Configuring Frame Relay Cisco IOS XR Software

This module describes the optional configurable Frame Relay parameters available on Packet-over-SONET/SDH (POS), multilink, and serial interfaces configured with Frame Relay encapsulation.

Feature History for Configuring Frame Relay Interfaces on Cisco IOS XR Software

Release	Modification
Release 3.4.0	This feature was introduced on the Cisco XR 12000 Series Router.
Release 3.5.0	This feature was updated to support IPv6. Layer 2 Tunnel Protocol Version 3 (L2TPv3) was supported on serial and POS interfaces with Frame Relay encapsulation.
Release 3.6.0	Multilink Frame Relay (FRF.16) and End-to-End Fragmentation (FRF.12) were introduced on the Cisco 1-Port Channelized STM-1/OC-3 shared port adapter (SPA) and the 2-Port and 4-Port Channelized T3 SPAs on the Cisco XR 12000 Series Routers.
Release 3.8.0	FRF.16 and FRF.12 were introduced on the Cisco 1-Port Channelized OC-12 SPA on the Cisco XR 12000 Series Routers. Support for Frame Relay was introduced on the following line cards on the Cisco XR 12000 Series Router: • Cisco 1-port Channelized OC-12/STM-1 Line Card • Cisco 4-port Channelized OC-12/STM-4 Line Card
Release 4.0.0	Support for the following frame relay features was added for the Cisco 8-Port Channelized T1/E1 SPA on the Cisco XR 12000 Series Router: • Multilink Frame Relay (FRF.16) • End-to-End Fragmentation (FRF.12) Support for fragmentation counters using the **fragment-counter** command was added for the following SPAs: • Cisco 1-Port Channelized OC-3/STM-1 SPA • Cisco 4-Port Channelized T3/DS0 SPA • Cisco 8-Port Channelized T1/E1 SPA

Contents

- Prerequisites for Configuring Frame Relay, page 488
- Information About Frame Relay Interfaces, page 488
- Configuring Frame Relay, page 495
- Configuration Examples for Frame Relay, page 511
- Additional References, page 515

Prerequisites for Configuring Frame Relay

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Before configuring Frame Relay, be sure that the following conditions are met:

- Your hardware must support POS or serial interfaces.
- You have enabled Frame Relay encapsulation on your interface with the **encapsulation frame relay** command, as described in the appropriate module:
  - To enable Frame Relay encapsulation on a multilink bundle interface, see the "Configuring Multilink Frame Relay Bundle Interfaces" section on page 500.
  - To enable Frame Relay encapsulation on a POS interface, see the *"Configuring POS Interfaces on Cisco IOS XR Software"* module in this manual.
  - To enable Frame Relay encapsulation on a serial interface, see the *Configuring Serial Interfaces on Cisco IOS XR Sotware* module in this manual.

Information About Frame Relay Interfaces

The following sections explain the various aspects of configuring Frame Relay interfaces:

- Frame Relay Encapsulation, page 488
- Multilink Frame Relay (FRF.16), page 491
- End-to-End Fragmentation (FRF.12), page 495

Frame Relay Encapsulation

On the Cisco XR 12000 Series Router, Frame Relay is supported on POS and serial main interfaces, and on PVCs that are configured under those interfaces. To enable Frame Relay encapsulation on an interface, use the **encapsulation frame-relay** command in interface configuration mode.

Frame Relay interfaces support two types of encapsulated frames:

- Cisco (this is the default)
- IETF

Use the **encapsulation frame-relay** command in interface configuration mode to configure Cisco or IETF encapsulation on a PVC.

Note If the encapsulation type is not configured explicitly for a PVC with the **encapsulation** command, then that PVC inherits the encapsulation type from the main interface.

The **encapsulation frame relay** and **encap (PVC)** commands are described in the following modules:

- To enable Frame Relay encapsulation on a POS interface, see the *"Configuring POS Interfaces on Cisco IOS XR Software"* module in this manual.
- To enable Frame Relay encapsulation on a serial interface, see the *Configuring Serial Interfaces on Cisco IOS XR Sotware* module in this manual.

When an interface is configured with Frame Relay encapsulation and no additional configuration commands are applied, the default interface settings shown in Table 23 are present. These default settings can be changed by configuration as described in this module.

Table 23 Frame Relay Encapsulation Default Settings

Parameter	Configuration File Entry	Default Settings	Command Mode
PVC Encapsulation	**encap** {**cisco** | **ietf**}	**cisco** **Note** When the **encap** command is not configured, the PVC encapsulation type is inherited from the Frame Relay main interface.	PVC configuration
Type of support provided by the interface	**frame-relay intf-type** {**dce** | **dte**}	**dte**	Interface configuration
LMI type supported on the interface	**frame-relay lmi-type** [**ansi** | **cisco** | **q933a**]	For a DCE, the default setting is **cisco**. For a DTE, the default setting is synchronized to match the LMI type supported on the DCE. **Note** To return an interface to its default LMI type, use the **no frame-relay lmi-type** [**ansi** | **cisco** | **q933a**] command.	Interface configuration
Disable or enable LMI	**frame-relay lmi disable**	LMI is enabled by default on Frame Relay interfaces. To reenable LMI on an interface after it has been disabled, use the **no frame-relay lmi disable** command.	Interface configuration

Note The default settings of LMI polling-related commands appear in Table 24 on page 490 and Table 25 on page 491.

LMI

The Local Management Interface (LMI) protocol monitors the addition, deletion, and status of PVCs. LMI also verifies the integrity of the link that forms a Frame Relay User-Network Interface (UNI).

Frame Relay interfaces supports the following types of LMI on UNI interfaces:

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

Use the **frame-relay lmi-type** command to configure the LMI type to be used on an interface.

Note The LMI type that you use must correspond to the PVCs configured on the main interface. The LMI type must match on both ends of a Frame Relay connection.

