the traffic to a specific PVC rather than allowing it to find a PVC carrying traffic of the next-lower service level. For example, PVC *x* , responsible for carrying precedence level 3 traffic, can be configured to bump its traffic to PVC *y* , responsible for carrying precedence level 6 traffic--provided that PVC *y* is configured to accept bumped traffic. If PVC *x* goes down, PVC *y* takes over. If PVC *y* is already down or goes down later, the alternate PVC selected will depend on the bumping rule for PVC *y* . If no alternate PVC can be found for bumped traffic, the entire PVC bundle goes down.

PVC-Bundle Protection Rules

Traffic bumping provides a way to keep a PVC bundle up and traffic flowing even though some individual PVCs may be down. Protection rules provide a way to force the PVC bundle down even though some individual PVCs are up and might be able to handle all the traffic, though perhaps not in a satisfactory manner.

You can configure a PVC bundle member as an individually protected PVC or as part of a PVC bundle protected group. Only one protected group may exist within a PVC bundle; however, many individually protected PVCs may exist. The protection rules add flexibility for controlling the PVC bundle state.

When any one individually protected PVC goes down, the entire bundle goes down. If all the PVCs in a protected group go down, the entire bundle goes down.

If no protection rule is specified, the PVC bundle goes down only when all the PVCs go down. However, protection is overridden if a PVC that has no place to bump its traffic goes down. In this case, the entire bundle will go down despite any protection rules that have been set up.

MPLS EXP-based Mapping

To enable MPLS EXP-based mapping, tag-switching must be enabled on the interface or subinterface by using the **tag-switching ip** command. When tag-switching is enabled, MPLS and IP packets can flow across the interface and PVC bundles that are configured for IP Precedence mapping are converted to MPLS EXP mapping. The PVC bundle functionality remains the same with respect to priority levels, bumping, and so on, but the **match precedence** command is replaced by **match exp**, and each **precedence** command is replaced by the **exp** command. The effect is that a bundle member PVC previously configured to carry precedence level 1 IP traffic now carries EXP level 1 MPLS traffic.

PVC bundles configured for DSCP mapping go down when tag-switching is enabled. The DSCP configuration for each bundle member PVC is reset, resulting in the PVCs being unmapped and Inverse ARP, bumping, and protection settings being unconfigured. The **match dscp** command is replaced by **match exp**command.

When tag-switching is disabled, the **match precedence**and **match dscp**commands are restored and the **exp** commands are replaced by **precedence** commands.

When tag-switching is enabled or disabled, PVC bundles configured for IP precedence mapping or MPLS EXP mapping will stay up and traffic will transmit over the appropriate bundle member PVCs.

How to Configure Frame Relay PVC Bundles with QoS Support for IP and MPLS

Configuring Frame Relay PVC Bundles with IP QoS Support

To configure Frame Relay PVC bundles for handling IP packets, perform the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip routing**
4. **ip cef**
5. Do one of the following:
    - **interface** *type number*
    -
    -
    -
    - **interface** *type number* **.** *subinterface-number*
    - **multipoint** | **point-to-point**]
6. **encapsulation frame-relay** [**cisco** |**ietf**]
7. **ip address** *ip-address* *mask* [**secondary**]
8. **frame-relay map** *protocol* *protocol-address* {*dlci*| **vc-bundle** *vc-bundle-name*} [**broadcast**] [**ietf**| **cisco**]
9. **frame-relay vc-bundle** *vc-bundle-name*
10. **encapsulation** [**cisco** | **ietf**]
11. **match** {**dscp** *dscp-value*| **precedence** *precedence-value*}
12. **pvc** *dlci* [*vc-name*]
13. **class** *name*
14. Do one of the following:
    - **precedence** {*level* | **other**}
    -
    -
    -
    - **dscp** {*level* | **other**}
15. **bump** {**explicit** *level* | **implicit** | **traffic**}
16. **protect** {**group** | **vc**}
17. **inarp**
18. **end**
19. Configure the PVC bundle on the peer router.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Router> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip routing** **Example:** `Router(config)# ip routing`	Enables IP routing.
Step 4	**ip cef** **Example:** `Router(config)# ip cef`	Enables Cisco Express Forwarding. **Note** For the Cisco 7500, enter **ip cef distributed**.

Step 5	Do one of the following:
	• **interface** *type number*
	•
	•
	•
	• **interface** *type number* **.** *subinterface-number*
	• **multipoint** | **point-to-point**]
	Example:
	`Router(config)# interface serial 0`
	Example:
	Example:
	Example:
	`Router(config)# interface serial 1.1 multipoint`
		Specifies the interface type and number and enters interface configuration mode.
		• Physical interfaces are multipoint subinterfaces by default.
		or
		Specifies the interface type and subinterface and enters subinterface configuration mode.
		• Once you create a specific type of subinterface (point-to-point or multipoint), you cannot change it without a reload. To change it, you must either reload the router or create another subinterface.

	Command or Action	Purpose
Step 6	**encapsulation frame-relay** [**cisco** |**ietf**] **Example:** `Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation. • The default encapsulation method is **cisco**.
Step 7	**ip address** *ip-address* *mask* [**secondary**] **Example:** `Router(config-if)# ip address 10.1.1.1 255.0.0.0`	Sets a primary IP address for the interface. • The optional **secondary** keyword specifies that the configured address is a secondary IP address. If this keyword is omitted, the configured address is the primary IP address.
Step 8	**frame-relay map** *protocol* *protocol-address* {*dlci*| **vc-bundle** *vc-bundle-name*} [**broadcast**] [**ietf**| **cisco**] **Example:** `Router(config-if)# frame-relay map ip 10.2.2.2 vc-bundle MAIN-1`	(Optional) Maps between a next-hop protocol address and a data-link connection identifier (DLCI) destination address, and creates a PVC bundle if it does not already exist. • The protocol-address is the destination IP address. • The **frame-relay map** command is required for multipoint interfaces if Inverse ARP has been disabled or is not supported at the other end of the connection.
Step 9	**frame-relay** **vc-bundle** *vc-bundle-name* **Example:** `Router(config-if)# frame-relay vc-bundle MAIN-1`	Creates a PVC bundle if it does not already exist, and enters Frame Relay VC-bundle configuration mode.
Step 10	**encapsulation** [**cisco** | **ietf**] **Example:** `Router(config-fr-vcb)# encapsulation ietf`	(Optional) Overrides the encapsulation type configured on the interface and configures the Frame Relay encapsulation type for the PVC bundle. • This command is available only when the PVC bundle is configured on a point-to-point subinterface.

Step 11	**match** {**dscp** *dscp-value*| **precedence** *precedence-value*}
	
	Example:
	
	`Router(config-fr-vcb)# match precedence 5`
		Establishes the type of matching to use between incoming packet headers and PVC-bundle members.
		
		• The default match type is **precedence**.

Step 12	**pvc** *dlci* [*vc-name*] **Example:** `Router(config-fr-vcb)# pvc 100 1a`	Creates a PVC-bundle member and enters Frame Relay VC-bundle-member configuration mode. • The *vc-name* argument is an optional name that can be used for referring to the PVC.

	Command or Action	Purpose

Step 13	**class** *name*
	
	Example:
	
	`Router(config-fr-vcb-vc)# class premium`
		(Optional) Assigns a map class to the PVC-bundle member defined in the previous step.

Step 14	Do one of the following:
	• **precedence** {*level* | **other**}
	•
	•
	•
	• **dscp** {*level* | **other**}
	
	Example:
	
	`Router(config-fr-vcb-vc)# precedence 6-7`
	
	Example:
	
	Example:
	
	Example:
	
	`Router(config-fr-vcb-vc)# dscp other`
		(Optional) Enters the mapped service level or range for the PVC-bundle member.
		• The **precedence** command is available when the PVC-bundle match type is set to **precedence**.
		• The precedence range is from 0 to 7.
		• The **dscp** command is available when the PVC-bundle match type is set to **dscp**.
		• The **dscp** range is from 0 to 63.
		• The **other** keyword is used to designate a PVC to handle all the remaining levels that have not been assigned to other PVCs in the bundle.
		• Critical non-IP traffic will automatically use precedence level 0.

Step 15	**bump** {**explicit** *level* | **implicit** | **traffic**}
	
	Example:
	
	`Router(config-fr-vcb-vc)# bump explicit 7`
		(Optional) Specifies the bumping rule for the PVC-bundle member.
		• The default bumping rule is implicit bumping.
		• Use the **explicit** *level* option to specify the service level to which traffic on this PVC will be bumped if the PVC goes down. In that event, the traffic will be directed to a PVC mapped with the service level configured here. If the PVC that picks up and carries the traffic also goes down, the traffic is subject to the bumping rules for that PVC. You can specify only one service level for bumping.
		• The PVC-bundle member accepts bumped traffic by default when the PVC-bundle match type is **precedence**. To configure the PVC to reject bumped traffic from another PVC-bundle member, use the **no bump traffic** command.

Step 16	**protect** {**group** | **vc**}	(Optional) Specifies the protection rule for the PVC-bundle member.

	Command or Action	Purpose
	Example: `Router(config-fr-vcb-vc)# protect group`	• By default, the PVC-bundle member is not protected. • If you use the **vc** keyword, the PVC bundle goes down whenever this PVC goes down. • If you use the **group** keyword, the PVC bundle goes down when the last PVC in the protected group goes down.
Step 17	**inarp** **Example:** `Router(config-fr-vcb-vc)# inarp`	(Optional) Enables Inverse ARP for the PVC-bundle member. • By default, Inverse ARP traffic uses the PVC configured for precedence level 6 or DSCP level 63.

Step 18	**end**
	Example:
	`Router(config-fr-vcb-vc)# end`
		Exits to privileged EXEC mode.

Step 19	Configure the PVC bundle on the peer router.	(Optional) While it is not necessary to configure a PVC bundle on the peer router, it is recommended that you do so for applications that rely on communications on the same PVC (such as TCP header-compression.)

Configuring Frame Relay PVC Bundles with MPLS QoS Support

To configure Frame Relay PVC bundles for handling MPLS packets, perform the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip routing**
4. **ip cef**
5. Do one of the following:
   - **interface** *type number*
   -
   -
   -
   - **interface** {*type slot* | *port-adapter* | *port.subinterface-number*} [**multipoint** | **point-to-point**]
6. **encapsulation frame-relay** [**cisco** | **ietf**]
7. **tag-switching ip**
8. **ip address** *ip-address mask* [**secondary**]
9. **frame-relay map** *protocol protocol-address* {*dlci*| **vc-bundle** *vc-bundle-name*} [**broadcast**] [**ietf**| **cisco**]
10. **frame-relay vc-bundle** *vc-bundle-name*
11. **encapsulation** [**ietf** | **cisco**]
12. **pvc** *dlci* [*vc-name*]
13. **class** *name*
14. **exp** {*level* | **other**}
15. **bump** {**explicit** *level* | **implicit** | **traffic**}
16. **protect** {**group** | **vc**}
17. **inarp**
18. **end**
19. Configure the PVC bundle on the peer router.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**ip routing** **Example:** `Router(config)# ip routing`	Enables IP routing.
Step 4	**ip cef** **Example:** `Router(config)# ip cef`	Enables Cisco Express Forwarding. **Note** For the Cisco 7500, enter **ip cef distributed**.

Step 5	Do one of the following:
	• **interface** *type number*
	•
	•
	•
	• **interface** {*type slot* | *port-adapter* | *port.subinterface-number*} [**multipoint** | **point-to-point**]
	
	Example:
	
	`Router(config)# interface serial 0`
	
	Example:
	
	Example:
	
	Example:
	
	Example:
	
	`Router(config)# interface serial 1.1 multipoint`
		Specifies the interface type and number and enters interface configuration mode.
		• Physical interfaces are multipoint subinterfaces by default.
		
		or
		
		Specifies the interface type and subinterface and enters subinterface configuration mode.
		• Once you create a specific type of subinterface (point-to-point or multipoint), you cannot change it without a reload. To change it, you need to either reload the router or create another subinterface.

Step 6	**encapsulation frame-relay** [**cisco** | **ietf**] **Example:** `Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation. • The default encapsulation method is **cisco**.

