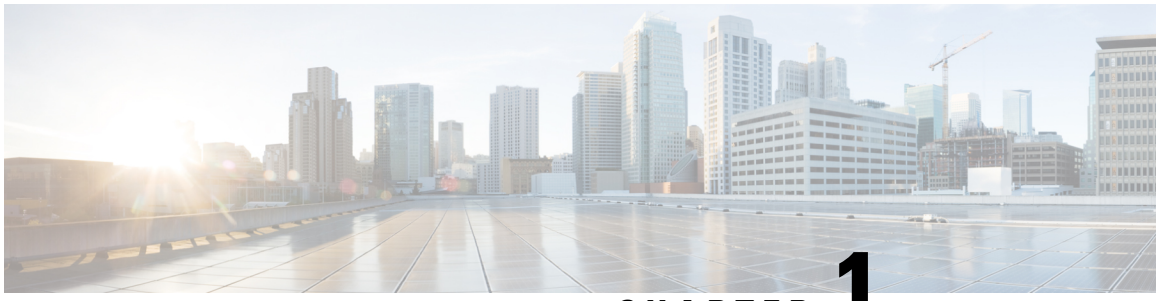
Wide-Area Networking Configuration Guide: Frame Relay, Cisco IOS XE Gibraltar 16.10.x

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

Read Me First

Important Information about Cisco IOS XE 16

Effective Cisco IOS XE Release 3.7.0E (for Catalyst Switching) and Cisco IOS XE Release 3.17S (for Access and Edge Routing) the two releases evolve (merge) into a single version of converged release—the Cisco IOS XE 16—providing one release covering the extensive range of access and edge products in the Switching and Routing portfolio.

Feature Information

Use Cisco Feature Navigator to find information about feature support, platform support, and Cisco software image support. An account on Cisco.com is not required.

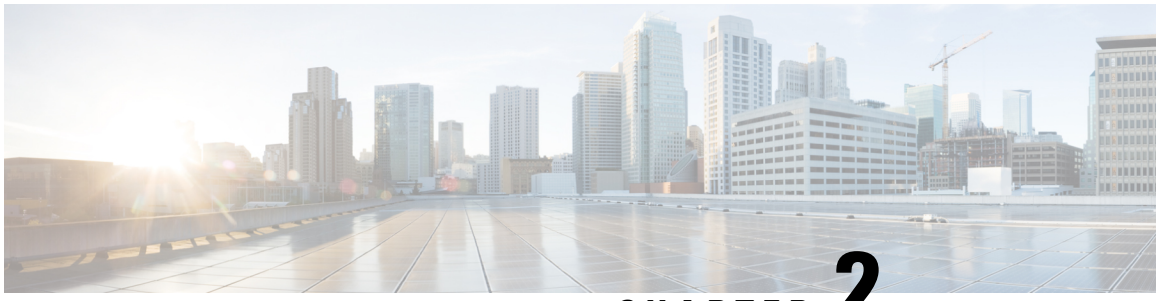
Related References

- Cisco IOS Command References, All Releases

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see What's New in Cisco Product Documentation.

To receive new and revised Cisco technical content directly to your desktop, you can subscribe to the What's New in Cisco Product Documentation RSS feed. RSS feeds are a free service.



CHAPTER 2

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.

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.

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.

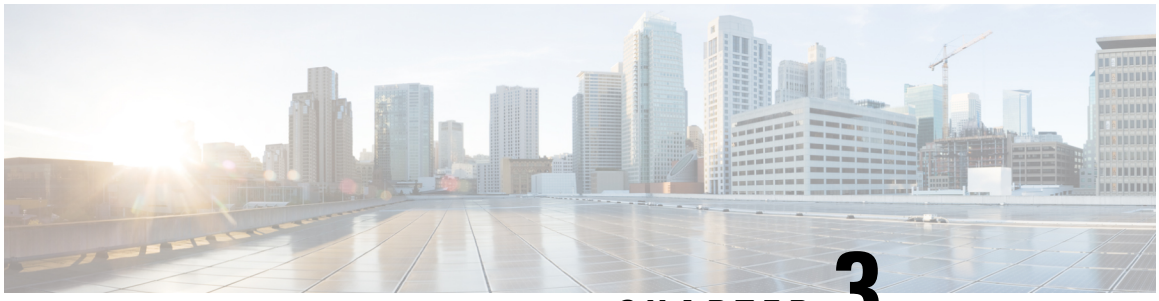
The L2VPN Pseudowire Redundancy feature enables you to set up backup pseudowires. You can configure the network with redundant pseudowires (PWs) and redundant network elements.

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.



CHAPTER 3

Configuring Frame Relay

Frame Relay is a high-performance Wide Area Network (WAN) protocol that operates at the physical and data link layers. The Cisco IOS XE Frame Relay implementation currently supports routing for IPv4, IPv6, and Multiprotocol Label Switching (MPLS).

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.

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.

Restrictions for Configuring Frame Relay

Cisco IOS XE software does not support the following:

- Multipoint permanent virtual circuits (PVCs)
- Switched virtual circuits (SVCs)
- Frame relay switching
- 4-byte extended addresses
- End-to-end keepalives
- FRF.9 payload compression

- Data stream compression
- Packet by packet encapsulation payload compression
- Multi-point frame-relay
- Legacy frame-relay traffic shaping (Cisco IOS XE software supports only policy map-based MQC.)
- MQC based frame relay traffic shaping is not supported on frame relay main interface.
- Function "set fr-de" for HQos configuration

Information About Frame Relay

Frame Relay Hardware Configurations

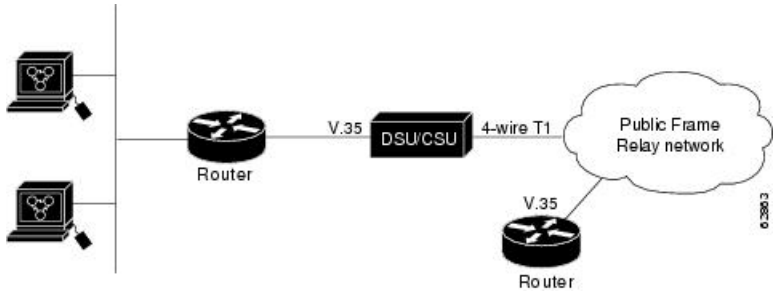
You can create Frame Relay connections using one of the following hardware configurations:

- Devices and access servers connected directly to the Frame Relay switch
- Devices 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 Devices can connect to Frame Relay networks either by direct connection to a Frame Relay switch, through a direct connection to a Point of sale (POS) interface or a T1/T3 interface, or through CSU/DSUs. However, a single device 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 need not configure the LMI explicitly.

LMI autosense is active in the following situations:

- The device 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 Data Terminal Equipment (DTE).
- The LMI type is not explicitly configured.

Activating LMI Autosense

Status Request

When Local Management Interface (LMI) autosense is active, it sends out a full status request in all three LMI types to the switch. The order which is implemented in rapid succession is as follows:

- ANSI
- ITU
- Cisco

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.

MQC-Based Frame Relay Traffic Shaping

Legacy frame-relay traffic shaping is not supported. Cisco IOS XE software only supports policy map based MQC.

Traffic-Shaping Map Class for the Interface

If you specify a Frame Relay map class for a main interface, all the virtual circuits (VCs) on its subinterfaces inherit all the traffic-shaping parameters defined for the class. You can override the default for a specific data link connection identifier (DLCI) on a specific subinterface by using the **class** VC configuration command to assign the DLCI explicitly to a different class. For information about setting up subinterfaces, refer the section Configuring Frame Relay Subinterfaces, on page 25 .

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.

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 communicate with station B, and station B can communicate to station C, then station A should be able to communicate 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 are not 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 virtual circuits (VCs) .

Configuring Frame Relay subinterfaces ensures that a single physical interface is considered as multiple virtual interfaces. Hence, packets received on one virtual interface can be forwarded to another virtual interface even if they are configured on the same physical interface.

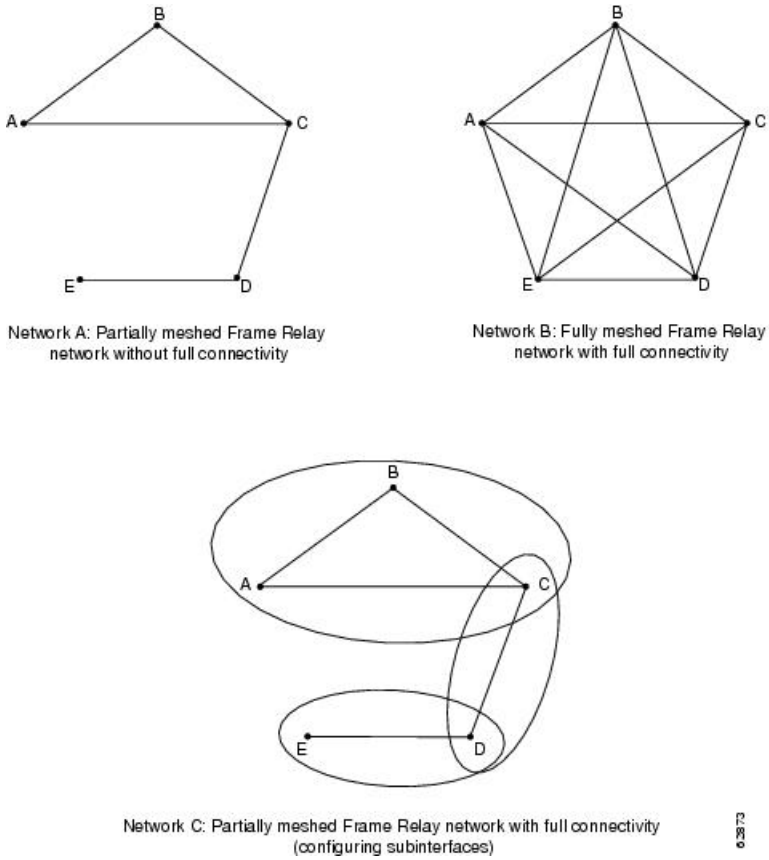
Subinterfaces address the limitations of Frame Relay networks by providing an option to subdivide a partially meshed Frame Relay network into a number of smaller, fully meshed (or point-to-point) subnetworks. Each

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, thus reducing the addressing burden that might otherwise result.)