If your router functions as a switch connected to another non-Frame Relay router, use the **frame-relay intf-type dce** command to configure the LMI type to support data communication equipment (DCE).

If your router is connected to a Frame Relay network, use the **frame-relay intf-type dte** command to configure the LMI type to support data terminal equipment (DTE).

Note LMI type auto-sensing is supported on DTE interfaces by default.

Use the **show frame-relay lmi** and **show frame-relay lmi-info** commands in EXEC mode to display information and statistics for the Frame Relay interfaces in your system. (When specifying the *type* and *interface-path-id* arguments, you must specify information for the main interface.) You can modify the error threshold, event count, and polling verification timer and then use the **show frame-relay lmi** command to gather information that can help you monitor and troubleshoot Frame Relay interfaces.

If the LMI type is **cisco** (the default LMI type), the maximum number of PVCs that can be supported under a single interface is related to the MTU size of the main interface. Use the following formula to calculate the maximum number of PVCs supported on a card or SPA :

```
(MTU - 13)/8 = maximum number of PVCs
```

The default number of PVCs supported on POS PVCs configured with **cisco** LMI is 557, while the default number of PVCs supported on serial PVCs configured with **cisco** LMI is 186.

For LMI types that are not from Cisco, up to 992 PVCs are supported under a single main interface.

Note If a specific LMI type is configured on an interface, use the **no frame-relay lmi-type** [**ansi** | **cisco** | **q933a**] command to bring the interface back to the default LMI type.

Table 24 describes the commands that can be used to modify LMI polling options on PVCs configured for a DCE.

Table 24 LMI Polling Configuration Commands for DCE

Parameter	Configuration File Entry	Default Settings
Sets the error threshold on a DCE interface.	**lmi-n392dce** *threshold*	**3**
Sets the monitored event count.	**lmi-n393dce** *events*	**4**
Sets the polling verification timer on the DCE end.	**lmi-t392dce** *seconds*	**15**

Table 25 describes the commands that can be used to modify LMI polling options on PVCs configured for a DTE.

Table 25 LMI Polling Configuration Commands for DTE

Parameter	Configuration File Entry	Default Settings
Set the number of Line Integrity Verification (LIV) exchanges performed before requesting a full status message.	**lmi-n391dte** *polling-cycles*	**6**
Sets the error threshold.	**lmi-n392dte** *threshold*	**3**
Sets the monitored event count.	**lmi-n393dte** *events*	**4**
Sets the polling interval (in seconds) between each status inquiry from the DTE end.	**frame-relay lmi-t391dte** *seconds*	**10**

Multilink Frame Relay (FRF.16)

Multilink Frame Relay (MFR) is supported only on the following shared port adapters (SPAs):

- Cisco 1-Port Channelized STM-1/OC-3 SPA
- 2-Port and 4-Port Channelized T3 SPA
- Cisco 1-port Channelized OC-12 SPA
- Cisco 8-Port Channelized T1/E1 SPA

Multilink Frame Relay High Availability

MFR supports the following levels of high availability support:

- MFR supports a process restart, but some statistics will be reset during a restart of certain processes.
- MFR member links remain operational during a route processor (RP) switchover.

Multilink Frame Relay Configuration Overview

A multilink Frame Relay interface is part of a multilink bundle that allows Frame Relay encapsulation on its interfaces. You create a multilink Frame Relay interface by configuring the following components:

- MgmtMultilink controller
- Multilink bundle interface that allows Frame Relay encapsulation
- Bundle identifier name
- Multilink Frame Relay subinterfaces
- Bundle interface bandwidth class
- Serial interfaces

MgmtMultilink Controller

You configure a multilink bundle under a controller, using the following commands:

controller MgmtMultilink *rack/slot/bay/controller-id*

bundle *bundleId*

This configuration creates the controller for a generic multilink bundle. The controller ID number is the zero-based index of the controller chip. Currently, the SPAs that support multilink Frame Relay have only one controller per bay; therefore, the controller ID number is always zero (0).

Multilink Bundle Interface

After you create the multilink bundle, you create a multilink bundle interface that allows Frame Relay encapsulation, using the following commands:

interface multilink *interface-path-id*

encapsulation frame-relay

This configuration allows you to create multilink Frame Relay subinterfaces under the multilink bundle interface.

Note After you set the encapsulation on a multilink bundle interface to Frame Relay, you cannot change the encapsulation if the interface has member links or any member links associated with a multilink bundle.

Bundle Identifier Name

Note Bundle identifier name is configurable only under Frame Relay Forum 16.1 (FRF 16.1).

The bundle identifier (**bid**) name value identifies the bundle interface at both endpoints of the interface. The bundle identifier name is exchanged in the information elements to ensure consistent link assignments.

By default, the interface name, for example, Multilink 0/4/1/0/1, is used as the bundle identifier name. However, you can optionally create a name using the **frame-relay multilink bid** command.

Note Regardless of whether you use the default name or create a name using the **frame-relay multilink bid** command, each bundle must have a unique name. If the same name is used by different bundles, the bundles will flap indefinitely.

The bundle identifier name can be up to 50 characters including the null termination character. The bundle identifier name is configured at the bundle interface level and is applied to each member link.

You configure the bundle identifier name using the following commands:

interface multilink *interface-path-id*

frame-relay multilink bid *bundle-id-name*

Multilink Frame Relay Subinterfaces

You configure a multilink Frame Relay subinterface, using the following command:

interface multilink *interface-path-id*[*.subinterface* {**l2transport** | **point-to-point**}]

You can configure up to 992 subinterfaces on a multilink bundle interface.

Note You configure specific Frame Relay interface features at the subinterface level.