	Command or Action	Purpose
Step 7	**tag-switching ip** **Example:** `Router(config-if)# tag-switching ip`	Enables label switching of IPv4 packets on an interface.
Step 8	**ip address** *ip-address* *mask* [**secondary**] **Example:** `Router(config-if)# ip address 10.1.1.1 255.0.0.0`	Sets a primary IP address for the interface. • The optional **secondary** keyword specifies that the configured address is a secondary IP address. If this keyword is omitted, the configured address is the primary IP address.
Step 9	**frame-relay map** *protocol* *protocol-address* {*dlci*| **vc-bundle** *vc-bundle-name*} [**broadcast**] [**ietf**| **cisco**] **Example:** `Router(config-if)# frame-relay map ip 10.2.2.2 vc-bundle MAIN-1`	(Optional) Maps between a next-hop protocol address and a DLCI destination address, and creates a PVC bundle if it does not already exist. • The protocol-address is the destination IP address. • The **frame-relay map** command is required for multipoint interfaces if Inverse ARP has been disabled or is not supported at the other end of the connection.
Step 10	**frame-relay vc-bundle** *vc-bundle-name* **Example:** `Router(config-if)# frame-relay vc-bundle MAIN-1`	Creates a PVC bundle if it does not already exist, and enters Frame Relay VC-bundle configuration mode.
Step 11	**encapsulation** [**ietf** | **cisco**] **Example:** `Router(config-fr-vcb)# encapsulation ietf`	(Optional) Overrides the encapsulation type configured on the interface and configures the Frame Relay encapsulation type for the PVC bundle. • This command is available only when the PVC bundle is configured on a point-to-point subinterface.
Step 12	**pvc** *dlci* [*vc-name*] **Example:** `Router(config-fr-vcb)# pvc 100 1a`	Creates a PVC-bundle member and enters Frame Relay VC-bundle-member configuration mode. • The *vc-name* argument is an optional name that can be used for referring to the PVC.

Step 13	**class** *name*
	
	Example:
	
	`Router(config-fr-vcb-vc)# class premium`
		(Optional) Assigns a map class to the PVC-bundle member.

Step 14	**exp** {*level* | **other**}
	
	Example:
	
	`Router(config-fr-vcb-vc)# exp 6-7`
		(Optional) Enters the mapped EXP level or range for the PVC-bundle member.
		• The **exp** command is available only when tag-switching has been enabled.

	Command or Action	Purpose
		• The EXP level values are from 0 to 7. • The **other** keyword is used to designate a PVC to handle all the remaining levels that have not been assigned to other PVCs in the bundle.

Step 15	**bump** {**explicit** *level* | **implicit** | **traffic**}
	
	Example:
	
	`Router(config-fr-vcb-vc)# bump explicit 7`
		(Optional) Specifies the bumping rule for the PVC-bundle member defined above.
		• The default bumping rule is implicit bumping.
		• Use the **explicit** *level* option to specify the EXP level to which traffic on this PVC will be bumped if the PVC goes down. In that event, the traffic will be directed to a PVC mapped with the EXP level configured here. If the PVC that picks up and carries the traffic also goes down, the traffic is subject to the bumping rules for that PVC. You can specify only one EXP level for bumping.
		• To configure the PVC to reject bumped traffic from another PVC-bundle member, use the **no bump traffic** command.

Step 16	**protect** {**group** | **vc**}
	
	Example:
	
	`Router(config-fr-vcb-vc)# protect group`
		(Optional) Specifies the protection rule for the PVC-bundle member defined above.
		• By default, the PVC-bundle member is not protected.
		• If you use the **vc** keyword, the PVC bundle goes down whenever this PVC goes down.
		• If you use the **group** keyword, the PVC bundle goes down when the last PVC in the protected group goes down.

Step 17	**inarp** **Example:** `Router(config-fr-vcb-vc)# inarp`	(Optional) Enables Inverse ARP for the PVC-bundle member defined above. • By default, Inverse ARP traffic uses the PVC configured for EXP level 6.

Step 18	**end**
	
	Example:
	
	`Router(config-fr-vcb-vc)# end`
		(Optional) Exits to privileged EXEC mode.

Step 19	Configure the PVC bundle on the peer router.	(Optional) While it is not necessary to configure a PVC bundle on the peer router, it is recommended that you do so for applications that rely on communications on the same PVC (such as TCP header-compression.)

Verifying Frame Relay PVC Bundles Configuration

To verify the configuration and operation of Frame Relay PVC bundles with QoS support, perform the following optional steps:

SUMMARY STEPS

1. **enable**
2. **show frame-relay vc-bundle** *vc-bundle-name* [**detail**
3. **show frame-relay map**
4. **show frame-relay pvc**
5. **show frame-relay ip rtp header-compression** [**interface** *type number*]
6. **show frame-relay ip tcp header-compression** [**interface** *type number*]
7. **show adjacency** [*type number*] [**detail**] [**summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show frame-relay vc-bundle** *vc-bundle-name* [**detail** **Example:** `Router# show frame-relay vc-bundle mp-3-static`	Displays status, bumping information, protection information, and active and configured precedence or DSCP levels for the PVCs in a PVC bundle.
Step 3	**show frame-relay map** **Example:** `Router# show frame-relay map`	Displays the current Frame Relay map entries and information about the connections.
Step 4	**show frame-relay pvc** **Example:** `Router# show frame-relay pvc`	Displays PVC statistics for the PVC-bundle members.

	Command or Action	Purpose
Step 5	**show frame-relay ip rtp header-compression** [**interface** *type number*] **Example:** `Router# show frame-relay ip rtp header-compression`	Displays Frame Relay Real-Time Transport Protocol (RTP) header compression statistics for PVC bundles.
Step 6	**show frame-relay ip tcp header-compression** [**interface** *type number*] **Example:** `Router# show frame-relay ip tcp header-compression serial 1/4`	Displays Frame Relay TCP/IP header compression statistics for PVC bundles.
Step 7	**show adjacency** [*type number*] [**detail**] [**summary**] **Example:** `Router# show adjacency`	Displays Cisco Express Forwarding adjacency table information.

Monitoring and Maintaining Frame Relay PVC Bundles

To monitor and maintain Frame Relay PVC bundles, perform this task.

SUMMARY STEPS

1. **enable**
2. **debug frame-relay adjacency** {**pvc**[*dlci*] | **vc-bundle** [*vc-bundle-name*]}
3. **debug frame-relay vc-bundle** {**detail** | **state-change**} [*vc-bundle-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug frame-relay adjacency** {**pvc**[*dlci*] | **vc-bundle** [*vc-bundle-name*]}	Displays information pertaining to an adjacent node that has one or more Frame Relay PVCs or PVC bundles.

	Command or Action	Purpose
	Example: `Router# debug frame-relay adjacency pvc`	• Use this command to monitor adjacency activity.
Step 3	**debug frame-relay vc-bundle** {**detail** | **state-change**} [*vc-bundle-name*] **Example:** `Router# debug frame-relay vc-bundle state-change`	Displays information about the Frame Relay PVC bundles configured on a router. • Use this command to monitor state changes and Inverse ARP activity for one or all of the PVC bundles and bundle members configured on a router. **Note** Using the **detail** keyword generates a large number of debugs that can quickly fill up a log buffer.

Configuration Examples for Frame Relay PVC Bundles with QoS Support for IP and MPLS

PVC Bundles with IP QoS Support on Interfaces Example

The following example shows the configuration of five PVC bundles with IP precedence and DSCP mapping. Two bundles are configured on the main interface, one bundle with static mapping and one with dynamic mapping. Two bundles are configured on a multipoint subinterface, one bundle with static mapping and one with dynamic mapping. One bundle is configured on a point-to-point subinterface.

```
configure terminal
ip routing
ip cef
interface Serial 1/4
 encapsulation frame-relay
 frame-relay intf-type dte
 ip address 10.1.1.1 255.0.0.0
 frame-relay map ip 192.168.2.2 vc-bundle MAIN-1-static
 frame-relay vc-bundle MAIN-1-static
 match precedence
 pvc 100 1a
 precedence other
 pvc 101 1b
 precedence 1
 pvc 102 1c
 precedence 2
 pvc 103 1d
 precedence 3
 pvc 104 1e
 precedence 4
 pvc 105 1f
 precedence 5
 pvc 106 1g
 precedence 6
 pvc 107 1h
 frame-relay vc-bundle MAIN-2-dynamic
 match precedence
 pvc 200
 precedence 0
 pvc 201
 precedence 1
 pvc 202
 precedence 2
 pvc 203
 precedence 3
 pvc 204
 precedence 4
 pvc 205
 precedence 5
 pvc 206
 precedence 6
 pvc 207
 precedence 7
interface Serial 1/4.1 multipoint
 ip address 172.16.1.1 255.0.0.0
 frame-relay map ip 172.17.2.2 vc-bundle MP-3-static
 frame-relay vc-bundle MP-3-static
  match precedence
 pvc 300 3a
 precedence 0
 pvc 301 3b
 precedence 1
 pvc 302 3c
 precedence 2
 pvc 303 3d
 precedence 3
 pvc 304 3e
 precedence 4
 pvc 305 3f
 precedence 5
 pvc 306 3g
 precedence 6
 pvc 307 3h
 precedence 7
interface Serial 1/4.1 multipoint
 frame-relay vc-bundle MP-4-dynamic
 match precedence
 match dscp
 pvc 400 4a
 dscp other
 pvc 401 4b
 dscp 10-19
 pvc 402 4c
 dscp 20-29
 pvc 403 4d
 dscp 30-39
 pvc 404 4e
 dscp 40-49
 pvc 405 4f
 dscp 50-59
 pvc 406 4g
 dscp 60-62
 pvc 407 4h
 dscp 63
 end
interface Serial 1/4.2 point-to-point
 ip address 192.168.2.1 255.0.0.0
 frame-relay vc-bundle P2P-5
 match precedence
 pvc 500 5a
 precedence 0
 pvc 501 5b
 precedence 1
 pvc 502 5c
 precedence 2
 pvc 503 5d
 precedence 3
 pvc 504 5e
 precedence 4
 pvc 505 5f
 precedence 5
 pvc 506 5g
 precedence 6
 pvc 507 5h
 precedence 7
```

PVC Bundle with IP QoS Support with Multiple QoS Parameters Example

The following example shows the configuration of a Frame Relay PVC bundle with DSCP-based mapping. The bundle member PVCs are configured with bumping, protection, and other parameters.

```
interface Serial 1/4.2 point-to-point
 frame-relay vc-bundle BUNDLE-SEFEN
 encapsulation ietf
 match dscp
  pvc 301
 dscp other
 bump explicit 45
 protect group
 class CIR-64000
 pvc 302
 dscp 40-49
 bump explicit 20
 no bump traffic
 protect vc
 inarp
 pvc 303
 dscp 30-39
 bump implicit
 protect group
```

PVC Bundle with MPLS QoS Support Example

The following example shows the configuration of four Frame Relay PVC bundle members with MPLS EXP level support in the PVC bundle named "user1".

```
interface serial 0.1 point-to-point
 encapsulation frame-relay
 ip address 10.1.1.1
 tag-switching ip
 frame-relay vc-bundle user1
 pvc 100 ny-control
 class control
 exp 7
 protect vc
 pvc 101 ny-premium
 class premium
 exp 6-5
 bump explicit 7
 no bump traffic
 protect group
 pvc 102 my-priority
 class priority
 exp 4-2
 protect group
 pvc 103 ny-basic
 class basic
 exp other
```

protect group

Verifying Frame Relay PVC Bundle Configuration Examples

The following examples show output for the commands that can be used to verify Frame Relay PVC bundle configuration.

Sample Output for the show frame-relay vc-bundle Command

The following example shows the Frame Relay PVC bundle named "MP-4-dynamic" with PVC protection applied. Note that in this PVC bundle, DLCI 400 is configured to bump traffic explicitly to the PVC that handles DSCP level 40, which is DLCI 404. All the other DLCIs are configured for implicit bumping. In addition, all the DLCIs are configured to accept bumped traffic.

The asterisk (*) before PVC 4a indicates that this PVC was configured with the **precedence other** command, which means the PVC will handle all levels that are not explicitly configured on other PVCs.

In this example all PVCs are up so the values in the "Active level" fields match the values in the "Config level" fields. If a PVC goes down and its traffic is bumped, the "Active level" field value for the PVC that went down is cleared. The "Active level" field values for the PVC that the traffic bumped to will be updated to include the levels of the PVC that went down.