Note Cisco IOS XE software supports configuration of point-to-point subinterfaces.

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 work with certain protocols (for example, IP). However, this network does not work with other protocols (for example, AppleTalk), because nodes C and D do not relay the packet out at the same interface on which it was received. To make this network fully functional, we need to created a fully meshed network (network B). However, a fully meshed network requires a large number of permanent virtual circuits (PVCs), which may not be economically feasible.

Figure 2: Using Subinterfaces to Provide Full Connectivity on a Partially Meshed Frame Relay Network



By using subinterfaces, you can divide the Frame Relay network into 3 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 2 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

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

Note The **frame-relay interface-dlci** command is typically used on subinterfaces; however, it can also be applied to main interfaces. The 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 permanent virtual circuit (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 device or access server. Instead, you can simply avoid using that subinterface number and use a different subinterface number.

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 Address Resolution Protocol (ARP) is a method of building dynamic address mappings in Frame Relay networks that run DECnet, IP, and Novell IPX. Inverse ARP allows the device or access server to discover the protocol address of a device associated with the virtual circuit (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 data link connection identifier (DLCI) (see the section Configuring Static Address Mapping, on page 20 for more information).

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.

Frame Relay Fragmentation

End-to-End FRF.12 Fragmentation

The purpose of end-to-end Frame Relay Fragmentation 12 (FRF.12) is to support real-time and non-real-time data packets on lower-speed links without causing excessive delay to the real-time data transmission. 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 for fragmentation.

Note When Frame Relay fragmentation is configured, Weighted Fair Queuing (WFQ) or Low Latency Queuing (LLQ) is mandatory. If a map class is configured for Frame Relay fragmentation and the queuing 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 XE Quality of Service Solutions Configuration Guide* .

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.

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 an 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, as described in the *Configuring an Individual IP Map for TCP IP Header Compression* section.

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. It provides 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,*A Transport Protocol for Real-Time Applications* . RTP is not intended for data traffic, which uses TCP or UDP.

For configuration tasks and examples of RTP header compression using Frame Relay encapsulation, see the *Cisco IOS XE IP Multicast Configuration Guide* .

The commands for configuring this feature are available in the *Cisco IOS IP Multicast Command Reference.*

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 requirements, use the following command in interface configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *typenumber*
4. **frame-relay map** *protocol protocol-address dlci* [**broadcast**] [**ietf**] [**cisco**]
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	**frame-relay map** *protocol protocol-address dlci* [**broadcast**] [**ietf**] [**cisco**] **Example:** `Device(config-if)#`	Enables and specifies the Frame Relay encapsulation method.
Step 5	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

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:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-if)# end`	
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 Local Management Interface (LMI). By default, this interval is 10 seconds. According to the LMI protocol, the keepalive interval must be less than the corresponding interval on the switch. To set the keepalive interval, use the following command in interface configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *typenumber*
4. **keepalive** *keepalive period*
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	**keepalive** *keepalive period* **Example:** `Device(config-if)# keepalive 300`	Sets the keepalive interval. • *keepalive period*- Valid range is from 0 to 32767. **Note** To disable keepalives on networks that do not utilize LMI, use the **no keepalive** command.

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

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.

Configuring MQC-Based Frame Relay Traffic Shaping

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. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class class-default**
5. **bandwidth** {*bandwidth-in-kbps* | **remaining** | **percent** }
6. **priority** [*bandwidth-in-kbps* | **level** | **percent**]
7. **shape average** {*rate-in-bps* |**percent**}
8. **shape adaptive** *rate-in-bps*

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	**policy-map** *policy-map-name* **Example:** `Device(config)# policy-map testmap`	Specifies a policy map to define and enters policy map configuration mode.
Step 4	**class class-default** **Example:** `Device(config-pmap)# class class-default`	Specifies a system default class and enters policy-map class configuration .
Step 5	**bandwidth** {*bandwidth-in-kbps*	**remaining**	**percent** } **Example:** `Device(config-pmap-c)#  bandwidth 50`	Configures a minimum bandwidth guarantee for a class.
Step 6	**priority** [*bandwidth-in-kbps*	**level**	**percent**] **Example:** `Device(config-pmap-c)# priority 150`	Assigns priority to a class of traffic belonging to a policy map.
Step 7	**shape average** {*rate-in-bps* |**percent**} **Example:** `Device(config-pmap-c)# shape average 8000`	Shapes traffic to the indicated bit rate according to the algorithm specified.
Step 8	**shape adaptive** *rate-in-bps* **Example:** `Device(config-pmap-c)# shape adaptive 9000`	Shapes traffic to the indicated bit rate according to the algorithm specified.

Customizing Frame Relay for Your Network

Configuring Frame Relay Subinterfaces

Configuring Subinterfaces

Note Multipoint DLCI configurations are currently not supported. Cisco IOS XE software supports point-to-point connections.

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. • Cisco IOS XE software only supports point-to-point subinterfaces.
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.

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.
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.

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 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 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 Traffic Shaping

Example Configuring Class-Based Weighted Fair Queueing

The following example provides a sample configuration for Class-Based Weighted Fair Queueing (CBWFQ) with FRTS:

```
class-map voice
 match ip dscp ef
policy-map llq
 class voice
  priority 32
policy-map shape-policy-map
 class class-default
  shape average 64000
  shape adaptive 32000
  service-policy llq
map-class frame-relay shape-map-class

service-policy output shape-policy-map
interface serial 0/0
encapsulation frame-relay
interface serial 0/0.1 point-to-point
ip address 192.168.1.1 255.255.255.0
frame-relay interface-dlci 100
class shape-map-class
```

Example Configuring Class-Based Weighted Fair Queueing with Fragmentation

The following example provides a sample configuration for CBWFQ and fragmentation with FRTS. This configuration example is exactly the same as the example shown in the Example Configuring Class-Based Weighted Fair Queueing section, with the addition of the **frame-relay fragment** command to configure fragmentation.

```
class-map voice
 match ip dscp ef
policy-map llq
 class voice
  priority 32
policy-map shape-policy-map
 class class-default
  shape average 64000
  shape adaptive 32000
  service-policy llq
map-class frame-relay shape-map-class
 frame-relay fragment 80
 service-policy output shape-policy-map
interface serial 0/0
 encapsulation frame-relay
interface serial 0/0.1 point-to-point
 ip address 192.168.1.1 255.255.255.0
```

```
 frame-relay interface-dlci 100
  class shape-map-class
```

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 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 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 XE Wide-Area Networking configuration tasks	*Cisco IOS XE Wide-Area Networking Configuration Guide*, Release 2
Wide-Area networking commands	*Cisco IOS Wide-Area Networking Command Reference*

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

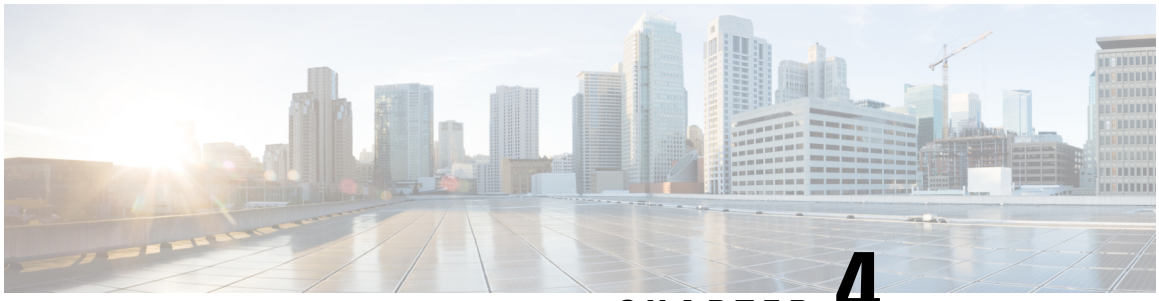
Feature Information for Configuring Frame Relay

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 1: Feature Information for Configuring Frame Relay

Feature Name	Releases	Feature Information
Frame Relay	Cisco IOS XE Release 2.1	Frame Relay is a high-performance WAN protocol that operates at the physical and data link layers.
Frame Relay Encapsulation	Cisco IOS XE Release 2.1	Frame Relay supports encapsulation of all supported protocols in conformance with RFC 1490, allowing interoperability between multiple vendors.
Frame Relay Fragmentation (FRF.12)	Cisco IOS XE Release 2.1	End-to-end FRF.12 fragmentation supports 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.
Local Management Interface	Cisco IOS XE Release 2.1	Local Management Interface (LMI) autosense enables an interface to determine the LMI type supported by a switch. With the support for LMI autosense, you do not need to configure the LMI explicitly.