Multilink Frame-Relay Subinterface Features

The following commands are available to set specific features on a multilink Frame Relay bundle subinterface:

- **mtu** *MTU size*
- **description**
- **shutdown**
- **bandwidth** *bandwidth*
- **service-policy** {**input** | **output**} *policymap-name*

Note When entering the **service-policy** command, which enables you to attach a policy map to a multilink Frame Relay bundle subinterface, you must do so while in Frame Relay PVC configuration mode. For more information, see Configuring Multilink Frame Relay Bundle Interfaces, page 500.

Bundle Interface Bandwidth Class

Note Bandwidth class is configurable only under a multilink bundle interface.

You can configure one of three types of bandwidth classes on a multilink Frame Relay interface:

- a—Bandwidth Class A
- b—Bandwidth Class B
- c—Bandwidth Class C

When Bandwidth Class A is configured and one or more member links are up (PH_ACTIVE), the bundle interface is also up and BL_ACTIVATE is signaled to the Frame Relay connections. When all the member links are down, the bundle interface is down and BL_DEACTIVATE is signaled to the Frame Relay connections.

When Bandwidth Class B is configured and all the member links are up (PH_ACTIVE), the bundle interface is up and BL_ACTIVATE is signaled to the Frame Relay connections. When any member link is down, the bundle interface is down and BL_ACTIVATE is signaled to the Frame Relay connections.

When Bandwidth Class C is configured, you must also set the bundle link threshold to a value between 1 and 255. The threshold value is the minimum number of links that must be up (PH_ACTIVE) for the bundle interface to be up and for BL_ACTIVATE to be signaled to the Frame Relay connections. When the number of links that are up falls below this threshold, the bundle interface goes down and BL_DEACTIVATE is signaled to the Frame Relay connections. When 1 is entered as the threshold value, the behavior is identical to Bandwidth Class A. If you enter a threshold value that is greater than the number of member links that are up, the bundle remains down.

You configure the bandwidth class for a Frame Relay multilink bundle interface using the following commands:

interface multilink *interface-path-id*

frame-relay multilink bandwidth-class {**a** | **b** | **c** [*threshold*]}

The default is a (Bandwidth Class A).

Serial Interfaces

After the T3 and T1 controllers are configured, you can add serial interfaces to the multilink Frame Relay bundle subinterface by configuring the serial interface, encapsulating it as multilink Frame Relay (mfr), assigning it to the bundle interface (specified by the multilink group number), and configuring a name for the link. You may also configure MFR acknowledge timeout value, retry count for retransmissions and hello interval, for the bundle link.

You configure a multilink Frame Relay serial interface using the following commands:

interface serial *rack/slot/module/port/t1-num:channel-group-number*

encapsulation mfr

multilink group *group number*

frame-relay multilink lid *link-id name*

frame-relay multilink ack *ack-timeout*

frame-relay multilink hello *hello-interval*

frame-relay multilink retry *retry-count*

Note All serial links in an MFR bundle inherit the value of the **mtu** command from the multilink interface. Therefore, you should not configure the **mtu** command on a serial interface before configuring it as a member of an MFR bundle. The Cisco IOS XR software blocks attempts to configure a serial interface as a member of an MFR bundle if the interface is configured with a nondefault MTU value as well as attempts to change the **mtu** command value for a serial interface that is configured as a member of an MFR bundle.

Show Commands

You can verify a multilink Frame Relay serial interface configuration using the following **show** commands:

show frame-relay multilink location *node id*

show frame-relay multilink interface serial *interface-path-id* [**detail** | **verbose**]

The following example shows the display output of the **show frame-relay multilink location** command:

```
RP/0/0/CPU0:router# show frame-relay multilink  location  0/4/cpu0
Member interface: Serial0/4/2/0/9:0, ifhandle 0x05007b00
HW state = Up, link state = Up
Member of bundle interface Multilink0/4/2/0/2 with ifhandle 0x05007800

Bundle interface: Multilink0/4/2/0/2, ifhandle 0x05007800
    Member Links: 4 active, 0 inactive
    State = Up,   BW Class = C (threshold   3)
    Member Links:
    Serial0/4/2/0/12:0, HW state = Up, link state = Up
    Serial0/4/2/0/11:0, HW state = Up, link state = Up
    Serial0/4/2/0/10:0, HW state = Up, link state = Up
    Serial0/4/2/0/9:0, HW state = Up, link state = Up

Member interface: Serial0/4/2/0/10:0, ifhandle 0x05007c00
HW state = Up, link state = Up
Member of bundle interface Multilink0/4/2/0/2 with ifhandle 0x05007800

Member interface: Serial0/4/2/0/11:0, ifhandle 0x05007d00
HW state = Up, link state = Up
Member of bundle interface Multilink0/4/2/0/2 with ifhandle 0x05007800
```

```
Member interface: Serial0/4/2/0/12:0, ifhandle 0x05007e00
HW state = Up, link state = Up
Member of bundle interface Multilink0/4/2/0/2 with ifhandle 0x05007800
```

The following example shows the display output of

```
RP/0/0/CPU0:router# show frame-relay multilink interface serial 0/4/2/0/10:0

Member interface: Serial0/4/2/0/10:0, ifhandle 0x05007c00
HW state = Up, link state = Up
Member of bundle interface Multilink0/4/2/0/2 with ifhandle 0x05007800
```

End-to-End Fragmentation (FRF.12)

You can configure an FRF.12 end-to-end fragmentation connection using the data-link connection identifier (DLCI). However, it must be done on a channelized Frame Relay serial interface.

Note The **fragment end-to-end** command is not allowed on Packet-over-SONET/SDH (POS) interfaces or under the DLCI of a multilink Frame Relay bundle interface.

You configure FRF.12 end-to-end fragmentation on a DLCI connection using the following command:

fragment end-to-end *fragment-size*

The *fragment-size* argument defines the size of the fragments, in bytes, for the serial interface.

Note On a DLCI connection, we highly recommend that you configure an egress service policy that classifies packets into high and low priorities, so that interleaving of high-priority and low-priority fragments occurs.

Configuring Frame Relay

The following sections describe how to configure Frame Relay interfaces.