The first three PVCs in the following example make up a protected group. All three of these PVCs must go down before the bundle will go down. The last two PVCs are protected PVCs: if either of these PVCs go down, the bundle will go down.

```
Router# show frame-relay vc-bundle MP-4-dynamic
MP-4-dynamic on Serial 1/4.1 - Status: UP Match-type: DSCP
Name   DLCI   Config.  Active     Bumping   PG/   CIR   Status
    level   level    to/accept    PV   kbps
*4a   400   0-9   0-9    40/Yes    pg     up
4b   401   10-19   10-19    9/Yes    pg     up
4c   402   20-29   20-29    19/Yes    pg     up
4d   403   30-39   30-39    29/Yes    -     up
4e   404   40-49   40-49    39/Yes    -     up
4f   405   50-59   50-59    49/Yes    -     up
4g   406   60-62   60-62    59/Yes    pv     up
4h   407   63   63    62/Yes    pv     up
Packets sent out on vc-bundle MP-4-dynamic : 0:
Router#
```

The following example shows the detail output of a PVC bundle. Note in this example that because all packet service levels are not handled, and because the PVCs are currently down, this bundle can never come up.

```
Router# show frame-relay vc-bundle x41 detail
x41 on Serial1/1 - Status: DOWN Match-type: DSCP
Name   DLCI   Config.   Active    Bumping    PG/   CIR   Status
    level   level    to/accept    PV   kbps
  410   50-62      49/Yes    -     down
  411   30,32,34,36,3..      29/Yes    -     down
Packets sent out on vc-bundle x41 : 0
Active configuration and statistics for each member PVC
DLCI   Output pkts     Active level
410   0     50-62
411   0     30,32,34,36,38-40
Router#
```

Sample Output for the show frame-relay map Command

The following sample output displays map and connection information for a PVC bundle called "MAIN-1-static":

```
Router# show frame-relay map
```

```
Serial1/4 (up):ip 10.2.2.2 vc-bundle MAIN-1-static, static,
              CISCO, status up
```

Sample Output for the show frame-relay pvc Command

The following sample output indicates that PVC 202 is a member of VC bundle "MAIN-1-static":

```
Router# show frame-relay pvc 202
PVC Statistics for interface Serial1/4 (Frame Relay DTE)
DLCI = 202, DLCI USAGE = LOCAL, PVC STATUS = STATIC, INTERFACE = Serial1/4
  input pkts 0             output pkts 45           in bytes 0
  out bytes 45000          dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  out bcast pkts 0         out bcast bytes 0
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 2000 bits/sec, 2 packets/sec
  pvc create time 00:01:25, last time pvc status changed 00:01:11
  VC-Bundle MAIN-1-static
```

Sample Output for the show adjacency Command

The following is sample output for the **show adjacency** command for a PVC bundle configured on serial subinterface 1/4.1. Each bundle member is listed. The bundle itself is indicated by "incomplete" because no traffic actually transmitted on that entry.

```
Router# show adjacency
      Protocol Interface                 Address
      IP       Serial1/4.1               10.2.2.2(4)
      IP       Serial1/4.1               10.2.2.2(4)
      IP       Serial1/4.1               10.2.2.2(4)
      IP       Serial1/4.1               10.2.2.2(4)
      IP       Serial1/4.1               10.2.2.2(4)
      IP       Serial1/4.1               10.2.2.2(4)
      IP       Serial1/4.1               10.2.2.2(4)
      IP       Serial1/4.1               10.2.2.2(4)
      IP       Serial1/4.1               10.2.2.2(5) (incomplete)
```

Monitoring and Maintaining Frame Relay PVC Bundles Examples

The following examples show output for the **debug frame-relay adjacency**and **debug frame-relay vc-bundle** commands, which can be used to troubleshoot Frame Relay PVC bundle operation. "FR-VCB" indicates output from the **debug frame-relay vc-bundle** command, and "FR-ADJ" indicates output from the **debug frame-relay adjacency**command.

Note Debug messages that are prefixed with "FR_ADJ" (instead of FR-ADJ") or "FR_VCB" (instead of "FR-VCB") indicate serious failures in the Frame Relay PVC bundle performance. Contact the Cisco Technical Assistance Center (TAC) if you see debug messages with these prefixes.

The following is sample output that shows a PVC bundle that uses static map coming up. PVC bundle member 100 comes up first, then the PVC bundle itself can come up.

```
Router# debug frame-relay vc-bundle state-change
Router# debug frame-relay adjacency vc-bundle

00:35:58:FR-VCB:MAIN-1-static:member 100 state changed to UP
00:35:58:FR-VCB:MAIN-1-static:state changed to UP
00:35:58:FR-ADJ:vcb MAIN-1-static:ip 10.2.2.2:adding primary adj
```

```
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:adding adj
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 0 00:35:58:FR-ADJ:vcb
MAIN-1-static:member 100:locking adj at index 1
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 2
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 3
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 4
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 5
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 6
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 7
00:35:58:%FR-5-DLCICHANGE:Interface Serial1/4 - DLCI 100 state changed to ACTIVE
00:35:58:FR-VCB:MAIN-1-static:member 101 state changed to UP
00:35:58:FR-ADJ:vcb MAIN-1-static:ip 10.2.2.2:updating primary adj
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:updating adj
00:35:58:FR-ADJ:vcb MAIN-1-static:member 101:adding adj
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:unlocking adj at index 1
00:35:58:FR-ADJ:vcb MAIN-1-static:member 101:locking adj at index 1
```

The following is sample output that shows a PVC bundle going down. Each bundle member PVC is marked for removal from Cisco Express Forwarding adjacency table, and then the adjacency for the PVC bundle itself is marked for removal. The adjacencies are actually removed from the table later when a background clean-up process runs.

```
00:38:35:FR-VCB:MP-3-static:state changed to DOWN
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 300:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 301:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 302:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 303:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 304:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 305:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:removing primary adj
```

The following is sample output that shows Inverse ARP information for the PVC bundle. PVC bundle member 406 is the only PVC in the bundle to handle Inverse ARP packets. The Inverse ARP packets coming in on other bundle member PVCs are dropped.

```
00:23:48:FR-VCB:MP-4-dynamic:inarp received on elected member 406
00:23:48:FR-VCB:MP-4-dynamic:installing dynamic map
00:23:48:FR-VCB:MP-4-dynamic:dropping inarp received on member 407
00:23:52:FR-VCB:MP-4-dynamic:sending inarp pkt on member 406
```

In the following example the PVC bundle goes down because the protected group goes down. All information about active transmission on each PVC is removed.

```
00:58:27:FR-VCB:MP-4-dynamic:member 402 state changed to DOWN
00:58:27:FR-VCB:MP-4-dynamic:protected group is DOWN
00:58:27:FR-VCB:MP-4-dynamic:state changed to DOWN
00:58:27:FR-VCB:MP-4-dynamic:active table reset
```

Additional References

Related Documents

Related Topic	Document Title
Frame Relay configuration tasks	*"Configuring Frame Relay chapter in the* Cisco IOS Wide-Area Networking Configuration Guide , Release 12.2

Related Topic	Document Title
Frame Relay commands	*"Frame Relay Commands chapter in the* Cisco IOS Wide-Area Networking Command Reference , Release 12.2
Frame Relay PVC interface priority queueing configuration tasks	"Configuring Weighted Fair Queueing " section in the Congestion Management chapter in the Cisco IOS Quality of Service Configuration Guide , Release 12.2

Standards

Standards	Title
None	--

MIBs

MIBs	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
None	--

Technical Assistance

Description	Link
The Cisco Technical Support & Documentation website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Feature Information for Frame Relay PVC Bundles with QoS Support for IP and MPLS

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 3: Feature Information for Frame Relay PVC Bundles with QoS Support for IP and MPLS

Feature Name	Releases	Feature Information
Frame Relay VC Bundling	12.2(13)T 12.2(28)SB 15.0(1)S	Frame Relay permanent virtual circuit (PVC) bundle functionality allows you to associate a group of Frame Relay PVCs with a single next-hop address. The following commands were introduced or modified: **bump (Frame Relay VC-bundle-member)**, **class**, **debug frame-relay adjacency**, **debug frame-relay vc-bundle**, **dscp (Frame Relay VC-bundle-member)**, **encapsulation (Frame Relay VC-bundle)**, **exp**, **frame-relay inverse-arp**, **frame-relay map**, **frame-relay vc-bundle**, **inarp (Frame Relay VC-bundle-member)**, **match**, **precedence (Frame Relay VC-bundle-member)**, **protect (Frame Relay VC-bundle-member)**, **pvc (Frame Relay VC-bundle)**, **show frame-relay ip rtp header-compression**, **show frame-relay ip tcp header-compression**, **show frame-relay map**, **show frame-relay pvc**, **show frame-relay vc-bundle**..

Glossary

DLCI --data-link connection identifier. Value that specifies a permanent virtual circuit (PVC) or switched virtual circuit (SVC) in a Frame Relay network.

FIFO queueing -- First-in, first-out queueing. FIFO involves buffering and forwarding of packets in the order of arrival. FIFO embodies no concept of priority or classes of traffic. There is only one queue, and all packets are treated equally. Packets are sent out an interface in the order in which they arrive.

Frame Relay traffic shaping --See FRTS.

FRF.12 --The FRF.12 Implementation Agreement was developed to allow long data frames to be fragmented into smaller pieces and interleaved with real-time frames. In this way, real-time voice and nonreal-time data frames can be carried together on lower-speed links without causing excessive delay to the real-time traffic.

FRTS --Frame Relay traffic shaping. FRTS uses queues on a Frame Relay network to limit surges that can cause congestion. Data is buffered and then sent into the network in regulated amounts to ensure that the traffic will fit within the promised traffic envelope for the particular connection.

PIPQ --Permanent virtual circuit (PVC) interface priority queueing. An interface-level priority queueing scheme in which prioritization is based on destination PVC rather than packet contents.

quality of service --Measure of performance for a transmission system that reflects its transmission quality and service availability.

VoFR --Voice over Frame Relay. Enables a router to carry voice traffic over a Frame Relay network. When voice traffic is sent over Frame Relay, the voice traffic is segmented and encapsulated for transit across the Frame Relay network using FRF.12 encapsulation.

Voice over Frame Relay --See VoFR.

WFQ --weighted fair queueing. Congestion management algorithm that identifies conversations (in the form of traffic streams), separates packets that belong to each conversation, and ensures that capacity is shared fairly among these individual conversations. WFQ is an automatic way of stabilizing network behavior during congestion and results in increased performance and reduced retransmission.

WRED --Weighted Random Early Detection. Combines IP Precedence and standard Random Early Detection (RED) to allow for preferential handling of voice traffic under congestion conditions without exacerbating the congestion. WRED uses and interprets IP Precedence to give priority to voice traffic over data traffic, dropping only data packets.

CHAPTER 6

Frame Relay PVC Interface Priority Queueing

The Frame Relay PVC Interface Priority Queueing feature provides an interface-level priority queueing scheme in which prioritization is based on destination permanent virtual circuit (PVC) rather than packet contents.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Feature Overview

The FR PIPQ feature provides an interface-level priority queueing scheme in which prioritization is based on destination permanent virtual circuit (PVC) rather than packet contents. For example, FR PIPQ allows you to configure a PVC transporting voice traffic to have absolute priority over a PVC transporting signalling traffic, and a PVC transporting signalling traffic to have absolute priority over a PVC transporting data.

FR PIPQ provides four levels of priority: high, medium, normal, and low. The Frame Relay packet is examined at the interface for the data-link connection identifier (DLCI) value. The packet is then sent to the correct priority queue based on the priority level configured for that DLCI.

Note When using FR PIPQ, configure the network so that different types of traffic are transported on separate PVCs. FR PIPQ is not meant to be used when an individual PVC carries different traffic types that have different quality of service (QoS) requirements.

You assign priority to a PVC within a Frame Relay map class. All PVCs using or inheriting that map class will be classed according to the configured priority. If a PVC does not have a map class associated with it, or if the map class associated with it does not have priority explicitly configured, then the packets on that PVC will be queued on the default "normal" priority queue.

If you do not enable FR PIPQ on the interface using the **frame-relay interface-queue priority**command in interface configuration mode, configuring PVC priority within a map class will not be effective. At this time you have the option to also set the size (in maximum number of packets) of the four priority queues.

FR PIPQ works with or without Frame Relay traffic shaping (FRTS) and FRF.12. The interface-level priority queueing takes the place of the FIFO queueing or dual FIFO queueing normally used by FRTS and FRF.12. PVC priority assigned within FR PIPQ takes precedence over FRF.12 priority, which means that all packets destined for the same PVC will be queued on the same interface queue whether they were fragmented or not.

Note Although high priority PVCs most likely will transport only small packets of voice traffic, you may want to configure FRF.12 on these PVCs anyway to guard against any unexpectedly large packets.

Benefits

FR PIPQ provides four levels of PVC priority: high, medium, normal, and low. This method of queueing ensures that time/delay-sensitive traffic such as voice has absolute priority over signalling traffic, and that signalling traffic has absolute priority over data traffic, providing different PVCs are used for the different types of traffic.