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.

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.

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

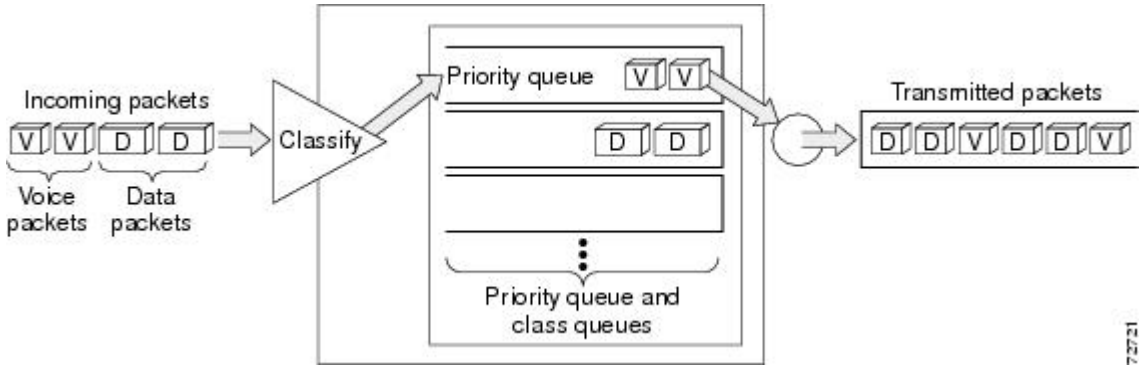
The Frame Relay Queueing and Fragmentation at the Interface feature simplifies the configuration of low-latency, low-jitter quality of service (QoS) by enabling the queueing policy and fragmentation configured on the main interface to apply to all permanent virtual circuits (PVCs) and subinterfaces under 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 3: 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.
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:**	(Optional) Shapes traffic to the indicated bit rate according to the algorithm specified.

	Command or Action	Purpose
	`Router(config-pmap-c)# shape peak 10`	
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. Router(config)# **interface** *type number*
2. Router(config-if)# **encapsulation frame-relay**
3. Router(config-if)# **frame-relay interface-dlci** *dlci*
4. Router(config-if-dlci)# **class** *name*
5. Router(config-if-dlci)# **exit**
6. Router(config)# **map-class frame-relay** *name*
7. Router(config-map-class)# **frame-relay fragment** *fragment-size* **end-to-end**
8. Router(config-map-class)# **no frame-relay adaptive-shaping**
9. Router(config-map-class)# **service-policy output** *policy-map-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number*	Configures an interface type and enters interface configuration mode.
Step 2	Router(config-if)# **encapsulation frame-relay**	Enables Frame Relay encapsulation.
Step 3	Router(config-if)# **frame-relay interface-dlci** *dlci*	Assigns a DLCI to a specified Frame Relay subinterface on the router.
Step 4	Router(config-if-dlci)# **class** *name*	Associates a map class with a specified DLCI.
Step 5	Router(config-if-dlci)# **exit**	Exits configuration mode.
Step 6	Router(config)# **map-class frame-relay** *name*	Specifies a map class to define QoS values for a Frame Relay SVC or PVC.

	Command or Action	Purpose
Step 7	Router(config-map-class)# **frame-relay fragment** *fragment-size* **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 8	Router(config-map-class)# **no frame-relay adaptive-shaping**	Disables Frame Relay adaptive traffic shaping.
Step 9	Router(config-map-class)# **service-policy output** *policy-map-name*	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.

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:

Command	Purpose
Router# **debug frame-relay fragment** [**event** | **interface** *type number dlci*]	Displays information related to Frame Relay fragmentation on a PVC.

Command	Purpose
`Router# show frame-relay fragment [interface type number [dlci]]`	Displays information about Frame Relay fragmentation.
`Router# show interfaces serial number`	Displays information about a serial interface.
`Router# show queue interface-type interface-number`	Displays the contents of packets inside a queue for a particular interface.
`Router# 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
Frame Relay configuration	*Cisco IOS Wide-Area Networking Configuration Guide , Release 12.4T*
Frame Relay commands	*Cisco IOS Wide-Area Networking Command Reference, Release 12.4T*

Standards

Standard	Title
FRF.16.1	*Multilink Frame Relay UNI/NNI Implementation Agreement,* May 2002

MIBs

MIB	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

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

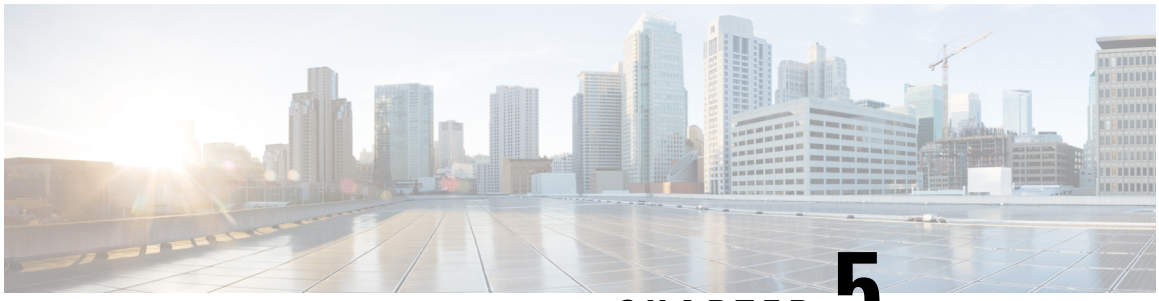
Feature Information for Frame Relay Queueing and Fragmentation at the Interface

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 Queueing and Fragmentation at the Interface	Cisco IOS XE Release 2.1	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.



CHAPTER 5

Frame Relay MIB Enhancements

The Cisco Frame Relay MIB describes managed objects that enable users to remotely monitor Frame Relay operations using Simple Network Management Protocol (SNMP). Frame Relay fragmentation is supported in the MIB.

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.

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 MIB Enhancements

The tasks in this document assume that you have configured Frame Relay and SNMP on your devices.

To access the information introduced by the Frame Relay MIB enhancements, you must have the Cisco Frame Relay MIB in the MIB file called CISCO-FRAME-RELAY-MIB.my compiled in your network management system (NMS) application. You can find this MIB on the Web at Cisco's MIB website at

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

Restrictions for Frame Relay MIB Enhancements

- Frame Relay-ATM Network Interworking (FRF.5)
- Frame Relay-ATM Service Interworking (FRF.8)
- Frame Relay switching

Information About Frame Relay MIB Enhancements

Feature Overview

The Cisco Frame Relay MIB describes managed objects that enable users to remotely monitor Frame Relay operations using SNMP. The Frame Relay MIB Enhancements feature extends the Cisco Frame Relay MIB by adding MIB objects to monitor the following Frame Relay functionality:

- Frame Relay fragmentation
- Input and output rates of individual virtual circuits (VCs)

The table below describes the MIB tables and objects that are introduced by the Frame Relay MIB enhancements. For a complete description of the MIB, see the Cisco Frame Relay MIB file CISCO-FRAME-RELAY-MIB.my, available through Cisco.com at the following URL:

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

Table 3: MIB Tables and Objects Introduced by the Frame Relay MIB Enhancements

Table or Object	Description
cfrFragTable	Table of Frame Relay fragmentation information.
cfrFRF5ConnectionTable	Table of Frame Relay-ATM Network Interworking connection information.
cfrFRF8ConnectionTable	Table of Frame Relay-ATM Service Interworking connection information.
cfrSwitchingTable	Table of Frame Relay switching entries.
cfrExtCircuitTxDataRate	Average rate, in bytes per second, at which data is transmitted on a circuit.
cfrExtCircuitTxPktRate	Average number of packets sent per second on a circuit.
cfrExtCircuitRcvDataRate	Average rate, in bytes per second, at which data is received on a circuit.
cfrExtCircuitRcvPktRate	Average number of packets received per second on a circuit.

The Frame Relay MIB Enhancements feature also modifies the **load-interval** command to enable you to configure the load interval per permanent virtual circuit (PVC). The load interval is the length of time for which data is used to compute load statistics, including input rate in bits and packets per second, output rate in bits and packets per second, load, and reliability. Before the introduction of this feature, the load interval could be configured only for the interface.

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.

How to Configure Frame Relay MIB Enhancements

Setting the Load Interval for a PVC

You can change the period of time over which a set of data is used for computing load statistics. Decisions, such as for dial backup, depend on these statistics. If you decrease the load interval, the average statistics are computed over a shorter period of time and are more responsive to bursts of traffic.

To change the length of time for which a set of data is used to compute load statistics for a 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)# **load-interval** *seconds*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config-if)# **frame-relay interface-dlci** *dlci*	Assigns a specific PVC to a DLCI[1], and enters Frame Relay DLCI configuration mode.
Step 2	router(config-fr-dlci)# **load-interval** *seconds*	Changes the length of time for which data is used to compute load statistics. The seconds argument must be a multiple of 30. The range is from 30 to 300 seconds. The default is 300 seconds.

Verifying the Load Interval

Use the **show running-config** command to verify that you have configured the load interval correctly.