- Modifying the Default Frame Relay Configuration on an Interface, page 495
- Disabling LMI on an Interface with Frame Relay Encapsulation, page 498
- Configuring Multilink Frame Relay Bundle Interfaces, page 500
- Configuring FRF.12 End-to-End Fragmentation on a Channelized Frame Relay Serial Interface, page 506

Modifying the Default Frame Relay Configuration on an Interface

Perform this task to modify the default Frame Relay parameters on a Packet-over-SONET/SDH (POS), multilink, or serial interface with Frame Relay encapsulation.

Prerequisites

Before you can modify the default Frame Relay configuration, you need to enable Frame Relay on the interface, as described in the following modules:

- To enable Frame Relay encapsulation on a POS interface, see the *"Configuring POS Interfaces on Cisco IOS XR Software"* module in this manual.
- To enable Frame Relay encapsulation on a serial interface, see the *Configuring Serial Interfaces on Cisco IOS XR Sotware* module in this manual.

Note Before enabling Frame Relay encapsulation on a POS or serial interface, make certain that you have not previously assigned an IP address to the interface. If an IP address is assigned to the interface, you will not be able to enable Frame Relay encapsulation. For Frame Relay, the IP address and subnet mask are configured on the subinterface.

Restrictions

- The LMI type must match on both ends of the connection for the connection to be active.
- Before you can remove Frame Relay encapsulation on an interface and reconfigure that interface with PPP or HDLC encapsulation, you must remove all interfaces, subinterface, LMI, and Frame Relay configuration from that interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **frame-relay intf-type** {**dce** | **dte**}
4. **frame-relay lmi-type** [**ansi** | **cisco** | **q933a**]
5. **encap** {**cisco** | **ietf**}
6. **end**
   or
   commit
7. **show interfaces** [**summary** | [*type interface-path-id*] [**brief** | **description** | **detail** | **accounting** [**rates**]]] [**location** *node-id*]

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

	Command or Action	Purpose
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface pos 0/4/0/1`	Enters interface configuration mode.
Step 3	**frame-relay intf-type** {**dce** | **dte**} **Example:** `RP/0/0/CPU0:router(config-if)# frame-relay intf-type dce`	Configures the type of support provided by the interface. • If your router functions as a switch connected to another router, use the **frame-relay intf-type dce** command to configure the LMI type to support data communication equipment (DCE). • If your router is connected to a Frame Relay network, use the **frame-relay intf-type dte** command to configure the LMI type to support data terminal equipment (DTE). **Note** The default interface type is DTE.
Step 4	**frame-relay lmi-type** [**ansi** | **q933a** | **cisco**] **Example:** `RP/0/0/CPU0:router(config-if)# frame-relay lmi-type ansi`	Selects the LMI type supported on the interface. • Enter the **frame-relay lmi-type ansi** command to use LMI as defined by ANSI T1.617a-1994 Annex D. • Enter the **frame-relay lmi-type cisco** command to use LMI as defined by Cisco (not standard). • Enter the **frame-relay lmi-type q933a** command to use LMI as defined by ITU-T Q.933 (02/2003) Annex A. **Note** The default LMI type is Cisco.
Step 5	**encap** {**cisco** | **ietf**} **Example:** `RP/0/0/CPU0:router (config-fr-vc)# encap ietf`	Configures the encapsulation for a Frame Relay PVC. **Note** If the encapsulation type is not configured explicitly for a PVC, then that PVC inherits the encapsulation type from the main interface.

	Command or Action	Purpose

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 7	**show interfaces** [**summary** | [*type interface-path-id*] [**brief** | **description** | **detail** | **accounting** [**rates**]]] [**location** *node-id*] **Example:** `RP/0/0/CPU0:router# show interface pos 0/4/0/1`	(Optional) Verifies the configuration for the specified interface.

Disabling LMI on an Interface with Frame Relay Encapsulation

Perform this task to disable LMI on interfaces that have Frame Relay encapsulation.

Note LMI is enabled by default on interfaces that have Frame Relay encapsulation enabled. To reenable LMI on an interface after it has been disabled, use the **no frame-relay lmi disable** command in interface configuration mode.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **frame-relay lmi disable**
4. **end**
   or
   commit
5. **show interfaces** [**summary** | [*type interface-path-id*] [**brief** | **description** | **detail** | **accounting** [**rates**]]] [**location** *node-id*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface POS 0/4/0/1`	Enters interface configuration mode.
Step 3	**frame-relay lmi disable** **Example:** `RP/0/0/CPU0:router(config-if)# frame-relay lmi disable`	Disables LMI on the specified interface.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**show interfaces** [**summary** | [*type interface-path-id*] [**brief** | **description** | **detail** | **accounting** [**rates**]]] [**location** *node-id*] **Example:** `RP/0/0/CPU0:router# show interfaces POS 0/1/0/0`	(Optional) Verifies that LMI is disabled on the specified interface.

Configuring Multilink Frame Relay Bundle Interfaces

Perform these steps to configure a multilink Frame Relay (MFR) bundle interface and its subinterfaces.

Restrictions

- MFR does not support the FRF.16.1 Vendor Extension Information element.
- All member links in a multilink Frame Relay bundle interface must be of the same type (for example, T1s or E1s). The member links must have the same framing type, such as point-to-point, and they must have the same bandwidth class.
- All member links must be full T1s or E1s. Fractional links, such as DS0s, are not supported.
- All member links must reside on the same SPA; otherwise, they are considered to be unrelated bundles.
- All member links must be connected to the same line card or SPA at the far end.
- A maximum of 992 MFR subinterfaces is supported on each main interface, based on the supported DLCI range 16–1007.
- A maximum of 4096 subinterfaces are supported per line card or SPA.
- The Cisco 8-Port Channelized T1/E1 SPA has the following additional guidelines:
  - A maximum of 8 T1/E1 links per SPA is supported.
  - A maximum of 8 links per bundle is supported.
  - A maximum of 8 MFR bundles per SPA is supported, with a minimum of one link per bundle.
- All serial links in an MFR bundle inherit the value of the **mtu** command from the multilink interface. Therefore, you should not configure the **mtu** command on a serial interface before configuring it as a member of an MFR bundle. The Cisco IOS XR software blocks the following:
  - Attempts to configure a serial interface as a member of an MFR bundle if the interface is configured with a nondefault MTU value.
  - Attempts to change the **mtu** command value for a serial interface that is configured as a member of an MFR bundle.