Restrictions

The following restrictions apply to FR PIPQ:

- FR PIPQ is not supported on loopback or tunnel interfaces, or interfaces that explicitly disallow priority queueing.
- FR PIPQ is not supported with hardware compression.
- FR PIPQ cannot be enabled on an interface that is already configured with queueing other than FIFO queueing. FR PIPQ can be enabled if WFQ is configured, as long as WFQ is the default interface queueing method.

Related Features and Technologies

The following features and technologies are related to FR PIPQ:

- FRTS
- FRF.12

Related Documents

The following documents provide information related to FR PIPQ:

- *Cisco IOS Wide-Area Networking Configuration Guide* , Release 12.1
- *Cisco IOS Wide-Area Networking Command Reference* , Release 12.1

Supported Platforms

- Cisco 1000 series
- Cisco 1400 series
- Cisco 1600
- Cisco 1700
- Cisco 1750
- Cisco 2500
- Cisco 2600
- Cisco 3600
- Cisco 3810
- Cisco 4500
- Cisco 4700
- Cisco 7200
- Cisco 7500 (in nondistributed mode)

Supported Standards and MIBs and RFCs

Standards

No new or modified standards are supported by this feature.

MIBs

No new or modified MIBs are supported by this feature.

For descriptions of supported MIBs and how to use MIBs, see the Cisco MIB web site on CCO at http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml.

RFCs

No new or modified standards are supported by this feature.

Prerequisites

- PVCs should be configured to carry a single type of traffic.
- The network should be configured with adequate call admission control to prevent starvation of any of the priority queues.

Configuration Tasks

Configuring PVC Priority in a Map Class

To configure PVC priority within a map class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **map-class frame-relay** *map-class-name*
2. Router(config-map-class)# **frame-relay interface-queue priority** {**high** | **medium**| **normal** | **low**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **map-class frame-relay** *map-class-name*	Specifies a Frame Relay map class.
Step 2	Router(config-map-class)# **frame-relay interface-queue priority** {**high** | **medium**| **normal** | **low**}	Assigns a PVC priority level to a Frame Relay map class.

Enabling FR PIPQ and Setting Queue Limits

To enable FR PIPQ and set the priority queue sizes, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **interface** *type number* [*name-tag*]
2. Router(config-if)# **encapsulation frame-relay**[**cisco** | **ietf**]
3. Router(config-if)# **frame-relay interface-queue priority** [*high-limit medium-limit normal-limit low-limit*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number* [*name-tag*]	Configures an interface type and enters interface configuration mode.
Step 2	Router(config-if)# **encapsulation frame-relay**[**cisco** | **ietf**]	Enables Frame Relay encapsulation.
Step 3	Router(config-if)# **frame-relay interface-queue priority** [*high-limit medium-limit normal-limit low-limit*]	Enables FR PIPQ and sets the priority queue limits.

Assigning a Map Class to a PVC

To assign a map class to a specific PVC, use the following commands beginning in interface configuration mode:

SUMMARY STEPS

1. Router(config-if)# **frame-relay interface-dlci** *dlci*
2. Router(config-fr-dlci)# **class** *map-class-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config-if)# **frame-relay interface-dlci** *dlci*	Specifies a single PVC on a Frame Relay interface.
Step 2	Router(config-fr-dlci)# **class** *map-class-name*	Associates a map class with a specified PVC.

Verifying FR PIPQ

To verify the configuration of FR PIPQ, use one or more of the following commands in privileged EXEC mode:

Command	Purpose
`Router#` **show frame-relay pvc** [**interface** *interface*] [*dlci*]	Displays statistics about PVCs for Frame Relay interfaces.
`Router#` **show interfaces** [*type number*] [*first*] [*last*]	Displays the statistical information specific to a serial interface.
`Router#` **show queueing** [**custom** | **fair** | **priority** | **random-detect** [**interface** *atm_subinterface* [**vc** [[*vpi*/] *vci*]]]]	Lists all or selected configured queueing strategies.

Monitoring and Maintaining FR PIPQ

To monitor and maintain FR PIPQ, use one or more of the following commands in privileged EXEC mode:

Command	Purpose
`Router#` **debug priority**	Debugs priority output queueing.
`Router#` **show frame-relay pvc** [**interface** *interface*] [*dlci*]	Displays statistics about PVCs for Frame Relay interfaces.
`Router#` **show interfaces** [*type number*] [*first*] [*last*]	Displays the statistical information specific to a serial interface.
`Router#` **show queue** *interface-name interface-number* [**vc** [*vpi*/] *vci*] [*queue-number*]	Displays the contents of packets inside a queue for a particular interface or VC.
`Router#` **show queueing** [**custom** | **fair** | **priority** | **random-detect** [**interface** *atm_subinterface* [**vc** [[*vpi*/] vci]]]]	Lists all or selected configured queueing strategies.

Configuration Examples

FR PIPQ Configuration Example

This example shows the configuration of four PVCs on serial interface 0. DLCI 100 is assigned high priority, DLCI 200 is assigned medium priority, DLCI 300 is assigned normal priority, and DLCI 400 is assigned low priority.

The following commands configure Frame Relay map classes with PVC priority levels:

```
Router(config)# map-class frame-relay HI
Router(config-map-class)# frame-relay interface-queue priority high
Router(config-map-class)# exit
Router(config)# map-class frame-relay MED
Router(config-map-class)# frame-relay interface-queue priority medium
Router(config-map-class)# exit
Router(config)# map-class frame-relay NORM
Router(config-map-class)# frame-relay interface-queue priority normal
Router(config-map-class)# exit
Router(config)# map-class frame-relay LOW
Router(config-map-class)# frame-relay interface-queue priority low
Router(config-map-class)# exit
```

The following commands enable Frame Relay encapsulation and FR PIPQ on serial interface 0. The sizes of the priority queues are set at a maximum of 20 packets for the high priority queue, 40 for the medium priority queue, 60 for the normal priority queue, and 80 for the low priority queue.

```
Router(config)# interface Serial0
Router(config-if)# encapsulation frame-relay
Router(config-if)# frame-relay interface-queue priority 20 40 60 80
```

The following commands assign priority to four PVCs by associating the DLCIs with the configured map classes:

```
Router(config-if)# frame-relay interface-dlci 100
Router(config-fr-dlci)# class HI
Router(config-fr-dlci)# exit
Router(config-if)# frame-relay interface-dlci 200
Router(config-fr-dlci)# class MED
Router(config-fr-dlci)# exit
Router(config-if)# frame-relay interface-dlci 300
Router(config-fr-dlci)# class NORM
Router(config-fr-dlci)# exit
Router(config-if)# frame-relay interface-dlci 400
Router(config-fr-dlci)# class LOW
Router(config-fr-dlci)# exit
```

Glossary

DLCI --data-link connection identifier. Value that specifies a permanent virtual circuit (PVC) or switched virtual circuit (SVC) in a Frame Relay network.

FIFO queueing -- First-in, first-out queueing. FIFO involves buffering and forwarding of packets in the order of arrival. FIFO embodies no concept of priority or classes of traffic. There is only one queue, and all packets are treated equally. Packets are sent out an interface in the order in which they arrive.

Frame Relay traffic shaping --See FRTS.

FRF.12 --The FRF.12 Implementation Agreement was developed to allow long data frames to be fragmented into smaller pieces and interleaved with real-time frames. In this way, real-time voice and nonreal-time data frames can be carried together on lower-speed links without causing excessive delay to the real-time traffic.

FRTS --Frame Relay traffic shaping. FRTS uses queues on a Frame Relay network to limit surges that can cause congestion. Data is buffered and then sent into the network in regulated amounts to ensure that the traffic will fit within the promised traffic envelope for the particular connection.

PIPQ --Permanent virtual circuit (PVC) interface priority queueing. An interface-level priority queueing scheme in which prioritization is based on destination PVC rather than packet contents.

quality of service --Measure of performance for a transmission system that reflects its transmission quality and service availability.

VoFR --Voice over Frame Relay. Enables a router to carry voice traffic over a Frame Relay network. When voice traffic is sent over Frame Relay, the voice traffic is segmented and encapsulated for transit across the Frame Relay network using FRF.12 encapsulation.

Voice over Frame Relay --See VoFR.

WFQ --weighted fair queueing. Congestion management algorithm that identifies conversations (in the form of traffic streams), separates packets that belong to each conversation, and ensures that capacity is shared fairly among these individual conversations. WFQ is an automatic way of stabilizing network behavior during congestion and results in increased performance and reduced retransmission.

WRED --Weighted Random Early Detection. Combines IP Precedence and standard Random Early Detection (RED) to allow for preferential handling of voice traffic under congestion conditions without exacerbating the congestion. WRED uses and interprets IP Precedence to give priority to voice traffic over data traffic, dropping only data packets.

CHAPTER 7

Frame Relay IP RTP Priority

This feature module describes the Frame Relay IP RTP Priority feature.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Feature Overview

The Frame Relay IP RTP Priority feature provides a strict priority queueing scheme on a Frame Relay permanent virtual circuit (PVC) for delay-sensitive data such as voice. Voice traffic can be identified by its Real-Time Transport Protocol (RTP) port numbers and classified into a priority queue configured by the **frame-relay ip rtp priority**command. The result of using this feature is that voice is serviced as strict priority in preference to other nonvoice traffic.

This feature extends the functionality offered by the **ip rtp priority** command by supporting Frame Relay PVCs. This feature allows you to specify a range of User Datagram Protocol (UDP) ports whose voice traffic is guaranteed strict priority service over any other queues or classes using the same output interface. Strict priority means that if packets exist in the priority queue, they are dequeued and sent first--that is, before packets in other queues are dequeued.

Benefits

The strict priority queueing scheme allows delay-sensitive data such as voice to be dequeued and sent first--that is, before packets in other queues are dequeued. Delay-sensitive data is given preferential treatment over other traffic. This process is performed on a per-PVC basis, rather than at the interface level.

Related Features and Technologies

The Frame Relay IP RTP Priority feature is related to the following features:

- IP RTP Priority
- Class-based weighted fair queueing (CBWFQ)
- Priority queueing
- Weighted fair queueing (WFQ)

Related Documents

- *Quality of Service Solutions Configuration Guide* , Cisco IOS Release 12.0
- *Quality of Service Solutions Command Reference* , Cisco IOS Release 12.0
- *Class-Based Weighted Fair Queueing*
- *IP RTP Priority*

Supported Platforms

- Cisco 1003
- Cisco 1004
- Cisco 1005
- Cisco 1600 series
- Cisco 2500 series
- Cisco 2600 series
- Cisco 3600 series
- Cisco 3800 series
- Cisco 4000 series (Cisco 4000, 4000-M, 4500, 4500-M, 4700, 4700-M)
- Cisco 5200 series

- Cisco 7000 series
- Cisco 7200 series
- Cisco 7500 series

This feature runs on the platforms listed. However, it is most useful on voice supported platforms, such as the Cisco 2600 series, Cisco 3600 series, Cisco 7200 series, and Cisco 7500 Route Switch Processor (RSP) series.

Supported Standards and MIBs and RFCs

Standards

None

MIBs

No new or modified MIBs are supported by this feature.

RFCs

None

Prerequisites

Frame Relay traffic shaping (FRTS) and Frame Relay Fragmentation (FRF.12) must be configured before the Frame Relay IP RTP Priority feature is used.

Configuration Tasks

Configuring Frame Relay IP RTP Priority

To reserve a strict priority queue on a Frame Relay PVC for a set of RTP packet flows belonging to a range of UDP destination ports, use the following command in map-class configuration mode.

Note Because the **frame-relay ip rtp priority** command gives absolute priority over other traffic, it should be used with care. In the event of congestion, if the traffic exceeds the configured bandwidth, then all the excess traffic is dropped.

Command	Purpose
`Router(config-map-class)#` **frame-relay ip rtp priority** *starting-rtp-port-number port-number-range bandwidth*	Reserves a strict priority queue for a set of RTP packet flows belonging to a range of UDP destination ports.

Verifying Frame Relay IP RTP Priority

To verify the Frame Relay IP RTP Priority feature, use one of the following commands in EXEC mode:

Command	Purpose
`Router#` **show frame relay pvc**	Displays statistics about PVCs for Frame Relay interfaces.
`Router#` **show queue** *interface-type interface-number*	Displays fair queueing configuration and statistics for a particular interface.
`Router#` **show traffic-shape queue**	Displays information about the elements queued at a particular time at the VC data link connection identifier (DLCI) level.

Monitoring and Maintaining Frame Relay IP RTP Priority

To tune your RTP bandwidth or decrease RTP traffic if the priority queue is experiencing drops, use the following command in EXEC mode:

Command	Purpose
`Router#` **debug priority**	Displays priority queueing output if packets are dropped from the priority queue.