Configuration Examples for Frame Relay MIB Enhancements

Example Setting the Load Interval for a PVC

In the following example, the load interval is set to 60 seconds for a Frame Relay PVC with the DLCI 100:

```
interface serial 1/1
 frame-relay interface-dlci 100
  load-interval 60
```

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

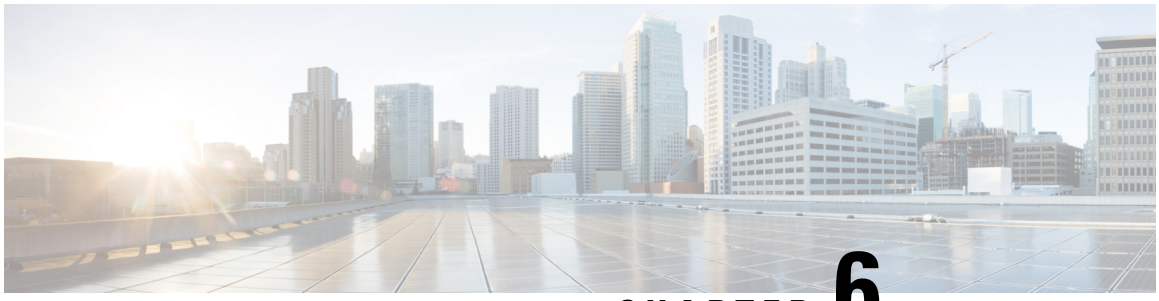
Feature Information for Frame Relay MIB Enhancements

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 4: Feature Information for Frame Relay MIB Enhancements

Feature Name	Releases	Feature Information
Frame Relay MIB Enhancements	Cisco IOS XE Release 2.1	The Cisco Frame Relay MIB describes managed objects that enable users to remotely monitor Frame Relay operations using SNMP.



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.

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 Interface Priority Queueing

- 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.

Restrictions for Frame Relay PVC Interface Priority Queueing

- 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.

Information About Frame Relay PVC Interface Priority Queueing

Feature Overview

The Cisco Frame Relay MIB describes managed objects that enable users to remotely monitor Frame Relay operations using Simple Network Management Protocol (SNMP). The Frame Relay MIB Enhancements feature extends the Cisco Frame Relay MIB by adding MIB objects to monitor the following Frame Relay functionality:

- Frame Relay fragmentation
- Frame Relay-ATM Network Interworking (FRF.5)
- Frame Relay-ATM Service Interworking (FRF.8)
- Frame Relay switching
- Input and output rates of individual virtual circuits (VCs)

The table below describes the MIB tables and objects that are introduced by the Frame Relay MIB enhancements. For a complete description of the MIB, see the Cisco Frame Relay MIB file CISCO-FRAME-RELAY-MIB.my, available through Cisco.com at the following URL:

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

Table 5: MIB Tables and Objects Introduced by the Frame Relay MIB Enhancements

Table or Object	Description
cfrFragTable	Table of Frame Relay fragmentation information.
cfrFRF5ConnectionTable	Table of Frame Relay-ATM Network Interworking connection information.
cfrFRF8ConnectionTable	Table of Frame Relay-ATM Service Interworking connection information.
cfrSwitchingTable	Table of Frame Relay switching entries.
cfrExtCircuitTxDataRate	Average rate, in bytes per second, at which data is transmitted on a circuit.
cfrExtCircuitTxPktRate	Average number of packets sent per second on a circuit.

Table or Object	Description
cfrExtCircuitRcvDataRate	Average rate, in bytes per second, at which data is received on a circuit.
cfrExtCircuitRcvPktRate	Average number of packets received per second on a circuit.

The Frame Relay MIB Enhancements feature also modifies the **load-interval** command to enable you to configure the load interval per permanent virtual circuit (PVC). The load interval is the length of time for which data is used to compute load statistics, including input rate in bits and packets per second, output rate in bits and packets per second, load, and reliability. Before the introduction of this feature, the load interval could be configured only for the interface.

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.

How to Configure Frame Relay PVC Interface Priority Queueing

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.

Command	Purpose
`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 for Frame Relay PVC Interface Priority Queuing