SUMMARY STEPS

1. **configure**
2. **controller MgmtMultilink** *rack/slot/bay/controller-id*
3. **exit**
4. **controller t3** *interface-path-id*
5. **mode** *type*
6. **clock source** {**internal** | **line**}
7. **exit**
8. **controller** {**t1** | **e1**} *interface-path-id*
9. **channel-group** *channel-group-number*
10. **timeslots** *range*
11. **exit**

12. **exit**
13. **interface multilink** *interface-path-id*[.*subinterface* {**l2transport** | **point-to-point**}]
14. **encapsulation frame-relay**
15. **frame-relay multilink bid** *bundle-id-name*
16. **frame-relay multilink bandwidth-class** {**a** | **b** | **c** [*threshold*]}
17. **multilink fragment-size** *size* [**fragment-counter**]
18. **exit**
19. **interface multilink** *interface-path-id*[.*subinterface* {**l2transport** | **point-to-point**}]
20. **ipv4 address** *ip-address*
21. **pvc** *dlci*
22. **service-policy** {**input** | **output**} *policy-map*
23. **exit**
24. **exit**
25. **interface serial** *interface-path-id*
26. **encapsulation mfr**
27. **multilink group** *group-id*
28. **frame-relay multilink lid** *link-id name*
29. **frame-relay multilink ack** *ack-timeout*
30. **frame-relay multilink hello** *hello-interval*
31. **frame-relay multilink retry** *retry-count*
32. **exit**
33. **end**
    or
    commit
34. **exit**
35. **show frame-relay multilink interface** *type interface-path-id* [**detail** | **verbose**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# config`	Enters global configuration mode.
Step 2	**controller MgmtMultilink** *rack/slot/bay/controller-id* **Example:** `RP/0/0/CPU0:router(config)# controller MgmtMultilink 0/1/0/0`	Creates the controller for a generic multilink bundle in the *rack/slot/bay/controller-id* notation and enters the multilink management configuration mode. The controller ID number is the zero-based index of the controller chip. Currently, the SPAs that support multilink Frame Relay have only one controller per bay; therefore, the controller ID number is always zero (0).
Step 3	**exit** **Example:** `RP/0/0/CPU0:router(config-mgmtmultilink)# exit`	Exits the multilink management configuration mode.
Step 4	**controller t3** *interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# controller t3 0/1/0/0`	Specifies the T3 controller name in the *rack/slot/module/port* notation and enters T3 configuration mode.
Step 5	**mode** *type* **Example:** `RP/0/0/CPU0:router(config-t3)# mode t1`	Configures the type of multilinks to channelize; for example, 28 T1s.
Step 6	**clock source** {**internal** | **line**} **Example:** `RP/0/0/CPU0:router(config-t3)# clock source internal`	(Optional) Sets the clocking for individual E3 links. **Note** The default clock source is **internal**. **Note** When configuring clocking on a serial link, you must configure one end to be **internal**, and the other end to be **line**. If you configure **internal** clocking on both ends of a connection, framing slips occur. If you configure **line** clocking on both ends of a connection, the line does not come up.
Step 7	**exit** **Example:** `RP/0/0/CPU0:router(config-t3)# exit`	Exits T3/E3 controller configuration mode.
Step 8	**controller** {**t1** | **e1**} *interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# controller t1 0/1/0/0/0`	Enters T1 or E1 configuration mode.

	Command or Action	Purpose
Step 9	**channel-group** *channel-group-number* **Example:** `RP/0/0/CPU0:router(config-t1)# channel-group 0`	Creates a T1 channel group and enters channel group configuration mode for that channel group.
Step 10	**timeslots** *range* **Example:** `RP/0/0/CPU0:router(config-t1-channel_group)# timeslots 1-24`	Associates one or more DS0 time slots to a channel group and creates an associated serial subinterface on that channel group. • For T1 controllers—Range is from 1 to 24 time slots. • For E1 controllers—Range is from 1 to 31 time slots. • You can assign all time slots to a single channel group, or you can divide the time slots among several channel groups.
Step 11	**exit** **Example:** `RP/0/0/CPU0:router(config-t1-channel_group)# exit`	Exits channel group configuration mode.
Step 12	**exit** **Example:** `RP/0/0/CPU0:router(config-t1)# exit`	Exits T1 configuration mode.
Step 13	**interface multilink** *interface-path-id*[.*subinterface* {**l2transport** | **point-to-point**}] **Example:** `RP/0/0/CPU0:router(config)# interface Multilink 0/1/0/0/100`	Creates a multilink bundle interface where you can specify Frame Relay encapsulation for the bundle. You create multilink Frame Relay subinterfaces under the multilink bundle interface.
Step 14	**encapsulation frame-relay** **Example:** `Router(config-if)# encapsulation frame-relay`	Specifies the Frame Relay encapsulation type.
Step 15	**frame-relay multilink bid** *bundle-id-name* **Example:** `Router(config-if)# frame-relay multilink bid MFRBundle`	(Optional) By default, the interface name, for example, Multilink 0/4/1/0/1, is used as the bundle identifier name. However, you can optionally create a name using the **frame-relay multilink bid** command. **Note** Regardless of if you use the default name or create a name using the **frame-relay multilink bid** command, each bundle must have a unique name. If the same name is used by different bundles, the bundles will flap indefinitely.