Configuration Examples

Frame Relay IP RTP Priority Configuration Example

The following example first configures the Frame Relay map class called voip and then applies the map class to PVC 100 to provide strict priority service to matching RTP packets:

```
map-class frame-relay voip
 frame-relay cir 256000
 frame-relay bc 2560
 frame-relay be 600
 frame-relay mincir 256000
 no frame-relay adaptive-shaping
 frame-relay fair-queue
 frame-relay fragment 250
 frame-relay ip rtp priority 16384 16380 210
interface Serial5/0
 ip address 10.10.10.10 255.0.0.0
 no ip directed-broadcast
 encapsulation frame-relay
 no ip mroute-cache
 load-interval 30
 clockrate 1007616
 frame-relay traffic-shaping
 frame-relay interface-dlci 100
  class voip
 frame-relay ip rtp header-compression
 frame-relay intf-type dce
```

In this example, RTP packets on PVC 100 with UDP ports in the range 16384 to 32764 will be matched and given strict priority service.

CHAPTER 8

PPP over Frame Relay

The PPP over Frame Relay feature allows a router to establish end-to-end Point-to-Point Protocol (PPP) sessions over Frame Relay.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for PPP over Frame Relay

Before you can configure PPP over Frame Relay, Frame Relay must be enabled on the router using the **encapsulation frame-relay** command.

Restrictions for PPP over Frame Relay

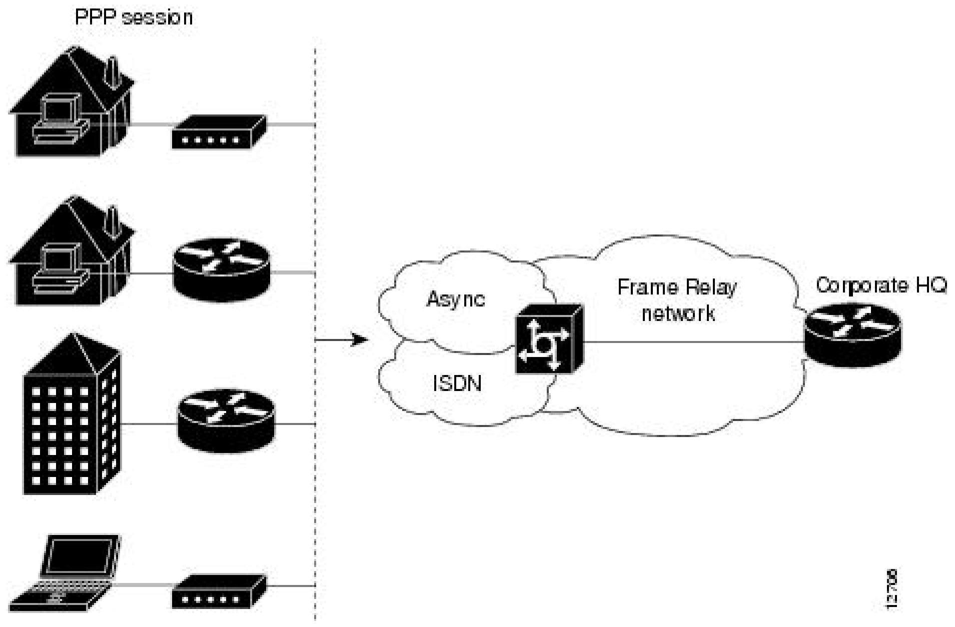
- Only Frame Relay permanent virtual circuits (PVCs) are supported.
- Only the Internet Protocol (IP) is supported.

Information About PPP over Frame Relay

Overview of PPP over Frame Relay

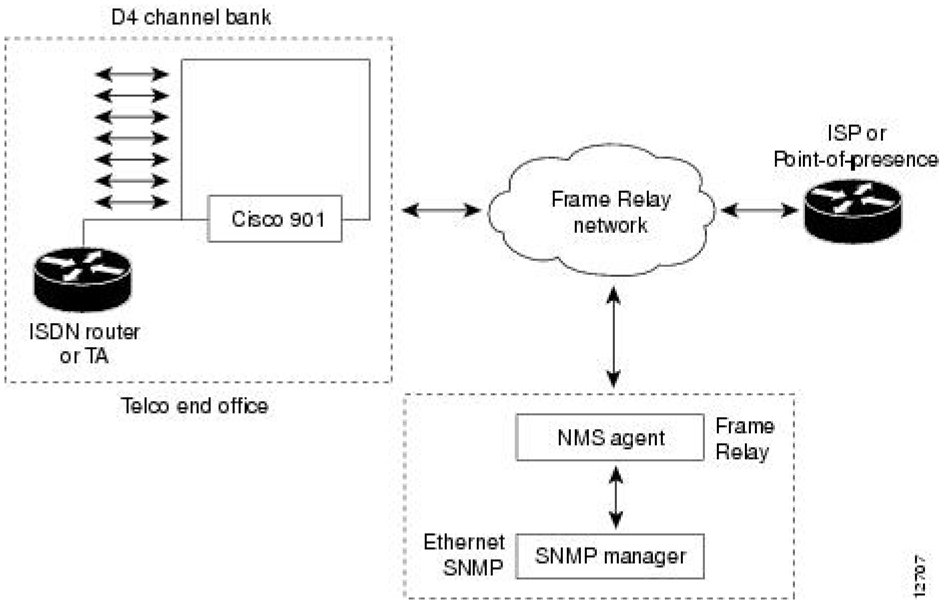
The PPP over Frame Relay feature allows a router to establish end-to-end Point-to-Point Protocol (PPP) sessions over Frame Relay. IP datagrams are transported over the PPP link using RFC 1973 compliant Frame Relay framing. This feature is useful for remote users running PPP to access their Frame Relay corporate networks as shown in the figure below.

Figure 16: PPP over Frame Relay Scenario



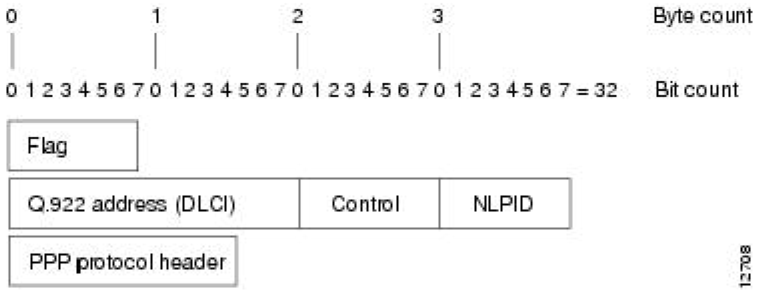
The figure below shows a connectivity scenario using the Cisco 90i D4 channel card, which is capable of supporting Integrated Services Digital Network (ISDN) Digital Service Loop (DSL), PPP, or Frame Relay, which connects to an Internet Service Provider (ISP) or corporate network.

Figure 17: PPP over Frame Relay Frame Format



PPP over Frame Relay is compliant with the functionality and encapsulation specifications as outlined in RFC 1973. The frame format is shown in the figure below.

Figure 18: PPP over Frame Relay Frame Format



A PPP connection is established over the Frame Relay PVC. The PPP session does not occur unless the associated Frame Relay PVC is in an "active" state. The Frame Relay VC can coexist with other circuits using different Frame Relay encapsulation methods, such as RFC 1490 and Cisco proprietary, over the same Frame Relay link. There can be multiple PPP-in-Frame-Relay circuits existing on one Frame Relay link.

One PPP connection resides on one virtual access interface, which is internally created from a virtual template interface. A virtual access interface is cloned from a virtual template interface. The virtual access interfaces is coexistent with the creation of the Frame Relay circuit when the corresponding DLCI is configured. One virtual template interface, containing all the necessary PPP and network protocol information, is shared by multiple virtual access interfaces. Hardware compression and fancy queuing algorithms, such as weighted fair queuing, custom queuing, and priority queuing, are not applied to virtual access interfaces. Once a Frame

Relay circuit is established using PPP over Frame Relay, all incoming and outgoing packets on this circuit are under RFC 1973 PPP-in-Frame-Relay encapsulation compliance until this DLCI is removed from the configuration.

The breakdown of the Frame Relay frame format components is listed in the table below.

Table 4: PPP Frame Relay Format Descriptions

Field	Description
Flag	A single byte that indicates the beginning or end of a frame.
Address	A two-byte field that indicates the logical connection that maps to the physical channel; the DLCI.
Control	A single byte that calls for transmission of user data. PPP over Frame Relay uses a value of 0X03, which indicates that the frame is an unnumbered information (UI) frame.
NLPID	Network layer protocol ID, which is a single byte that uniquely identifies a PPP packet to Frame Relay.
PPP protocol	Identifies the PPP packet type.

Benefits

- Allows end-to-end PPP sessions over Frame Relay.
- Supports the 90i IDSL Channel Unit that supports both Frame Relay and PPP on an ISDN DSL.

How to Configure PPP over Frame Relay

Enabling PPP over Frame Relay

Perform the following task to configure the physical interface that carries the PPP session and links to the appropriate virtual template interface.

Before You Begin

After you configure the Cisco router or access server for Frame Relay encapsulation, you must configure the physical interface with the PVC and apply a virtual template with PPP encapsulation to the DLCI that it applies to.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface type** *number* **.** *subinterface-number* {**multipoint** | **point-to-point**}
4. **frame-relay interface-dlci** *dlci* [**ppp** *virtual-template-name-string*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

Step 3	**interface type** *number* **.** *subinterface-number* {**multipoint** | **point-to-point**}
	
	Example:
	
	`Router(config)# interface serial 2.1 point-to-point`
		Specifies the interface and enters interface configuration mode.

Step 4	**frame-relay interface-dlci** *dlci* [**ppp** *virtual-template-name-string*] **Example:** `Router(config-if)# frame-relay interface-dlci`	Defines the PVC and maps it to the virtual template.

Step 5	**end**
	
	Example:
	
	`Router(config-if)# exit`
		Exits the current configuration mode and returns to privileged EXEC mode.

Configuration Examples for PPP over Frame Relay

Example PPP over Frame Relay DTE

The following example configures a router as a data terminating equipment (DTE) device for PPP over Frame Relay. Subinterface 2.1 contains the necessary DLCI and virtual template information. The virtual template interface (interface virtual-template 1) contains the PPP information that is applied to the PPP session associated with DLCI 32 on serial subinterface 2.1.

```
interface serial 2
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
interface serial 2.1 point-to-point
 frame-relay interface-dlci 32 ppp virtual-template1
!
interface Virtual-Template1
 ip unnumbered ethernet 0
 ppp authentication chap pap
```

Note By default, the encapsulation type for a virtual template interface is PPP encapsulation; therefore, the **encapsulation ppp** command will not show up when viewing the running configuration on the router.

Example PPP over Frame Relay DCE

The following example configures a router to act as a data communications equipment (DCE) device. Typically, a router is configured as a DCE if connecting directly to another router or if connected to a 90i D4 channel unit, which is connected to a telco channel bank. The three commands required for this type of configuration are **frame-relay switching**, **frame-relay intf-type dce**, and **frame-relay route** commands.

```
frame-relay switching
!
interface Serial2/0:0
 no ip address
 encapsulation frame-relay IETF
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 31 interface Serial1/2 100
 frame-relay interface-dlci 32 ppp Virtual-Template1
!
interface Serial2/0:0.2 point-to-point
 no ip address
 frame-relay interface-dlci 40 ppp Virtual-Template2
!
interface Virtual-Template1
 ip unnumbered Ethernet0/0
 peer default ip address pool default
 ppp authentication chap pap
 !
interface Virtual-Template2
 ip address 209.165.200.225 255.255.255.224
 ppp authentication chap pap
```

Note By default, the encapsulation type for a virtual template interface is PPP encapsulation; therefore, the **encapsulation ppp** command will not show up when viewing the running configuration on the router.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
WAN commands	*Cisco IOS Wide-Area Networking Command Reference*
Configuring Frame Relay	Configuring Frame Relay

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1973	*PPP in Frame Relay*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for PPP over Frame Relay

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 5: Feature Information for PPP over Frame Relay

Feature Name	Releases	Feature Information
PPP over Frame Relay	11.3(4)T 15.0(1)S	The PPP over Frame Relay feature allows a router to establish end-to-end PPP sessions over Frame Relay. The following commands were introduced or modified: **debug frame-relay ppp**, **frame-relay interface-dlci**, **show frame-relay pvc**.

Glossary

data-link connection identifier (DLCI) --A value that specifies a PVC or SVC in a Frame Relay network. In the basic Frame Relay specification, DLCIs are locally significant (connected devices might use different values to specify the same connection). In the LMI extended specification, DLCIs are globally significant (DLCIs specify individual end devices).