• Monitoring and Maintaining FR PIPQ, on page 61

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
```

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 Frame Relay PVC Interface Priority Queueing

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 Frame Relay PVC Interface Priority Queueing

Feature Name	Releases	Feature Information
Frame Relay PVC Interface Priority Queueing	Cisco IOS XE Release 2.1	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.

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.

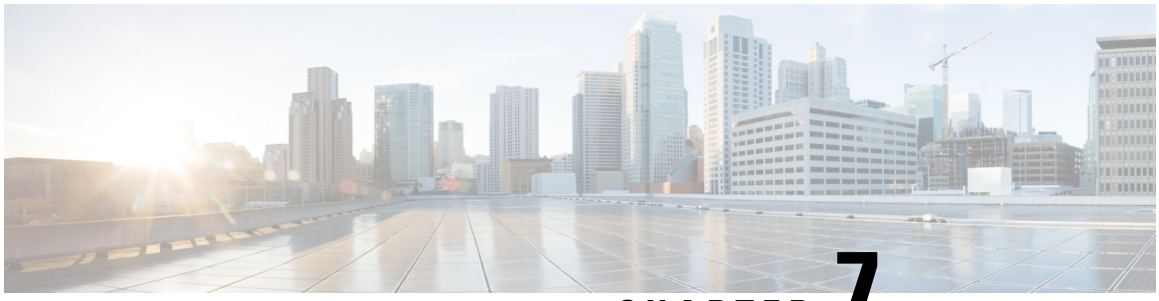
FRTS --Frame Relay traffic shaping. FRTS uses queues on a Frame Relay network to limit surges that can cause congestion. Data is buffered and then sent into the network in regulated amounts to ensure that the traffic will fit within the promised traffic envelope for the particular connection.

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.

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

ASR1K Frame Relay - Multilink (MLFR-FRF.16)

The ASR1K Frame Relay - Multilink (MLFR-FRF.16) feature is based on the Frame Relay Forum Multilink Frame Relay User-to-Network Interface/Network-to-Network Interface (UNI/NNI) Implementation Agreement (FRF.16.1) on Cisco Aggregation Services Routers. This feature provides a cost-effective way to increase the 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 UNI in Frame Relay networks.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see 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 for ASR1K Frame Relay - Multilink (MLFR-FRF.16), on page 80.

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

Prerequisites for ASR1K Frame Relay - Multilink (MLFR-FRF.16)

- MFR must be configured on the peer device.

Restrictions for ASR1K Frame Relay - Multilink (MLFR-FRF.16)

- Only the 2-octet Frame Relay format is supported.
- Only T1 and E1 speed members are supported in a bundle.
- All member links of a bundle must be of the same type.
- The following Shared Port Adapter (SPA) types are supported:
  - SPA-2XCT3/DS0
  - SPA-4XCT3/DS0
  - SPA-8XCHT1/E1
  - SPA-1XCHOC12/DS0
  - SPA-1XCHSTM1/OC3
- The following features are not supported with the ASR1K Frame Relay - Multilink (MLFR-FRF.16) feature:
  - 3- or 4-octet headers
  - Data-link connection identifier (DLCI) address mapping
  - Discard Eligibility (DE) bit manipulation
  - E1/T1 fractional links within the bundle
  - Frame Relay broadcast queue
  - Frame Relay backward explicit congestion notification (BECN) and forward explicit congestion notification (FECN) counting
  - Frame Relay Permanent Virtual Circuit (PVC) interface priority queuing (PIPQ) including DLCI prioritization
  - Frame Relay switching including NNI and FRF2.1
  - Frame Relay Traffic Policing (FRTP)
  - Frame Relay Traffic Shaping (FRTS)
  - FRF.16.1 Fragmentation
  - Generic Traffic Shaping (GTS)
  - Inverse Address Resolution Protocol (ARP)
  - PVC configuration over MFR bundle interface
  - Point-to-multipoint subinterfaces
  - Switched Virtual Circuits (SVC)
- An ISDN interface and any type of virtual interface cannot be a bundle link.
- The Multilink Frame Relay MIB (RFC 3020) is not supported.
- FRF.9 hardware compression over MFR is not supported.

Information About ASR1K Frame Relay - Multilink (MLFR-FRF.16)

Benefits of ASR1K Frame Relay - Multilink (MLFR-FRF.16)

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. MFR provides a cost-effective solution to this problem by allowing multiple T1 lines to be aggregated into a single bundle of bandwidth.

Increased Service Resilience

When multiple physical interfaces are provisioned as a single bundle, they provide more service resilience than a single physical interface. If a link fails, the bundle continues to support the Frame Relay service by transmitting across the remaining bundle links.

Scalability

ASR1K supports up to 992 MFR bundles.

- MFR bundles—The following table shows the maximum number of MFR bundles supported on ASR1K based on the number of links in a bundle:

Table 7: Maximum MFR Bundles

Links per Bundle	Number of MFR Bundles
1	992
2	496
3	330
4	248

- Frame Relay DLCI—The number of Frame Relay DLCIs that can be configured on MFR subinterfaces equals the maximum number of MFR bundles on ASR1K. The maximum number of Frame Relay DLCIs that you can configure on MFR subinterfaces and in one MFR bundle is 992.
- MFR subinterface—Because only point-to-point interfaces are supported, the number of DLCIs supported is equal to the number of MFR subinterfaces. Therefore, the maximum number of MFR subinterfaces and the maximum number of MFR interfaces supported in one bundle is 992.
- Physical Links—The maximum number of physical links supported in a bundle is 10.

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.

Variable Bandwidth Class Support

MFR FRF.16 variable bandwidth class support allows you to specify the criterion used to activate or deactivate a Frame Relay bundle.

Class A Single Link

The Frame Relay bundle is provisioned when one or more bundle links issue a BL_ACTIVATE message to indicate that an 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 a bundle link 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.

Note Activate and deactivate messages are implementation-oriented messages only. They are not visible in the output of the debug commands.

Class B All Links

The Frame Relay bundle is provisioned when all bundle links issue a BL_ACTIVATE message to indicate that an 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

A Frame Relay bundle is provisioned when a minimum number of links in the configured bundle issue a BL_ACTIVATE message, causing the bundle to emulate a physical link by issuing a PH_ACTIVATE message to the data link layer.

When the number of bundle links 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

MFR provides load balancing across bundle links within a bundle. If a bundle link that is chosen for transmission happens to be busy transmitting a long packet, the load-balancing mechanism can try another link, thus solving problems encountered when delay-sensitive packets have to wait.

ASR1K FRF.12 Support on MFR Interfaces

The ASR1K FRF.12 Support on MFR Interfaces feature enables the transport of realtime, delay-sensitive (voice) and nonrealtime, delay-insensitive (data) packets over the same, relatively slow-speed PVC.

During the transmission of packets, the larger, nonrealtime packets are fragmented into a sequence of smaller, mostly fixed-sized packets, also called fragments. The realtime packets are interleaved among the fragments. While receiving the packets, the nonrealtime fragments are reassembled and the resulting packets are forwarded along with the realtime packets. This approach minimizes the delay that can occur when nonrealtime and realtime traffic flow over the same PVC.

Benefits of ASR1K FRF.12

The ASR1K FRF.12 functionality prevents delay in Frame Relay networks by allowing edge routers to fragment large data packets before transmitting them across the network.

Limitations of ASR1K FRF.12

If a Frame Relay access device does not support FRF.12 fragmentation, the ASR1K FRF.12 Support on MFR Interfaces 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 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 PVCs.

Selecting a Fragment Size

You should set the fragment size based on the lowest port speed between routers. For example, for a hub-and-spoke Frame Relay topology, where the hub has a T1 speed and the remote routers have 64 kb/s port speeds, the fragmentation size must be set for 64 kb/s speed on both routers. Any other PVCs that share the same physical interface must use the same fragmentation size used by the voice PVC.

With pure end-to-end FRF.12 fragmentation, you should select a fragment size that is larger than the voice packet size.

The following table shows the recommended fragmentation sizes for a serialization delay of 10 ms:

Table 8: Recommended Fragment Size for 10 ms Serialization Delay

Lowest Link Speed in Path	Recommended Fragment Size
56 kb/s	70 bytes
64 kb/s	80 bytes
128 kb/s	160 bytes
256 kb/s	320 bytes
512 kb/s	640 bytes
768 kb/s	1000 bytes
1536 kb/s	1600 bytes

How to Enable ASR1K Frame Relay - Multilink (MLFR-FRF.16)

Configuring an MFR Bundle

Perform this task to configure an MFR bundle.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface mfr***interface-number*
4. **frame-relay multilink bandwidth-class** [**a** | **b** | **c** [*threshold*]]
5. **frame-relay intf-type** [**dce** | **dte**]
6. **frame-relay multilink bid** *name*
7. **exit**
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.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface mfr***interface-number* **Example:** `Router(config)# interface mfr1`	Configures an MFR bundle interface and enters interface configuration mode.
Step 4	**frame-relay multilink bandwidth-class** [**a** | **b** | **c** [*threshold*]] **Example:** `Router(config-if)# frame-relay multilink bandwidth-class a`	(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 deactivate when all bundle links are down (default). • Class B (all links)—The bundle will activate when all bundle links are up and 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 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** | **dte**] **Example:** `Router(config-if)# frame-relay intf-type dce`	Configures a device to function as data communication equipment (DCE). • **dce**—(Optional) The router or access server functions as a switch connected to a router. • **dte**—(Optional) The router or access server is connected to a Frame Relay network. **Note** Only one end of a link should be configured as DCE. The other end will function as data terminal equipment (DTE), which is the default setting.
Step 6	**frame-relay multilink bid** *name* **Example:** `Router(config-if)# frame-relay multilink bid router1`	(Optional) Assigns a bundle identification name to an MFR 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**

	Command or Action	Purpose
		shutdown commands in interface configuration mode (assuming that the physical state of the link is always up).
Step 7	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 8	**interface mfr***interface-number***.***subinterface-number* **point-to-point** **Example:** `Router(config)# interface mfr1.1 point-to-point`	Configures a point-to-point MFR subinterface and enters subinterface configuration mode.
Step 9	**ip address** *ip-address mask* **Example:** `Router(config-subif)# ip address 10.0.1.1 255.255.255.0`	Configures an IP address for the subinterface.
Step 10	**frame-relay interface-dlci** *dlci* **Example:** `Router(config-subif)# frame-relay interface-dlci 100`	Assigns a DLCI to a Frame Relay subinterface and enters Frame Relay DLCI configuration mode. • The DLCI range is from 16 to 1007.
Step 11	**end** **Example:** `Router(config-fr-dlci)# end`	Exits Frame Relay DLCI configuration mode 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 an MFR Bundle Link

Tip To minimize the latency that results from the arrival order of packets, Cisco recommends bundling physical links of the same line speed in one bundle.

Perform this task to configure an MFR bundle link.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface serial** *number*
4. **encapsulation frame-relay mfr***number* [*name*]

5. **frame-relay multilink lid** *name*
6. **frame-relay multilink hello** *seconds*
7. **frame-relay multilink ack** *seconds*
8. **frame-relay multilink retry** *number*
9. **end**
10. **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.
Step 4	**encapsulation frame-relay mfr***number* [*name*] **Example:** `Router(config-if)# encapsulation frame-relay mfr1`	Creates an MFR bundle link and associates the link with a bundle.
Step 5	**frame-relay multilink lid** *name* **Example:** `Router(config-if)# frame-relay multilink lid first-link`	(Optional) Assigns a bundle link identification name to an MFR bundle link. • The bundle link identification (LID) is not functional 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 6	**frame-relay multilink hello** *seconds* **Example:** `Router(config-if)# frame-relay multilink hello 9`	(Optional) Configures the interval in seconds after which a bundle link will send out hello messages. • The default value is 10 seconds.
Step 7	**frame-relay multilink ack** *seconds* **Example:** `Router(config-if)# frame-relay multilink ack 6`	(Optional) Configures the interval (in seconds) for which a bundle link will wait for a hello message acknowledgment before resending the hello message. • The default value is 4 seconds.

	Command or Action	Purpose
Step 8	**frame-relay multilink retry** *number* **Example:** `Router(config-if)# frame-relay multilink retry 3`	(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.
Step 9	**end** **Example:** `Router(config-if)# end`	Ends the configuration session and returns to privileged EXEC mode.
Step 10	**show frame-relay multilink** **Example:** `Router# show frame-relay multilink`	(Optional) Displays the current Frame Relay multilink configuration.

Configuring FRF.12 on an MFR Bundle Interface

Before you begin

You must create a class map and a policy map before enabling FRF.12 fragmentation of Frame Relay frames. For the class map, define a differentiated services code point (DSCP) value as the match criterion.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface mfr***interface-number*
4. **no ip address**
5. **frame-relay fragment** *fragment-size* **end-to-end**
6. **service-policy output** *policy-map-name*
7. **exit**
8. **interface mfr***interface-number***.***subinterface-number* **point-to-point**
9. **ip address** *ip-address mask*
10. **frame-relay interface-dlci** *dlci-value*
11. **end**
12. **show frame-relay fragment** [**interface** *interface* [*dlci*]]

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	**interface mfr***interface-number* **Example:** `Router(config)# interface mfr1`	Configures an MFR bundle interface and enters interface configuration mode.
Step 4	**no ip address** **Example:** `Router(config-if)# no ip address`	Disables IP processing.
Step 5	**frame-relay fragment** *fragment-size* **end-to-end** **Example:** `Router(config-if)# frame-relay fragment 300 end-to-end`	Enables FRF.12 end-to-end fragmentation of Frame Relay frames. • The valid size range is from 16 to 1600. • 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 6	**service-policy output** *policy-map-name* **Example:** `Router(config-if)# service-policy output pmap1`	Attaches a policy map to an output interface that is to be used as the service policy for that interface.
Step 7	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 8	**interface mfr***interface-number.subinterface-number* **point-to-point** **Example:** `Router(config)# interface mfr1.1 point-to-point`	Configures a point-to-point MFR subinterface and enters subinterface configuration mode.
Step 9	**ip address** *ip-address mask* **Example:** `Router(config-subif)# ip address 10.1.1.1 255.255.255.0`	Configures the IP address of the subinterface.
Step 10	**frame-relay interface-dlci** *dlci-value* **Example:** `Router(config-subif)# frame-relay interface-dlci 100`	Assigns a DLCI to the MFR subinterface and enters Frame Relay DLCI configuration mode. • The DLCI range is from 16 to 1007.

	Command or Action	Purpose
Step 11	**end** **Example:** `Router(config-fr-dlci)# end`	Exits Frame Relay DLCI configuration mode and returns to privileged EXEC mode.
Step 12	Required: **show frame-relay fragment** [**interface** *interface* [*dlci*]] **Example:** `Router# show frame-relay fragment`	(Optional) Displays statistics about Frame Relay fragmentation.

Monitoring and Maintaining MFR Bundles and Bundle Links

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 MFR 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 MFR bundles and bundle links.
Step 4	**show interfaces mfr***number* **Example:** `Router# show interfaces mfr1`	(Optional) Displays information and packet statistics for the bundle interface.

Examples

The following is sample output from the **show frame-relay multilink** command. Because a particular bundle or bundle link is not specified, information about all bundles and bundle links is displayed.