	Command or Action	Purpose
Step 16	**frame-relay multilink bandwidth-class** {**a** | **b** | **c** [*threshold*]} **Example:** `Router(config-if)# frame-relay multilink bandwidth-class a`	Configures one of three types of bandwidth classes on a multilink Frame Relay interface: • a—Bandwidth Class A • b—Bandwidth Class B • c—Bandwidth Class C The default is **a** (Bandwidth Class A).
Step 17	**multilink fragment-size** *size* [**fragment-counter**] Example: `RP/0/0/CPU0:router(config-if)# multilink fragment-size 256 fragment-counter`	(Optional) Specifies the size of the multilink fragments, and optionally enables counting of the fragmented packets. The default is no fragments.
Step 18	**exit** **Example:** `RP/0/0/CPU0:router(config-if)# exit`	Exits interface configuration mode.
Step 19	**interface multilink** *interface-path-id*[.*subinterface* {**l2transport** | **point-to-point**}] **Example:** `RP/0/0/CPU0:router(config)# interface Multilink 0/1/0/0/100.16 point-to-point`	Creates a multilink subinterface in the *rack/slot/bay/controller-id bundleId.subinterace* [**point-to-point** | **l2transport**] notation and enters the subinterface configuration mode. • **l2transport**—Treat as an attachment circuit • **point-to-point**—Treat as a point-to-point link You can configure up to 992 subinterfaces on a multilink bundle interface. The DLCIs are 16 to 1007.
Step 20	**ipv4 address** *ip-address* **Example:** `RP/0/0/CPU0:router(config-subif)# ipv4 address 3.1.100.16 255.255.255.0`	Assigns an IP address and subnet mask to the interface in the format: *A.B.C.D/prefix* or *A.B.C.D/mask*
Step 21	**pvc** *dlci* **Example:** `RP/0/0/CPU0:router (config-subif)# pvc 16`	Creates a POS permanent virtual circuit (PVC) and enters Frame Relay PVC configuration submode. Replace *dlci* with a PVC identifier, in the range from 16 to 1007. **Note** Only one PVC is allowed per subinterface.
Step 22	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/0/CPU0:router(config-fr-vc)# service-policy output policy-mapA`	Attaches a policy map to an input subinterface or output subinterface. When attached, the policy map is used as the service policy for the subinterface. **Note** For information on creating and configuring policy maps, refer to *Cisco IOS XR Modular Quality of Service Configuration Guide*.
Step 23	**exit** **Example:** `RP/0/0/CPU0:router(config-fr-vc)# exit`	Exits the Frame-Relay virtual circuit mode.

	Command or Action	Purpose
Step 24	**exit** **Example:** `RP/0/0/CPU0:router(config-subif)# exit`	Exits the subinterface configuration mode.
Step 25	**interface serial** *interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface serial 0/1/0/0/0/0:0`	Specifies the complete interface number with the *rack/slot/module/port/T3Num/T1num:instance* notation.
Step 26	**encapsulation mfr** **Example:** `RP/0/0/CPU0:router(config)# encapsulation mfr`	Enables multilink Frame Relay on the serial interface.
Step 27	**multilink group** *group-id* **Example:** `RP/0/0/CPU0:router(config-if)# multilink group 100`	Specifies the multilink group ID for this interface.
Step 28	**frame-relay multilink lid** *link-id name* **Example:** `RP/0/0/CPU0:router(config-if)# frame-relay multilink lid sj1`	Configures a name for the Frame Relay multilink bundle link. **Note** Each link within a bundle must have a unique name. If the same name is used by different links in the same bundle, the bundles will flap indefinitely.
Step 29	**frame-relay multilink ack** *ack-timeout* **Example:** `RP/0/0/CPU0:router(config-if)# frame-relay multilink ack 5`	Configures the acknowledge timeout value for the Frame Relay multilink bundle link.
Step 30	**frame-relay multilink hello** *hello-interval* **Example:** `RP/0/0/CPU0:router(config-if)# frame-relay multilink hello 60`	Configures the hello interval for the Frame Relay multilink bundle link.
Step 31	**frame-relay multilink retry** *retry-count* **Example:** `RP/0/0/CPU0:router(config-if)# frame-relay multilink retry 2`	Configures the retry count for retransmissions for the Frame Relay multilink bundle link.
Step 32	**exit** **Example:** `RP/0/0/CPU0:router(config-if)# exit`	Exits interface configuration mode.

	Command or Action	Purpose

Step 33	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 34	**exit** **Example:** `RP/0/0/CPU0:router(config)# exit`	Exits global configuration mode.
Step 35	**show frame-relay multilink interface** *type interface-path-id* [**detail** | **verbose**] **Example:** `RP/0/0/CPU0:router# show frame-relay multilink interface Multilink 0/5/1/0/1`	Shows the information retrieved from the interface description block (IDB), including bundle-specific information and Frame Relay information.

Configuring FRF.12 End-to-End Fragmentation on a Channelized Frame Relay Serial Interface

Perform the following steps to configure FRF.12 end-to-end fragmentation on a channelized Frame Relay serial interface.

Restrictions

The Cisco 8-Port Channelized T1/E1 SPA on the Cisco XR 12000 Series Router supports fragment sizes of 128, 512, and 256 bytes only.