Integrated Services Digital Network (ISDN) --Communication protocols offered by telephone companies that permit telephone networks to carry data, voice, and other source traffic.

Link Control Protocol (LCP) --A protocol that establishes, configures, and tests data link connections used by PPP.

permanent virtual circuit (PVC) --Virtual circuit that is permanently established. PVCs save bandwidth associated with circuit establishment and teardown in situations where certain virtual circuits must exist all the time.

Point-to-Point Protocol (PPP) --A protocol that encapsulates network layer protocol information over point-to-point links. The RFC for PPP is RFC 1661.

virtual circuit (VC) -- A logical circuit created to ensure reliable communication between two network devices. A virtual circuit can be either permanent (a PVC) or switched (an SVC). Virtual circuits are used in Frame Relay and X.25.

CHAPTER 9

Multilink Frame Relay FRF.16.1

The Multilink Frame Relay (FRF.16.1) feature introduces functionality based on the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16.1). This feature provides a cost-effective way to increase bandwidth for particular applications by enabling multiple serial links to be aggregated into a single bundle of bandwidth. Multilink Frame Relay (MFR) is supported on User-to-Network Interfaces (UNI) and Network-to-Network Interfaces (NNI) in Frame Relay networks.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Multilink Frame Relay FRF.16.1

- Multilink Frame Relay must be configured on the peer device.

Restrictions for Multilink Frame Relay FRF.16.1

- ISDN interfaces and any type of virtual interface cannot be a bundle link.
- Frame Relay fragmentation (FRF.12) is not supported in Cisco IOS releases 12.0(17)S, 12.2(8)T, and 12.2(14)S.
- The multilink Frame Relay MIB (RFC 3020) is not supported.
- FRF.9 hardware compression over multilink Frame Relay is not supported.

Information About Multilink Frame Relay FRF.16.1

Benefits of Multilink Frame Relay FRF.16.1

Flexible Pool of Bandwidth

By combining multiple physical interfaces into a bundle, you can design a Frame Relay interface that has more bandwidth than is available from any single physical interface. For example, many new network applications require more bandwidth than is available on a T1 line. One option is to invest in a T3 line; however, T3 lines can be expensive and are not available in some locations. Multilink Frame Relay provides a cost-effective solution to this problem by allowing multiple T1 lines to be aggregated into a single bundle of bandwidth.

Greater Service Resilience When Links Fail

Greater service resilience is provided when multiple physical interfaces are provisioned as a single bundle. When a link fails, the bundle continues to support the Frame Relay service by transmitting across the remaining bundle links.

Link Integrity Protocol Control Messages

For link management, each end of a bundle link follows the MFR Link Integrity Protocol and exchanges link-control messages with its peer (the other end of the bundle link). For a bundle link to be brought up, each end of the link must complete an exchange of ADD_LINK and ADD_LINK_ACK messages. To maintain the link, both ends periodically initiate the exchange of HELLO and HELLO_ACK messages. This exchange of hello messages and acknowledgments serves as a keepalive mechanism for the link. If a router is sending hello messages but not receiving acknowledgments, it will resend the hello message up to a configured maximum number of times. If the router exhausts the maximum number of retries, the bundle link line protocol is considered down (nonoperational).

The bundle link interface's line protocol status is considered up (operational) when the peer device acknowledges that it will use the same link for the bundle. The line protocol remains up when the peer device acknowledges the hello messages from the local router.

The bundle interface's line protocol status is considered up when the Frame Relay data-link layer at the local router and peer device is synchronized using the Local Management Interface (LMI), when LMI is enabled. The bundle line protocol remains up as long as the LMI keepalives are successful.

Variable Bandwidth Class Support

Multilink Frame Relay (FRF.16.1) variable bandwidth class support allows you to specify the criterion used to activate or deactivate a Frame Relay bundle. Consistent with the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16.1), bandwidth classes A (single link), B (all links), and C (threshold) are supported.

Class A (Single Link)

The Frame Relay bundle is provisioned when one or more bundle links indicate by issuing a BL_ACTIVATE message that operational bandwidth is available. When this occurs, the bundle emulates a physical link by issuing a PH_ACTIVATE message to the data-link layer.

When the operational bandwidth of a bundle link fails to meet operational requirements (for instance, if it is in rollback mode), the bundle link issues a BL_DEACTIVATE message. When all bundle links are down in a class A bundle, a PH_DEACTIVATE message is sent to the data-link layer, indicating that the Frame Relay bundle cannot accept frames.

Class B (All Links)

The Frame Relay bundle is provisioned when all bundle links indicate by issuing a BL_ACTIVATE message that operational bandwidth is available. When this occurs, the bundle emulates a physical link by issuing a PH_ACTIVATE message to the data-link layer.

When the operational bandwidth of a bundle link fails to meet operational requirements (for instance, if it is in loopback mode), the bundle link issues a BL_DEACTIVATE message. When any bundle link is down in a class B bundle, a PH_DEACTIVATE message is sent to the data-link layer, indicating that the Frame Relay bundle cannot accept frames.

Class C (Threshold)

The Frame Relay bundle is provisioned when the minimum number of links in the configured bundle issue a BL_ACTIVATE message. When this occurs, the bundle emulates a physical link by issuing a PH_ACTIVATE message to the data-link layer.

When the number of bundle links that are issuing a BL_ACTIVATE message falls below the configured threshold value, a PH_DEACTIVATE message is sent to the data-link layer, indicating that the Frame Relay bundle cannot accept frames.

Load Balancing with Multilink Frame Relay FRF.16.1

Multilink Frame Relay provides load balancing across the bundle links within a bundle. If a bundle link chosen for transmission happens to be busy transmitting a long packet, the load-balancing mechanism can try another link, thus solving the problems seen when delay-sensitive packets have to wait.

How to Enable Multilink Frame Relay FRF.16.1

Configuring a Multilink Frame Relay Bundle

To configure the bundle interface for multilink Frame Relay, perform the steps in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface mfr** *interface-number*
4. Do one of the following:
    - **frame-relay multilink bandwidth-class** [**a** | **b** | **c** [*threshold*]]
5. **frame-relay intf-type dce**
6. **frame-relay multilink bid** *name*
7. **frame-relay multilink output-threshold** *bytes*
8. **interface mfr** *interface-number* **.** *subinterface-number* **point-to-point**
9. **ip address** *ip-address mask*
10. **frame-relay interface-dlci** *dlci*
11. **end**
12. **show frame-relay multilink**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface mfr** *interface-number* **Example:** `Router(config)# interface mfr mfr1`	Configures a multilink Frame Relay bundle interface.

	Command or Action	Purpose

Step 4	Do one of the following:
	• **frame-relay multilink bandwidth-class** [**a** | **b** | **c** [*threshold*]]
	Example:
	`Router(config-if)# frame-relay multilink bandwidth-class a`
	Example:
	Example:
	Example:
	`Router(config-if)# frame-relay multilink bandwidth-class b`
	Example:
	Example:
	Example:
	`Router(config-if)# frame-relay multilink bandwidth-class c 3`
		(Optional) Specifies the bandwidth class criterion used to activate or deactivate a Frame Relay bundle.
		• Class A (single link)--The bundle will activate when any bundle link is up and will deactivate when all bundle links are down (default).
		• Class B (all links)--The bundle will activate when all bundle links are up and will deactivate when any bundle link is down.
		• Class C (threshold)--The bundle will activate when the minimum configured number of bundle links is up (the threshold) and will deactivate when the minimum number of configured bundle links fails to meet the threshold.
		Note If no bandwidth class criterion is specified by using the **frame-relay multilink bandwidth-class**command, the Frame Relay bundle will default to class A (single link).

Step 5	**frame-relay intf-type dce** **Example:** `Router(config-if)# frame-relay intf-type dce`	Configures a device to function as the data circuit-terminating equipment (DCE). • Only one end of a link should be configured as the DCE. The other end will function as the data terminal equipment (DTE), which is the default setting. • This command can be used only if Frame Relay switching has been enabled by entering the **frame-relay switching** command in global configuration mode.

	Command or Action	Purpose
Step 6	**frame-relay multilink bid** *name* **Example:** `Router(config-if)# frame-relay multilink bid router1`	(Optional) Assigns a bundle identification name to a multilink Frame Relay bundle. • The bundle identification (BID) will not go into effect until the interface has gone from the "down" state to the "up" state. One way to bring the interface down and back up again is by using the **shutdown** and **no shutdown** commands in interface configuration mode.
Step 7	**frame-relay multilink output-threshold** *bytes* **Example:** `Router(config-if)# frame-relay multilink output-threshold 500`	(Optional) Configures the number of bytes that a bundle link will transmit before the load-balancing mechanism causes transmission to roll over to the next available link. • When configured on the bundle interface, this command applies to all bundle links in the bundle.
Step 8	**interface mfr** *interface-number* **.** *subinterface-number* **point-to-point** **Example:** `Router(config-if)# interface mfr1.1 point-to-point`	Configures a point-to-point multilink Frame Relay subinterface.
Step 9	**ip address** *ip-address mask* **Example:** `Router(config-subif)# ip address 10.0.1.1 255.255.255.0`	Configures the IP address for the subinterface.
Step 10	**frame-relay interface-dlci** *dlci* **Example:** `Router(config-subif)# frame-relay interface-dlci 100`	Assigns a data-link connection identifier (DLCI) to a Frame Relay subinterface.

Step 11	**end**
	
	Example:
	
	`Router(config-subif)# end`
		Ends the configuration session and returns to privileged EXEC mode.

Step 12	**show frame-relay multilink** **Example:** `Router# show frame-relay multilink`	(Optional) Displays the current Frame Relay multilink configuration.

Configuring a Multilink Frame Relay Bundle Link

To configure a bundle link interface for multilink Frame Relay, perform the steps in this section.

Tip To minimize latency that results from the arrival order of packets, we recommend bundling physical links of the same line speed in one bundle.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface serial** *number*
4. **encapsulation frame-relay mfr** *number* [*name*]
5. **frame-relay multilink output-threshold** *bytes*
6. **frame-relay multilink lid** *name*
7. **frame-relay multilink hello** *seconds*
8. **frame-relay multilink ack** *seconds*
9. **frame-relay multilink retry** *number*
10. **end**
11. **show frame-relay multilink**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface serial** *number* **Example:** `Router(config)# interface serial 5/0`	Selects a physical interface and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**encapsulation frame-relay mfr** *number* [*name*] **Example:** `Router(config-if)# encapsulation frame-relay mfr1`	Creates a multilink Frame Relay bundle link and associates the link with a bundle.
Step 5	**frame-relay multilink output-threshold** *bytes* **Example:** `Router(config-if)# frame-relay multilink output-threshold 500`	(Optional) Configures the number of bytes that a bundle link will transmit before the load-balancing mechanism causes transmission to roll over to the next available link.
Step 6	**frame-relay multilink lid** *name* **Example:** `Router(config-if)# frame-relay multilink lid first-link`	(Optional) Assigns a bundle link identification name with a multilink Frame Relay bundle link. • The bundle link identification (LID) will not go into effect until the interface has gone from the "down" state to the "up" state. One way to bring the interface down and back up again is by using the **shutdown** and **no shutdown** commands in interface configuration mode.
Step 7	**frame-relay multilink hello** *seconds* **Example:** `Router(config-if)# frame-relay multilink hello 9`	(Optional) Configures the interval at which a bundle link will send out hello messages. • The default value is 10 seconds.
Step 8	**frame-relay multilink ack** *seconds* **Example:** `Router(config-if)# frame-relay multilink ack 6`	(Optional) Configures the number of seconds that a bundle link will wait for a hello message acknowledgment before resending the hello message. • The default value is 4 seconds.
Step 9	**frame-relay multilink retry** *number* **Example:** `Router(config-if)# frame-relay multilink retry 3`	(Optional) Configures the maximum number of times that a bundle link will resend a hello message while waiting for an acknowledgment. • The default value is 2 tries.

Step 10	**end**
	
	Example:
	
	`Router(config-if)# end`
		Ends the configuration session and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 11	**show frame-relay multilink** **Example:** `Router# show frame-relay multilink`	(Optional) Displays the current Frame Relay multilink configuration.

Monitoring and Maintaining Multilink Frame Relay FRF.16.1

To monitor and maintain multilink Frame Relay, perform the steps in this section.

SUMMARY STEPS

1. **enable**
2. **debug frame-relay multilink** [**control** [**mfr** *number* | **serial** *number*]]
3. **show frame-relay multilink** [**mfr** *number* | **serial** *number*] [**detailed**]
4. **show interfaces mfr** *number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug frame-relay multilink** [**control** [**mfr** *number* | **serial** *number*]] **Example:** `Router# debug frame-relay multilink control mfr1`	(Optional) Displays debug messages for multilink Frame Relay bundles and bundle links.
Step 3	**show frame-relay multilink** [**mfr** *number* | **serial** *number*] [**detailed**] **Example:** `Router# show frame-relay multilink mfr1 detailed`	(Optional) Displays configuration information and statistics about multilink Frame Relay bundles and bundle links.