```
Router# show frame-relay multilink

Bundle: mfr1, State = down, class = A, fragmentation disabled
 BID = router1
 Bundle links:
  Serial3/1, HW state = Administratively down, link state = Down, LID = second-link
  Serial3/2, HW state = up, link state = Add_sent, LID = first-link
Bundle: mfr1, State = down, class = B, fragmentation disabled
 BID = router1
 Bundle links:
  Serial3/0, HW state = Administratively down, link state = Down, LID = third-link
  Serial3/1, HW state = Administratively down, link state = Down, LID = second-link
  Serial3/2, HW state = up, link state = Add_sent, LID = first-link
```

The following is sample output from the **show frame-relay multilink** command when a Frame Relay bundle is configured as bandwidth class C (threshold):

```
Router# show frame-relay multilink

Bundle: mfr2, State = down, class = C (threshold 100), fragmentation disabled
 BID = router2
 Bundle links:
  Serial3/1, HW state = Administratively down, link state = Down, LID = cisco2
  Serial3/0, HW state = Administratively down, link state = Down, LID = cisco1
```

The following is sample output from the **show frame-relay multilink** command when the **serial** *number* keyword and argument pair is specified. It displays information about the specified bundle link.

```
Router# show frame-relay multilink Serial 3/2

Bundle links:
  Serial3/2, HW state = up, link state = Add_sent, LID = first-link
  Bundle interface = mfr1,  BID = router1
```

The following is sample output from the **show frame-relay multilink** command when the **serial** *number* keyword and argument pair and the **detailed** option are specified. Detailed information about the specified bundle links is displayed.

```
Router# show frame-relay multilink Serial 3/2 detail

Bundle links:
  Serial3/2, HW state = up, link state = Add_sent, LID = first-link
  Bundle interface = mfr1,  BID = router1
    Cause code = none, Ack timer = 6, Hello timer = 9,
    Max retry count = 3, Current count = 0,
    Peer LID = , RTT = 0 ms
    Statistics:
    Add_link sent = 110, Add_link rcv'd = 0,
    Add_link ack sent = 0, Add_link ack rcv'd = 0,
    Add_link rej sent = 0, 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
```

Configuration Examples for ASR1K Frame Relay - Multilink (MLFR-FRF.16)

Example: Configuring Multilink Frame Relay

The following example shows the configuration of bundle MFR1, where serial interfaces 3/0 and 3/2 are configured as bundle links:

```
interface MFR1
 no ip address
 frame-relay intf-type dce
 frame-relay multilink bid router1
!
interface MFR1.1 point-to-point
 ip address 10.0.0.1 255.255.255.0
 frame-relay interface-dlci 100
interface Serial3/0
 encapsulation frame-relay MFR1
 frame-relay multilink lid first-link
 frame-relay multilink hello 9
 frame-relay multilink retry 3
 frame-relay multilink ack 4
interface Serial3/2
 encapsulation frame-relay MFR1
 frame-relay multilink lid first-link
 frame-relay multilink hello 8
 frame-relay multilink ack 3
 frame-relay multilink retry 2
```

Example: Configuring Variable Bandwidth Class Support

The following example shows how to configure Frame Relay bundle MFR2 to use the class B (all links) criterion to get activated or deactivated:

```
interface MFR2
 frame-relay multilink bandwidth-class b
 frame-relay intf-type dce
 frame-relay multilink bid router2
 exit
interface MFR2.2 point-to-point
 ip address 10.1.1.10 255.255.255.0
 frame-relay interface-dlci 145
  end
```

Example: Configuring FRF.12 on an MFR Interface

The following example shows how to configure FRF.12 on an MFR interface:

```
class-map match-any tos_111
 match dscp cs7
policy-map voip
 class tos_111
  priority percent 100
interface mfr1
 frame-relay multilink bid 1
```

```
 frame-relay multilink bandwidth-class a
 frame-relay fragment 100 end-to-end
 service-policy output voip
interface mfr1.1 point-to-point
 ip address 70.1.1.1 255.255.255.0
 frame-relay interface-dlci 100
```

The following output shows the result of the above configuration:

```
Router# show frame-relay fragment

interface  dlci  frag-type   size  in-frag  out-frag  dropped-frag
mfr1.1     100   end-to-end  100   0        0         0
```

The size column displays the configured fragment size in bytes.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
WAN commands: complete command syntax, command mode, defaults, usage guidelines, and examples	Cisco IOS Wide-Area Networking Command Reference
Frame Relay configuration	Configuring Frame Relay

Standards and RFCs

Standard/RFC	Title
FRF.16.1	*Multilink Frame Relay UNI/NNI Implementation Agreement,* May 2002

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 ASR1K Frame Relay - Multilink (MLFR-FRF.16)

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 9: Feature Information for ASR1K Frame Relay - Multilink (MLFR-FRF.16)

Feature Name	Releases	Feature Information
ASR1K Frame Relay - Multilink (MLFR-FRF.16)	Cisco IOS XE Release 3.4S	The ASR1K Frame Relay - Multilink (MLFR-FRF.16) feature is based on the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16.1) on Aggregation Service Routers. 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 retry**, **interface mfr**, **show frame-relay multilink**.
ASR1K FRF.12 Support on MFR Interfaces	Cisco IOS XE Release 3.5S	The following sections provide information about this feature: • ASR1K FRF.12 Support on MFR Interfaces • Configuring FRF.12 on an MFR Bundle Interface

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.

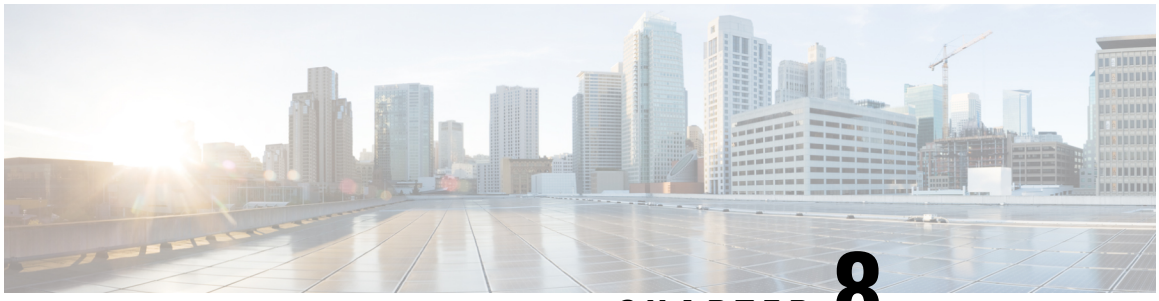
LMI --Local Management Interface. A set of enhancements to the basic Frame Relay specification. LMI includes support for a keepalive mechanism, which verifies that data is flowing; a multicast mechanism, which provides the network server with its local DLCI and the multicast DLCI; global addressing, 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 8

Frame Relay show Command and debug Command Enhancements

The Frame Relay show Command and debug Command Enhancements feature provides the ability to filter the output of certain Frame Relay **show** and **debug** commands on the basis of the interface and data-link connection identifier (DLCI). These enhancements facilitate network scalability and simplify network management and troubleshooting.

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.

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 show Command and debug Command Enhancements

Overview of the Frame Relay show Command and debug Command Enhancements

This feature introduces the following enhancments:

- The **show frame-relay map** command has been enhanced to allow map information to be displayed for specific interfaces and DLCIs.

- The **show frame-relay ip tcp header-compression** and **show frame-relay ip rtp header-compression** commands have been enhanced to allow header-compression information to be displayed for specific DLCIs.
- The **summary** keyword was added to the **show frame-relay pvc** command, allowing a summary of all PVCs on the system to be displayed.
- Conditional debugging support, which allows debug output to be filtered on the basis of interface and DLCI, was introduced for the following commands:
  - **debug frame-relay end-to-end**
  - **debug frame-relay events**
  - **debug frame-relay fragment**
  - **debug frame-relay fragment event**
  - **debug frame-relay ip**
  - **debug frame-relay ppp**
  - **debug frame-relay verbose**

Note Conditional debugging for Frame Relay **debug** commands is configured by using the **debug condition** command.

Benefits of the Frame Relay Show Command and Debug Command Enhancements

The Frame Relay show Command and debug Command Enhancements allow the output for some Frame Relay **show** commands and **debug** commands to be filtered on the basis of interface and DLCI. This enhancement saves network administrators time and frustration by eliminating the need to look through a large amount of output for information about a specific interface or DLCI. These enhancements can also reduce the amount of CPU processing time that is required to generate large amounts of **show** and **debug** output.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS XE Wide-Area Networking configuration tasks	*Cisco IOS XE Wide-Area Networking Configuration Guide*, Release 2
Wide-Area networking commands	*Cisco IOS Wide-Area Networking Command Reference*

Standards

Standard	Title
None	--

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
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

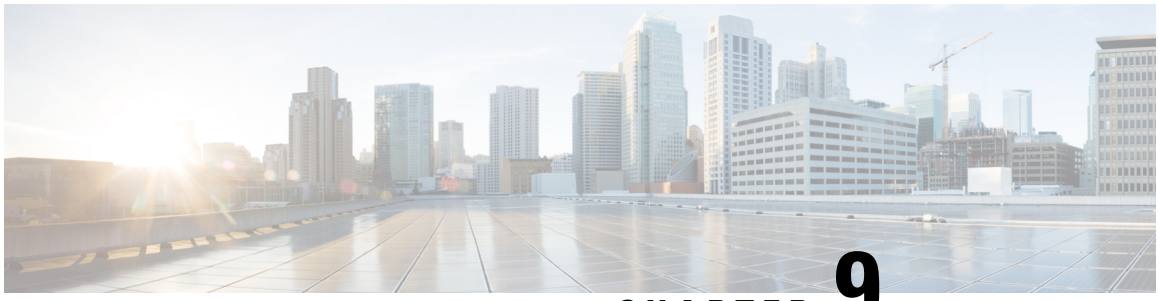
Feature Information for Frame Relay show Command and debug Command Enhancements

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 10: Feature Information for Frame Relay show Command and debug Command Enhancements

Feature Name	Releases	Feature Information
Frame Relay show Command and debug Command Enhancements	Cisco IOS XE Release 2.1	The Frame Relay show Command and debug Command Enhancements feature provides the ability to filter the output of certain Frame Relay **show** and **debug** commands on the basis of the interface and DLCI.



CHAPTER 9

L2VPN Local Switching—Frame Relay-Ethernet/VLAN

L2VPN Local Switching—Frame Relay-Ethernet/VLAN feature allows you to switch Frame Relay and Ethernet frames between two interfaces on the same device.

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.

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.

Restrictions for L2VPN Local Switching—Frame Relay-Ethernet/VLAN

The following functions are not supported:

- Frame Relay-to-Ethernet IP-Mode local switching
- Frame Relay-to-Ethernet VLAN-Mode local switching
- Frame Relay Multilink Frame Relay (MFR)

Information About L2VPN Local Switching—Frame Relay-Ethernet/VLAN

L2VPN Local Switching—Frame Relay-Ethernet/VLAN Overview

The L2VPN Local Switching—Frame Relay-Ethernet/VLAN feature switches a Frame Relay frame to an Ethernet VLAN/QinQ frame over the same provider edge (PE) device. Only Ethernet (bridged) interworking mode is supported to switch packets between Frame Relay link and Ethernet VLAN/QinQ. In a bridged interworking mode, the MAC header is considered as the payload of Frame Relay frames.

The L2VPN Local Switching—Frame Relay-Ethernet/VLAN supports the following functions:

- The Frame Relay-Ethernet bridge mode local switching in data-link connection identifier (DLCI) mode.
- Port interface and subinterface Ethernet attachment circuit (AC) type with single tag or double tags (Q-in-Q).
- Cisco and IETF Frame Relay encapsulation.

The Frame Relay-Ethernet local switching topology is illustrated in the figure below.

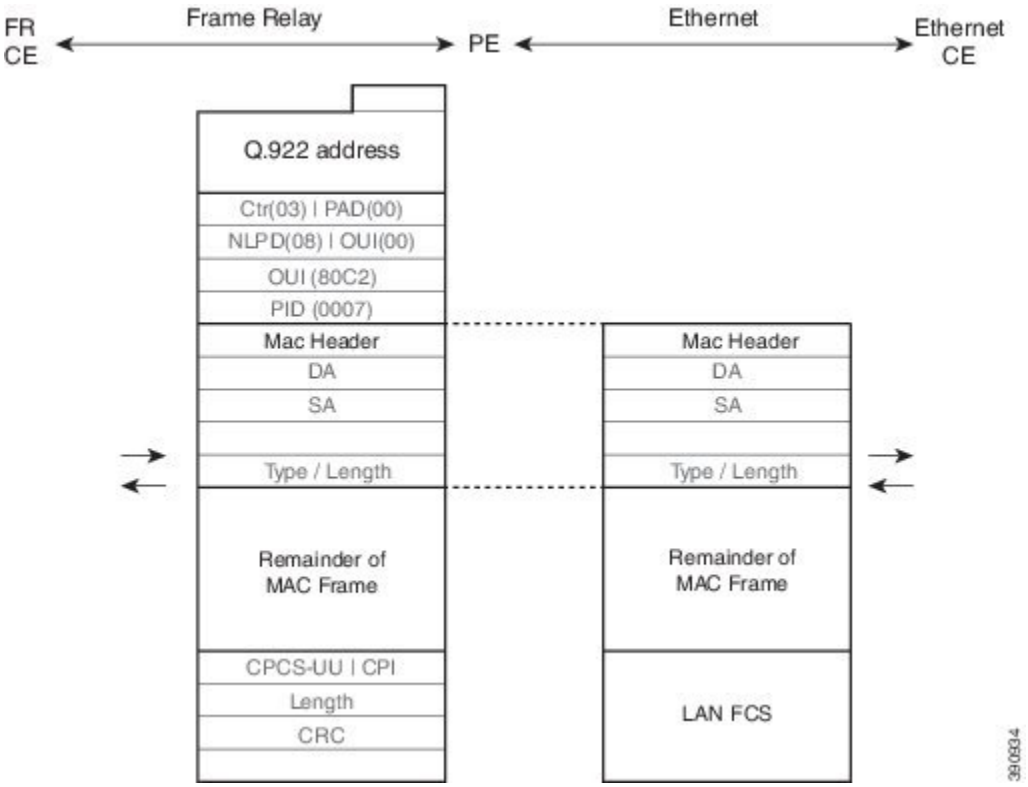
Figure 4: Frame Relay-Ethernet Local Switching Topology

Frame Relay to Ethernet Port-Bridged Interworking

Frame Relay-Ethernet port-bridged interworking provides interoperability between a Frame Relay attachment virtual circuit (VC) and an Ethernet attachment VC connected to the same provider edge (PE) device. The bridged encapsulation is used that corresponds to the bridged (Ethernet) interworking mechanism.

Based on RFC 2427, *Multiprotocol Interconnect over Frame Relay*, the interworking is done at the PE connected to the Frame Relay attachment VC as shown in the figure below.

Figure 5: Protocol Stack for Frame Relay to Ethernet Port Bridged Interworking



The processing of Frame Relay-Ethernet port local switching is described as follows:

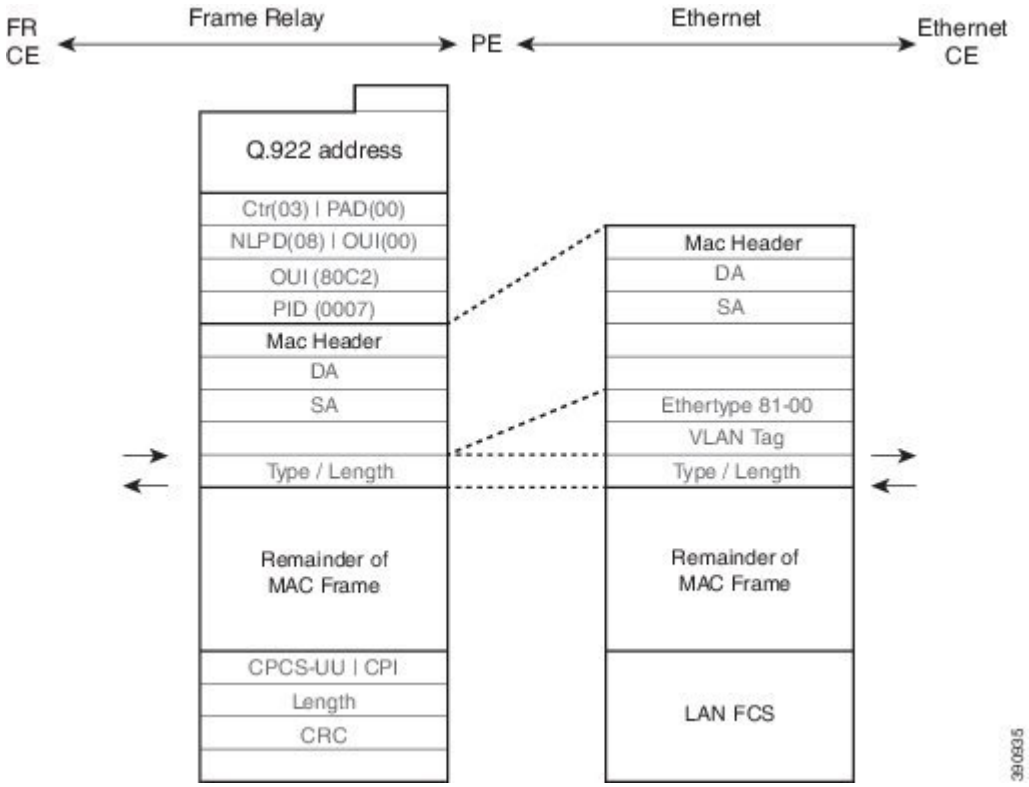
- In the direction from Frame Relay to Ethernet:
  - On the Frame Relay side, the Frame Relay header and trailer are removed. The packet is forwarded to Ethernet side.
  - On the Ethernet side, the MAC header is ignored.
- In the direction from Ethernet to Frame Relay:
  - On the Ethernet side, the MAC header is ignored.
  - On the Frame Relay side, the Frame Relay header is generated and added to the packet that is sent to the Frame Relay customer edge (CE) device.

Frame Relay to Ethernet VLAN/QinQ–Bridged Interworking

Frame Relay to Ethernet VLAN/QinQ bridged interworking provides interoperability between a Frame Relay attachment virtual circuit (VC) and an Ethernet VLAN attachment VC connected to the same provider edge (PE) device. The bridged encapsulation is used that corresponds to the bridged (Ethernet) interworking mechanism.

Based on RFC 2427, *Multiprotocol Interconnect over Frame Relay*, the interworking function is implemented on the PE connected to the Frame Relay attachment VC as shown in the figure below.

Figure 6: Protocol Stack for Frame Relay to Ethernet VLAN/QinQ Bridged Interworking



The process of Frame Relay to VLAN/QinQ bridged interworking is described as follows:

- In the direction from Frame Relay to Ethernet:
  - On the Frame Relay side, the Frame Relay header and trailer are removed. The packet is forwarded to Ethernet side.
  - On the Ethernet side, one or two VLAN tags are generated per the configuration and inserted into L2 header, which is referred as VLAN tag push.
- In the direction from Ethernet to Frame Relay:
  - On the Ethernet side, the one or two VLAN tags are removed. The packet is then forwarded to Frame Relay side.
  - On the Frame Relay side, the Frame Relay header is generated and added to the packet that is sent to the Frame Relay customer edge (CE) device.

How To Configure L2VPN Local Switching—Frame Relay-Ethernet/VLAN

Configuring Frame Relay-Ethernet Port-Bridged Interworking

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no ip address**
5. **exit**
6. **interface** *type number*
7. **encapsulation frame-relay**
8. **frame-relay interface-dlci** *dlci* **switched**
9. **exit**
10. **connect** *connection-name type number dlci* **interworking ethernet**
11. **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** *type number* **Example:** `Device(config)# interface GigabitEthernet 0/0/0`	Specifies the interface type and number and enters interface configuration mode.
Step 4	**no ip address** **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 6	**interface** *type number* **Example:** `Device(config)# interface Serial 0/0/0:0`	Specifies the subinterface type and number and enters subinterface configuration mode.
Step 7	**encapsulation frame-relay** **Example:** `Device(config-subif)# encapsulation frame-relay`	Enables Frame Relay encapsulation.
Step 8	**frame-relay interface-dlci** *dlci* **switched** **Example:** `Device(config-subif)# frame-relay interface-dlci 57 switched`	Indicates that a Frame Relay data-link connection identifier (DLCI) is switched. The range is from 16 to 1007.
Step 9	**exit** **Example:** `Device(config-subif)# exit`	Exits subinterface configuration mode and returns to global configuration mode.
Step 10	**connect** *connection-name type number dlci* **interworking ethernet** **Example:** `Device(config)# connect Eth-Ser GigabitEthernet0/0/0 Serial0/0/0:0 57 interworking ethernet`	Creates Layer 2 data connections between two ports on the same device.
Step 11	**end** **Example:** `Device# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring Frame Relay-Ethernet VLAN/QinQ Interworking

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **encapsulation dot1q** *vlan-id* **second-dot1q** *second vlan-id*
5. **exit**
6. **interface** *type number*
7. **encapsulation frame-relay**
8. **frame-relay interface-dlci** *dlci* **switched**
9. **exit**
10. **connect** *connection-name type number dlci* **interworking ethernet**
11. **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** *type number* **Example:** `Device(config)# interface GigabitEthernet 0/0/0.1`	Specifies a subinterface type and number and enters subinterface configuration mode.
Step 4	**encapsulation dot1q** *vlan-id* **second-dot1q** *second vlan-id* **Example:** `Device(config-subif)# encapsulation dot1q 3 second-dot1q 4`	Specifies QinQ as the encapsulation method.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits subinterface configuration mode and returns to global configuration mode.
Step 6	**interface** *type number* **Example:** `Device(config)# interface Serial 0/0/0:0`	Specifies a subinterface type and number and enters subinterface configuration mode.
Step 7	**encapsulation frame-relay** **Example:** `Device(config-subif)# encapsulation frame-relay`	Enables Frame Relay encapsulation.
Step 8	**frame-relay interface-dlci** *dlci* **switched** **Example:** `Device(config-subif)# frame-relay interface-dlci 58 switched`	Indicates that a Frame Relay data-link connection identifier (DLCI) is switched.
Step 9	**exit** **Example:** `Device(config-subif)# exit`	Exits subinterface configuration mode and returns to global configuration mode.
Step 10	**connect** *connection-name type number dlci* **interworking ethernet** **Example:**	Creates Layer 2 data connections between two ports on the same device.

	Command or Action	Purpose
	`Device(config)# connect Eth-FR GigabitEthernet0/0/0.1 Serial0/0/0:0 58 interworking ethernet`	
Step 11	**end** **Example:** `Device# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuration Examples for L2VPN Local Switching—Frame Relay-Ethernet/VLAN

Example: Configuring Frame Relay-Ethernet Port Mode Bridged Interworking

The following example shows how to configure the Frame Relay-Ethernet port mode bridged interworking:

PE configuration:

```
interface GigabitEthernet0/0/1
   no ip address
   end
interface Serial0/1/2:0
   no ip address
   encapsulation frame-relay
   no keepalive
   frame-relay interface-dlci 60 switched
   end
connect FR-ETHQinQ Serial0/1/2:0 60 GigabitEthernet0/0/1 interworking ethernet
```

CE configuration: Frame-Relay-CE

```
 bridge irb
 bridge 16 protocol ieee
 bridge 16 route ip
 interface Serial2/0:0
   no ip address
   encapsulation frame-relay IETF
   no keepalive
  interface Serial2/0:0.1 point-to-point
   frame-relay interface-dlci 60
   bridge-group 60
  interface BVI16
   ip address 172.16.1.0 255.255.0.0
```

Ethernet-CE

```
interface GigabitEthernet0/0/1
 ip address 172.16.2.1 255.255.0.0
```

Example: Configuring Frame Relay-Ethernet VLAN 802.1Q Bridged Interworking

The following example shows how to configure Frame Relay-Ethernet VLAN 802.1Q bridged interworking:

PE configuration:

```
interface GigabitEthernet0/0/1.10
  encapsulation dot1Q 10
  end
interface Serial0/1/2:0
   no ip address
   encapsulation frame-relay
   no keepalive
   frame-relay interface-dlci 58 switched
   end
 connect FR-ETH1Q Serial0/1/2:0 58 GigabitEthernet0/0/1.10 interworking Ethernet
```

CE configuration: Frame Relay-CE

```
bridge irb
bridge 16 protocol ieee
bridge 16 route ip
interface Serial2/0:0
  no ip address
  encapsulation frame-relay IETF
  no keepalive
interface Serial2/0:0.1 point-to-point
  frame-relay interface-dlci 58
  bridge-group 16
interface BVI16
  ip address 172.18.1.2 255.255.0.0
```

Ethernet-CE

```
interface GigabitEthernet0/0/1.10
  encapsulation dot1Q 10
  ip address 172.17.2.1 255.255.0.0
```

Example: Configuring Frame Relay-VLAN QinQ Bridged Interworking

The following example shows how to configure Frame Relay-VLAN QinQ bridged interworking:

PE configuration:

```
interface GigabitEthernet0/0/1.11
  encapsulation dot1Q 11 second-dot1q 100
  end
interface Serial0/1/2:0
  no ip address
  encapsulation frame-relay
  no keepalive
  frame-relay interface-dlci 100 switched
  end
connect FR-ETHQinQ Serial0/1/2:0 100 GigabitEthernet0/0/1.11 interworking ethernet
```

CE configuration: Frame-Relay-CE

```
 bridge irb
 bridge 16 protocol ieee
 bridge 16 route ip
```

```
interface Serial2/0:0
  no ip address
  encapsulation frame-relay IETF
  no keepalive
interface Serial2/0:0.1 point-to-point
  frame-relay interface-dlci 100
  bridge-group 16
interface BVI16
  ip address 172.18.1.3 255.255.0.0
```

Ethernet-CE

```
interface GigabitEthernet0/0/1.10
  encapsulation dot1Q 10 sec 10
  ip address 172.19.1.1 255.255.0.0
```

Additional References for L2VPN Local Switching—Frame Relay-Ethernet/VLAN

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
WAN commands	Cisco IOS Wide Area Network Command Reference

RFCs

RFC	Title
RFC 2427	Multiprotocol Interconnect over Frame Relay

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 L2VPN Local Switching—Frame Relay-Ethernet/VLAN

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 11: Feature Information for L2VPN Local Switching—Frame Relay-Ethernet/VLAN

Feature Name	Releases	Feature Information
L2VPN Local Switching—Frame Relay-Ethernet/VLAN		The L2VPN Local Switching—Frame Relay-Ethernet/VLAN feature allows you to switch Frame Relay and Ethernet frames between two interfaces on the same device.