SUMMARY STEPS

1. **config**
2. **controller t3** *interface-path-id*
3. **mode** *type*
4. **clock source** {**internal** | **line**}
5. **exit**
6. **controller t1** *interface-path-id*
7. **channel-group** *channel-group-number*
8. **timeslots** *range*
9. **exit**
10. **exit**
11. **interface serial** *interface-path-id*
12. **encapsulation frame-relay**
13. **exit**
14. **interface serial** *interface-path-id*
15. **ipv4 address** *ip-address*
16. **pvc** *dlci*
17. **service-policy** {**input** | **output**} *policy-map*
18. **fragment end-to-end** *fragment-size*
19. **fragment-counter**
20. **exit**
21. **exit**
22. **exit**
23. **end**
    or
    commit
24. **exit**
25. **show frame-relay pvc** [**dlci** | **interface** | **location**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**config** **Example:** `RP/0/0/CPU0:router# config`	Enters global configuration mode.
Step 2	**controller t3** *interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# controller t3 0/1/0/0`	Specifies the T3 controller name in the *rack/slot/module/port* notation and enters T3 configuration mode.
Step 3	**mode** *type* **Example:** `RP/0/0/CPU0:router(config-t3)# mode t1`	Configures the type of multilinks to channelize; for example, 28 T1s.
Step 4	**clock source** {**internal** | **line**} **Example:** `RP/0/0/CPU0:router(config-t3)# clock source internal`	(Optional) Sets the clocking for individual E3 links. **Note** The default clock source is **internal**. **Note** When configuring clocking on a serial link, you must configure one end to be **internal**, and the other end to be **line**. If you configure **internal** clocking on both ends of a connection, framing slips occur. If you configure **line** clocking on both ends of a connection, the line does not come up.
Step 5	**exit** **Example:** `RP/0/0/CPU0:router(config-t3)# exit`	Exits T3/E3 or T1/E1 controller configuration mode.
Step 6	**controller t1** *interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# controller t1 0/1/0/0/0`	Enters T1 configuration mode.
Step 7	**channel-group** *channel-group-number* **Example:** `RP/0/0/CPU0:router(config-t1)# channel-group 0`	Creates a T1 channel group and enters channel group configuration mode for that channel group.

	Command or Action	Purpose
Step 8	**timeslots** *range* **Example:** `RP/0/0/CPU0:router(config-t1-channel_group)# timeslots 1-24`	Associates one or more DS0 time slots to a channel group and creates an associated serial subinterface on that channel group. • Range is from 1 to 24 time slots. • You can assign all 24 time slots to a single channel group, or you can divide the time slots among several channel groups. **Note** Each individual T1 controller supports a total of 24 DS0 time slots.
Step 9	**exit** **Example:** `RP/0/0/CPU0:router(config-t1-channel_group)# exit`	Exits channel group configuration mode.
Step 10	**exit** **Example:** `RP/0/0/CPU0:router(config-t1)# exit`	Exits T1 configuration mode.
Step 11	**interface serial** *interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface serial 0/1/0/0/0/0:0`	Specifies the complete interface number with the *rack/slot/module/port/T3Num/T1num:instance* notation.
Step 12	**encapsulation frame-relay** **Example:** `RP/0/0/CPU0:Router(config-if)# encapsulation frame-relay`	Specifies the Frame Relay encapsulation type.
Step 13	**exit** **Example:** `RP/0/0/CPU0:router(config-if)# exit`	Exits interface configuration mode.
Step 14	**interface serial** *interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface serial 1/0/0/0/0:0.1`	Specifies the complete subinterface number with the *rack*/*slot*/*module*/*port*[/*channel-num:channel-group-number*].*subinterface* notation.
Step 15	**ipv4 address** *ip-address* **Example:** `RP/0/0/CPU0:router(config-subif)# ipv4 address 3.1.100.16 255.255.255.0`	Assigns an IP address and subnet mask to the interface in the format: *A.B.C.D/prefix* or *A.B.C.D/mask*

	Command or Action	Purpose
Step 16	**pvc** *dlci* **Example:** `RP/0/0/CPU0:router (config-subif)# pvc 100`	Creates a POS permanent virtual circuit (PVC) and enters Frame Relay PVC configuration submode. Replace *dlci* with a PVC identifier, in the range from 16 to 1007. **Note** Only one PVC is allowed per subinterface.
Step 17	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/0/CPU0:router(config-fr-vc)# service-policy output policy-mapA`	Attaches a policy map to an input subinterface or output subinterface. When attached, the policy map is used as the service policy for the subinterface. **Note** For effective FRF.12 functionality (interleave specifically), you should configure an egress service policy with priority. **Note** For information on creating and configuring policy maps, refer to *Cisco IOS XR Modular Quality of Service Configuration Guide*,
Step 18	**fragment end-to-end** *fragment-size* **Example:** `RP/0/0/CPU0:router(config-fr-vc)# fragment end-to-end 100`	(Optional) Enables fragmentation of Frame Relay frames on an interface and specifies the size (in bytes) of the payload from the original frame that will go into each fragment. This number excludes the Frame Relay header of the original frame. Valid values are from 64 to 512, depending on your hardware.
Step 19	**fragment-counter** **Example:** `RP/0/0/CPU0:router(config-fr-vc)# fragment-counter`	(Optional) Enables fragmentation counters for a Frame Relay subinterface and PVC.
Step 20	**exit** **Example:** `RP/0/0/CPU0:router(config-fr-vc)# exit`	Exits the Frame-Relay virtual circuit mode.
Step 21	**exit** **Example:** `RP/0/0/CPU0:router(config-subif)# exit`	Exits the subinterface configuration mode.
Step 22	**exit** **Example:** `RP/0/0/CPU0:router(config-if)# exit`	Exits interface configuration mode.

	Command or Action	Purpose

Step 23	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 24	**exit** **Example:** `RP/0/0/CPU0:router(config)# exit`	Exits global configuration mode.
Step 25	**show frame-relay pvc** [**dlci** | **interface** | **location**] **Example:** `RP/0/0/CPU0:router# show frame-relay pvc 100`	Displays the information for the specified PVC DLCI, interface, or location.