	Command or Action	Purpose
Step 4	**show interfaces mfr** *number* **Example:** `Router# show interfaces mfr1`	(Optional) Displays information and packet statistics for the bundle interface.

Examples

The following example shows output for the **show frame-relay multilink** command. Because a particular bundle or bundle link is not specified, information for all bundles and bundle links is displayed:

```
Router# show frame-relay multilink

Bundle: MFR0, state up, class A, no fragmentation
 ID: Bundle-Dallas
 Serial5/1, state up/up, ID: BL-Dallas-1
 Serial5/3, state up/add-sent, ID: BL-Dallas-3
Bundle: MFR1, state down, class B, fragmentation
 ID: Bundle-NewYork#1
 Serial3/0, state up/up, ID: BL-NewYork-1
 Serial3/2, state admin-down/idle, ID: BL-NewYork-2
```

The following example shows output for the **show frame-relay multilink** command when a Frame Relay bundle is configured as bandwidth class C (threshold):

```
Router# show frame-relay multilink

Bundle: MFR0, state down, class C (threshold 3), no fragmentation
 ID: Bundle-Dallas
 Serial5/1, state up/up, ID: BL-Dallas-1
 Serial5/3, state up/add-sent, ID: BL-Dallas-3
```

The following example shows output for the **show frame-relay multilink** command when the **serial** *number* keyword and argument are specified. It displays information about the specified bundle link:

```
Router# show frame-relay multilink serial 3/2
 Bundle links :
 Serial3/2, HW state :down, Protocol state :Down_idle, LID :Serial3/2
 Bundle interface = MFR0,  BID = MFR0
```

The following examples show output for the **show frame-relay multilink** command when the **serial** *number* keyword and argument and the **detailed** option are specified. Detailed information about the specified bundle links is displayed. The first example shows a bundle link in the "idle" state. The second example shows a bundle link in the "up" state:

```
Router# show frame-relay multilink serial 3 detail
Bundle links:
  Serial3, HW state = up, link state = Idle, LID = Serial3
  Bundle interface = MFR0,  BID = MFR0
    Cause code = none, Ack timer = 4, Hello timer = 10,
    Max retry count = 2, Current count = 0,
    Peer LID = Serial5/3, RTT = 0 ms
    Statistics:
    Add_link sent = 0, Add_link rcv'd = 10,
    Add_link ack sent = 0, Add_link ack rcv'd = 0,
    Add_link rej sent = 10, Add_link rej rcv'd = 0,
    Remove_link sent = 0, Remove_link rcv'd = 0,
    Remove_link_ack sent = 0, Remove_link_ack rcv'd = 0,
    Hello sent = 0, Hello rcv'd = 0,
    Hello_ack sent = 0, Hello_ack rcv'd = 0,
    outgoing pak dropped = 0, incoming pak dropped = 0
Router# **show frame-relay multilink serial 3 detail**
Bundle links:
  Serial3, HW state = up, link state = Up, LID = Serial3
  Bundle interface = MFR0,  BID = MFR0
    Cause code = none, Ack timer = 4, Hello timer = 10,
    Max retry count = 2, Current count = 0,
    Peer LID = Serial5/3, RTT = 4 ms
    Statistics:
    Add_link sent = 1, Add_link rcv'd = 20,
    Add_link ack sent = 1, Add_link ack rcv'd = 1,
    Add_link rej sent = 19, Add_link rej rcv'd = 0,
    Remove_link sent = 0, Remove_link rcv'd = 0,
    Remove_link_ack sent = 0, Remove_link_ack rcv'd = 0,
    Hello sent = 0, Hello rcv'd = 1,
    Hello_ack sent = 1, Hello_ack rcv'd = 0,
    outgoing pak dropped = 0, incoming pak dropped = 0
```

Configuration Examples for Multilink Frame Relay FRF.16.1

Configuring Multilink Frame Relay Example

The following example shows the configuration of bundle "MFR1." Serial interfaces 5/0 and 6/0 are configured as bundle links:

```
interface MFR1
 no ip address
 mls qos trust dscp
 frame-relay intf-type dce
 frame-relay multilink bid router1
!
interface MFR1.1 point-to-point
 ip address 10.0.1.1 255.255.255.0
 ip pim sparse-mode
 mls qos trust dscp
 frame-relay interface-dlci 100
interface Serial5/0
 encapsulation frame-relay MFR1
 frame-relay multilink lid first-link
 frame-relay multilink hello 9
 frame-relay multilink retry 3
interface Serial6/0
 encapsulation frame-relay MFR1
 frame-relay multilink ack 4
```

Configuring Variable Bandwidth Class Support Example

The following example configures the Frame Relay bundle "MFR1" to use the class B (all links) criterion to be activated or deactivated:

```
interface MFR1
 ip address 10.1.1.1 255.255.255.0
 frame-relay interface-dlci 100
 frame-relay multilink bandwidth-class b
```

Additional References

Related Documents

Related Topic	Document Title
WAN commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Wide-Area Networking Command Reference*

Standards

Standard	Title
No new or modified standards are supported by this functionality.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this functionality.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for Multilink Frame Relay FRF.16.1

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 6: Feature Information for Multilink Frame Relay (FRF.16.1)

Feature Name	Releases	Feature Information
Multilink Frame Relay (FRF.16.1)	12.0(17)S 12.2(8)T 12.2(14)S12.2(28)SB 12.2(33)SRA 12.2(31)SB2 12.2(33)SXH	The Multilink Frame Relay (FRF.16.1) feature introduces functionality based on the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16.1). In 12.0(17)S, This feature was introduced on the Cisco 12000 series. The following commands were introduced or modified: **debug frame-relay multilink**, **encapsulation frame-relay mfr**, **frame-relay multilink ack**, **frame-relay multilink bandwidth-class**, **frame-relay multilink bid**, **frame-relay multilink hello**, **frame-relay multilink lid**, **frame-relay multilink output-threshold**, **frame-relay multilink retry**, **interface mfr**, **show frame-relay multilink**.
Frame Relay fragmentation (FRF.12)	12.3(9) 12.3(11)T 12.2(30)S	Frame Relay Fragmentation based upon FRF.12 allow long data frames to be fragmented into smaller pieces and interleaved with real-time voice frames or other delay-sensitive traffic. In 12.3(9), this feature was introduced.
Multilink Frame Relay (FRF.16.1) - Variable Bandwidth Class	12.0(30)S 12.4(2)T 15.0(1)S	Multilink Frame Relay (FRF.16.1) variable bandwidth class support allows you to specify the criterion used to activate or deactivate a Frame Relay bundle. In 12.0(30)S, this feature was introduced.

Glossary

BID --Bundle identification. The BID is the name used to identify the bundle. The BID can be assigned, or the default can be used.

BL_ACTIVATE --A message that controls the addition of a bundle link to a Frame Relay bundle.

BL_DEACTIVATE --A message that controls the removal a bundle link from a Frame Relay bundle.

bundle --A logical grouping of one or more physical interfaces using the formats and procedures of multilink Frame Relay. A bundle emulates a physical interface to the Frame Relay data-link layer. The bundle is also referred to as the *MFR interface* .

bundle link --An individual physical interface that is a member of a bundle.

DLCI --data-link connection identifier. A value that identifies a permanent virtual circuit (PVC) in a Frame Relay network.

HELLO message --A message that notifies a peer endpoint that the local endpoint is in the operational state (up).

HELLO_ACK --A message that notifies a peer endpoint that a hello message has been received.

LID --link identification. The LID is the name used to identify a bundle link. The LID can be assigned, or the default can be used.

LMI --Local Management Interface. A set of enhancements to the basic Frame Relay specification. LMI includes support for a keepalive mechanism, which verifies that data is flowing; a multicast mechanism, which provides the network server with its local DLCI and the multicast DLCI; global addressing, which gives DLCIs global rather than local significance in Frame Relay networks; and a status mechanism, which provides an ongoing status report on the DLCIs known to the switch.

NNI --Network-to-Network Interface. The interface between two Frame Relay devices that are both located in a private network or both located in a public network.

PH_ACTIVATE --A message that indicates that the Frame Relay bundle is up.

PH_DEACTIVATE --A message that indicates that the Frame Relay bundle is down.

UNI --User-to-Network Interface. The interface between a Frame Relay device in a public network and a Frame Relay device in a private network.

CHAPTER 10

Distributed Multilink Frame Relay FRF.16

Feature History

Release	Modification
12.0(24)S	The Distributed Multilink Frame Relay feature was introduced. This feature introduced Multilink Frame Relay (FRF.16) on VIP-enabled Cisco 7500 series routers.
12.3(4)T	This feature on VIP-enabled Cisco 7500 series routers was integrated into Cisco IOS Release 12.3(4)T.

This document describes the Distributed Multilink Frame Relay (dMFR) feature in Cisco IOS Release 12.0(24)S and Cisco IOS Release 12.3(4)T. The dMFR feature introduces MFR on VIP-enabled Cisco 7500 series routers. For information on MFR on other platforms, see the *Multilink Frame Relay (FRF.16)* document.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Feature Overview

The Distributed Multilink Frame Relay feature introduces functionality based on the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16) to VIP-enabled Cisco 7500 series routers. The Distributed Multilink Frame Relay feature provides a cost-effective way to increase bandwidth for particular applications by enabling multiple serial links to be aggregated into a single bundle of bandwidth. Multilink Frame Relay is supported on User-to-Network Interfaces (UNI) and Network-to-Network Interfaces (NNI) in Frame Relay networks.

Multilink Frame Relay Bundles and Bundle Links

The Multilink Frame Relay feature enables you to create a virtual interface called a *bundle* or *bundle interface* . The bundle interface emulates a physical interface for the transport of frames. The Frame Relay data link runs on the bundle interface, and Frame Relay virtual circuits are built upon it.

The bundle is made up of multiple serial links, called *bundle links* . Each bundle link within a bundle corresponds to a physical interface. Bundle links are invisible to the Frame Relay data-link layer, so Frame Relay functionality cannot be configured on these interfaces. Regular Frame Relay functionality that you want to apply to these links must be configured on the bundle interface. Bundle links are visible to peer devices. The local router and peer devices exchange link integrity protocol control messages to determine which bundle links are operational and to synchronize which bundle links should be associated with which bundles.

Link Integrity Protocol Control Messages

For link management, each end of a bundle link follows the MFR Link Integrity Protocol and exchanges link control messages with its peer (the other end of the bundle link). To bring up a bundle link, both ends of the link must complete an exchange of ADD_LINK and ADD_LINK_ACK messages. To maintain the link, both ends periodically exchange HELLO and HELLO_ACK messages. This exchange of hello messages and acknowledgments serve as a keepalive mechanism for the link. If a router is sending hello messages but not receiving acknowledgments, it will resend the hello message up to a configured maximum number of times. If the router exhausts the maximum number of retries, the bundle link line protocol is considered down (unoperational).

The bundle link interface's line protocol status is considered up (operational) when the peer device acknowledges that it will use the same link for the bundle. The line protocol remains up when the peer device acknowledges the hello messages from the local router.

The bundle interface's line status becomes up when at least one bundle link has its line protocol status up. The bundle interface's line status goes down when the last bundle link is no longer in the up state. This behavior complies with the class A bandwidth requirement defined in FRF.16.

The bundle interface's line protocol status is considered up when the Frame Relay data-link layer at the local router and peer device synchronize using the Local Management Interface (LMI), when LMI is enabled. The bundle line protocol remains up as long as the LMI keepalives are successful.

Load Balancing

Distributed Multilink Frame Relay provides load balancing across the bundle links within a bundle. If a bundle link chosen for transmission happens to be busy transmitting a long packet, the load balancing mechanism can try another link, thus solving the problems seen when delay-sensitive packets have to wait.

Benefits

Flexible Pool of Bandwidth

By combining multiple physical interfaces into a bundle, you can design a Frame Relay interface with more bandwidth than is available from any single physical interface. For example, many new network applications require more bandwidth than is available on a T1 line. One option is to invest in a T3 line; however, T3 lines can be expensive and are not available in some locations. Distributed Multilink Frame Relay provides a cost-effective solution to this problem by allowing multiple T1 lines to be aggregated into a single bundle of bandwidth.

Greater Service Resilience When Links Fail

Greater service resilience is provided when multiple physical interfaces are provisioned as a single bundle. When a link fails, the bundle continues to support the Frame Relay service by transmitting across the remaining bundle links.

Restrictions

The Distributed Multilink Frame Relay feature has the following restrictions:

- ISDN interfaces and any type of virtual interfaces cannot be a bundle link.
- Distributed CEF is limited to IP traffic only; all other protocols are processed using the Route Switch Processor (RSP).
- Frame Relay fragmentation (FRF.12) is not supported.
- Multilink Frame Relay MIB (RFC 3020) is not supported.
- FRF.9 hardware compression over multilink Frame Relay is not supported.
- Each link in a bundle must reside on the same port adapter and all links in a bundle must have identical configurations. The same bandwidth for each link in the bundle is also recommended because bundles that contain individual links with different bandwidths process packets less efficiently.
- Fragmentation is not supported on the transmitting interface when used in conjunction with Distributed Multilink Frame Relay.

- The maximum differential delay is 50 ms.
- All T1 lines can be combined into one bundle. A VIP2-50 with 4 or 8 MB of SRAM supports up to 16 T1 bundles per VIP and a VIP2-50 with 2 MB of SRAM supports up to 8 T1 bundles per VIP. A maximum of 40 T1 bundles per VIP can be used on a VIP4-80.
- All E1 lines can be combined into one bundle. A VIP2-50 with 4 or 8 MB of SRAM supports up to 12 E1 bundles per VIP and a VIP2-50 with 2 MB or SRAM supports up to 8 E1 bundles per VIP. A maximum of 32 E1 bundles per VIP can be used on a VIP4-80.

Related Documents

- Cisco IOS Wide-Area Networking Configuration Guide , Release 12.3
- Cisco IOS Wide-Area Networking Command Reference , Release 12.3
- Multilink Frame Relay (FRF.16) (provides information on nondistributed Multilink Frame Relay)
- Cisco IOS Release 12.3 T command references (information on 12.0 S and 12.3 T commands)

Supported Platforms

- Cisco 7500 series router with a VIP2-50 or greater

This feature works on the following port adapters:

-
    - PA-MC-T3
    - PA-MC-2T3+
    - PA-MC-E3
    - PA-MC-2E1
    - PA-MC-2T1
    - PA-MC-4T1
    - PA-MC-8T1
    - PA-MC-8E1
    - PA-MC-STM-1
    - PA-MC-8TE1+
    - PA-4T+
    - PA-8T

Determining Platform Support Through Cisco Feature Navigator

Cisco IOS software is packaged in feature sets that are supported on specific platforms. To get updated information regarding platform support for this feature, access Cisco Feature Navigator. Cisco Feature Navigator dynamically updates the list of supported platforms as new platform support is added for the feature.

Cisco Feature Navigator is a web-based tool that enables you to determine which Cisco IOS software images support a specific set of features and which features are supported in a specific Cisco IOS image. You can search by feature or release. Under the release section, you can compare releases side by side to display both the features unique to each software release and the features in common.

To access Cisco Feature Navigator, you must have an account on Cisco.com. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you. Qualified users can establish an account on Cisco.com by following the directions found at this URL:

http://www.cisco.com/register http://www.cisco.com/register

Cisco Feature Navigator is updated regularly when major Cisco IOS software releases and technology releases occur. For the most current information, go to the Cisco Feature Navigator home page at the following URL:

http://www.cisco.com/go/fn

Availability of Cisco IOS Software Images

Platform support for particular Cisco IOS software releases is dependent on the availability of the software images for those platforms. Software images for some platforms may be deferred, delayed, or changed without prior notice. For updated information about platform support and availability of software images for each Cisco IOS software release, refer to the online release notes or, if supported, Cisco Feature Navigator.

Supported Standards MIBs and RFCs

Standards

Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16.1), July 2001

MIBs

No new or modified MIBs are supported by this feature.

To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL:

http://tools.cisco.com/ITDIT/MIBS/servlet/index

If Cisco MIB Locator does not support the MIB information that you need, you can also obtain a list of supported MIBs and download MIBs from the Cisco MIBs page at the following URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

To access Cisco MIB Locator, you must have an account on Cisco.com. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you. Qualified users can establish an account on Cisco.com by following the directions found at this URL:

http://www.cisco.com/register

RFCs

No new or modified RFCs are supported by this feature.

Prerequisites

- Distributed Cisco Express Forwarding (CEF) must be enabled globally.
- Multilink Frame Relay must be configured on the peer device.
- The multilink Frame Relay peer device must not send frames that require assembly.

Configuration Tasks

Configuring a Multilink Frame Relay Bundle

To configure the bundle interface for Distributed Multilink Frame Relay, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **interface mfr** *number*
2. Router(config-if)# **frame-relay multilink bid** *name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface mfr** *number* **Example:**	Configures a multilink Frame Relay bundle interface.
Step 2	Router(config-if)# **frame-relay multilink bid** *name* **Example:**	(Optional) Assigns a bundle identification name to a multilink Frame Relay bundle. **Note** The bundle identification (BID) will not go into effect until the interface has gone from the down state to the up state. One way to bring the interface down and back up again is by using the **shut** and **no shut** commands in interface configuration mode.

Configuring a Multilink Frame Relay Bundle Link

To configure a bundle link interface for multilink Frame Relay, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **interface serial** *number*
2. Router(config-if)# **encapsulation frame-relay mfr** *number*[*name*]
3. Router(config-if)# **frame-relay multilink lid** *name*
4. Router(config-if)# **frame-relay multilink hello** *seconds*
5. Router(config-if)# **frame-relay multilink ack** *seconds*
6. Router(config-if)# **frame-relay multilink retry** *number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface serial** *number* **Example:**	Selects a physical interface and enters interface configuration mode.
Step 2	Router(config-if)# **encapsulation frame-relay mfr** *number*[*name*] **Example:**	Creates a multilink Frame Relay bundle link and associates the link with a bundle. **Tip** To minimize latency that results from the arrival order of packets, we recommend bundling physical links of the same line speed in one bundle.
Step 3	Router(config-if)# **frame-relay multilink lid** *name* **Example:**	(Optional) Assigns a bundle link identification name to a multilink Frame Relay bundle link. **Note** The bundle link identification (LID) will not go into effect until the interface has gone from the down state to the up state. One way to bring the interface down and back up again is by using the **shut** and **no shut** commands in interface configuration mode.
Step 4	Router(config-if)# **frame-relay multilink hello** *seconds* **Example:**	(Optional) Configures the interval at which a bundle link will send out hello messages. The default value is 10 seconds.
Step 5	Router(config-if)# **frame-relay multilink ack** *seconds* **Example:**	(Optional) Configures the number of seconds that a bundle link will wait for a hello message acknowledgment before resending the hello message. The default value is 4 seconds.
Step 6	Router(config-if)# **frame-relay multilink retry** *number* **Example:**	(Optional) Configures the maximum number of times a bundle link will resend a hello message while waiting for an acknowledgment. The default value is 2 tries.

Verifying Multilink Frame Relay

To verify multilink Frame Relay configuration, use the **show frame-relay multilink** command.

The following example shows output for the **show frame-relay multilink** command. Because a particular bundle or bundle link is not specified, information for all bundles and bundle links is displayed.

```
Router# show frame-relay multilink

Bundle: MFR0, state up, class A, no fragmentation
 ID: Bundle-Dallas
 Serial5/1, state up/up, ID: BL-Dallas-1
 Serial5/3, state up/add-sent, ID: BL-Dallas-3
Bundle: MFR1, state down, class B, fragmentation
 ID: Bundle-NewYork#1
 Serial3/0, state up/up, ID: BL-NewYork-1
 Serial3/2, state admin-down/idle, ID: BL-NewYork-2
```

The following example shows output for the **show frame-relay multilink** command with the **serial** *number* option. It displays information about the specified bundle link.

```
Router# show frame-relay multilink serial3/2
 Bundle links :
 Serial3/2, HW state :Administratively down, Protocol state :Down_idle, LID :Serial3/2
 Bundle interface = MFR0,  BID = MFR0
```

The following examples show output for the **show frame-relay multilink** command with the **serial** *number* and **detail** options. Detailed information about the specified bundle links is displayed. The first example shows a bundle link in the "idle" state. The second example shows a bundle link in the "up" state.

```
Router# show frame-relay multilink serial3 detail
 Bundle links:
  Serial3, HW state = up, link state = Idle, LID = Serial3
  Bundle interface = MFR0,  BID = MFR0
    Cause code = none, Ack timer = 4, Hello timer = 10,
    Max retry count = 2, Current count = 0,
    Peer LID = Serial5/3, RTT = 0 ms
    Statistics:
    Add_link sent = 0, Add_link rcv'd = 10,
    Add_link ack sent = 0, Add_link ack rcv'd = 0,
    Add_link rej sent = 10, Add_link rej rcv'd = 0,
    Remove_link sent = 0, Remove_link rcv'd = 0,
    Remove_link_ack sent = 0, Remove_link_ack rcv'd = 0,
    Hello sent = 0, Hello rcv'd = 0,
    Hello_ack sent = 0, Hello_ack rcv'd = 0,
    outgoing pak dropped = 0, incoming pak dropped = 0
Router# show frame-relay multilink serial3 detail
 Bundle links:
  Serial3, HW state = up, link state = Up, LID = Serial3
  Bundle interface = MFR0,  BID = MFR0
    Cause code = none, Ack timer = 4, Hello timer = 10,
    Max retry count = 2, Current count = 0,
    Peer LID = Serial5/3, RTT = 4 ms
    Statistics:
    Add_link sent = 1, Add_link rcv'd = 20,
    Add_link ack sent = 1, Add_link ack rcv'd = 1,
    Add_link rej sent = 19, Add_link rej rcv'd = 0,
    Remove_link sent = 0, Remove_link rcv'd = 0,
    Remove_link_ack sent = 0, Remove_link_ack rcv'd = 0,
    Hello sent = 0, Hello rcv'd = 1,
    Hello_ack sent = 1, Hello_ack rcv'd = 0,
    outgoing pak dropped = 0, incoming pak dropped = 0
```

Monitoring and Maintaining Distributed Multilink Frame Relay

To monitor and maintain Distributed Multilink Frame Relay, use one or more of the following commands in privileged EXEC mode:

Command	Purpose
`Router#` **debug frame-relay multilink** [**control** [**mfr** *number* | **serial** *number*]]	Displays debug messages for multilink Frame Relay bundles and bundle links.
`Router#` **show frame-relay multilink** [**mfr** *number* | **serial** *number*] [**detailed**]	Displays configuration information and statistics about multilink Frame Relay bundles and bundle links.
`Router#` **show interfaces mfr** *number*	Displays information and packet statistics for the bundle interface.

Configuration Examples

Distributed Multilink Frame Relay Configuration Example

The following example shows the configuration of bundle "MFR1". Serial interfaces 5/0 and 6/0 are configured as bundle links.

```
interface MFR1
 frame-relay multilink bid first-bundle
 frame-relay traffic-shaping
 frame-relay class ocean
interface MFR1.1 point-to-point
 ip address 1.1.1.1 255.255.255.0
 frame-relay interface-dlci 100
interface Serial5/0
 encapsulation frame-relay MFR1
 frame-relay multilink lid first-link
 frame-relay multilink hello 9
 frame-relay multilink retry 3
interface Serial6/0
 encapsulation frame-relay MFR1
 frame-relay multilink ack 4
```

Glossary

BID --bundle identification. BID is the name used to identify the bundle. The BID can be assigned or the default can be used.

bundle --A logical grouping of one or more physical interfaces using the formats and procedures of multilink Frame Relay. A bundle emulates a physical interface to the Frame Relay data-link layer. The bundle is also referred to as the *mfr interface* .

bundle link --An individual physical interface that is a member of a bundle.

DLCI --data-link connection identifier. Value that identifies a permanent virtual circuit (PVC) in Frame Relay network.

HELLO message --Message that notifies a peer endpoint that the local endpoint is in the operational state (up).

HELLO_ACK --Message that notifies a peer endpoint that a hello message has been received.

LID --link identification. LID is the name used to identify a bundle link. The LID can be assigned or the default can be used.

LMI --Local Management Interface. Set of enhancements to the basic Frame Relay specification. LMI includes support for a keepalive mechanism, which verifies that data is flowing; a multicast mechanism, which provides the network server with its local DLCI and the multicast DLCI; global addressing, which gives DLCIs global rather than local significance in Frame Relay networks; and a status mechanism, which provides an ongoing status report on the DLCIs known to the switch.

NNI --Network-to-Network Interface. The interface between two Frame Relay devices that are both located in a private network or both located in a public network.

UNI --User-to-Network Interface. The interface between a Frame Relay device in a public network and a Frame Relay device in a private network.