Configuration Examples for Frame Relay

This section provides the following configuration examples:

- Optional Frame Relay Parameters: Example, page 512
- Multilink Frame Relay: Example, page 514
- End-to-End Fragmentation: Example, page 515

Optional Frame Relay Parameters: Example

The following example shows how to bring up and configure a POS interface with Frame Relay encapsulation. In this example, the user modifies the default Frame Relay configuration so that the interface supports ANSI T1.617a-1994 Annex D LMI on DCE.

```
RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router(config)# interface POS 0/3/0/0
RP/0/0/CPU0:router(config-if)# encapsulation frame-relay IETF
RP/0/0/CPU0:router(config-if)# frame-relay intf-type dce
RP/0/0/CPU0:router(config-if)# frame-relay lmi-type ansi
RP/0/0/CPU0:router(config-if)# no shutdown
RP/0/0/CPU0:router(config-if)# end

Uncommitted changes found, commit them? [yes]: yes

RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router (config)# interface pos 0/3/0/0.10 point-to-point
RP/0/0/CPU0:router (config-subif)#ipv4 address 10.46.8.6/24
RP/0/0/CPU0:router (config-subif)# pvc 20
RP/0/0/CPU0:router (config-fr-vc)# encap ietf
RP/0/0/CPU0:router(config-subif)# commit
```

The following example shows how to disable LMI on a POS interface that has Frame Relay encapsulation configured:

```
RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router(config)# interface
RP/0/0/CPU0:router(config)# interface pos 0/3/0/0
RP/0/0/CPU0:router(config-if)# frame-relay lmi disable
RP/0/0/CPU0:router(config-if)# end

Uncommitted changes found, commit them? [yes]: yes
```

The following example shows how to reenable LMI on a serial interface:

```
RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router(config)# interface
RP/0/0/CPU0:router(config)# interface serial 0/3/0/0
RP/0/0/CPU0:router(config-if)# no frame-relay lmi disable
RP/0/0/CPU0:router(config-if)# end

Uncommitted changes found, commit them? [yes]: yes
```

The following example shows how to display Frame Relay statistics for LMI on all interfaces:

```
RP/0/0/CPU0:router# show frame-relay lmi

LMI Statistics for interface POS0/1/0/0/ (Frame Relay DCE) LMI TYPE = ANSI
Invalid Unnumbered Info 0             Invalid Prot Disc 0
Invalid Dummy Call Ref 0              Invalid Msg Type 0
Invalid Status Message 0              Invalid Lock Shift 9
Invalid Information ID 0              Invalid Report IE Len 0
Invalid Report Request 0              Invalid Keep IE Len 0
Num Status Enq. Rcvd 9444             Num Status Msgs Sent 9444
Num Full Status Sent 1578             Num St Enq. Timeouts 41
Num Link Timeouts 7

LMI Statistics for interface POS0/1/0/1/ (Frame Relay DCE) LMI TYPE = CISCO
Invalid Unnumbered Info 0             Invalid Prot Disc 0
Invalid Dummy Call Ref 0              Invalid Msg Type 0
Invalid Status Message 0              Invalid Lock Shift 0
Invalid Information ID 0              Invalid Report IE Len 0
```

```
Invalid Report Request 0              Invalid Keep IE Len 0
Num Status Enq. Rcvd 9481             Num Status Msgs Sent 9481
Num Full Status Sent 1588             Num St Enq. Timeouts 16
Num Link Timeouts 4
```

The following example shows how to create a serial subinterface with a PVC on the main serial interface:

```
RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router(config)# interface serial 0/3/0/0/0:0.10 point-to-point
RP/0/0/CPU0:router (config-subif)#ipv4 address 10.46.8.6/24
RP/0/0/CPU0:router (config-subif)# pvc 20
RP/0/0/CPU0:router (config-fr-vc)# encapsulation ietf
RP/0/0/CPU0:router(config-subif)# commit
```

The following example shows how to display information about all PVCs configured on your system:

```
RP/0/0/CPU0router# show frame-relay pvc

PVC Statistics for interface Serial0/3/2/0 (Frame Relay DCE)

              Active     Inactive      Deleted       Static
  Local          4           0            0            0
  Switched       0           0            0            0
  Dynamic        0           0            0            0

DLCI = 612, DLCI USAGE = LOCAL, ENCAP = CISCO, INHERIT = TRUE, PVC STATUS = ACTI
VE, INTERFACE = Serial0/3/2/0.1
  input pkts 0          output pkts 0          in bytes 0
  out bytes 0           dropped pkts 0         in FECN packets 0
  in BECN pkts 0        out FECN pkts 0        out BECN pkts 0
  in DE pkts 0          out DE pkts 0
  out bcast pkts 0      out bcast bytes 0
  pvc create time 00:00:00      last time pvc status changed 00:00:00

DLCI = 613, DLCI USAGE = LOCAL, ENCAP = CISCO, INHERIT = TRUE, PVC STATUS = ACTI
VE, INTERFACE = Serial0/3/2/0.2
  input pkts 0          output pkts 0          in bytes 0
  out bytes 0           dropped pkts 0         in FECN packets 0
  in BECN pkts 0        out FECN pkts 0        out BECN pkts 0
  in DE pkts 0          out DE pkts 0
  out bcast pkts 0      out bcast bytes 0
  pvc create time 00:00:00      last time pvc status changed 00:00:00

DLCI = 614, DLCI USAGE = LOCAL, ENCAP = CISCO, INHERIT = TRUE, PVC STATUS = ACTI
VE, INTERFACE = Serial0/3/2/0.3
  input pkts 0          output pkts 0          in bytes 0
  out bytes 0           dropped pkts 0         in FECN packets 0
  in BECN pkts 0        out FECN pkts 0        out BECN pkts 0
  in DE pkts 0          out DE pkts 0
  out bcast pkts 0      out bcast bytes 0
  pvc create time 00:00:00      last time pvc status changed 00:00:00

DLCI = 615, DLCI USAGE = LOCAL, ENCAP = CISCO, INHERIT = TRUE, PVC STATUS = ACTI
VE, INTERFACE = Serial0/3/2/0.4
  input pkts 0          output pkts 0          in bytes 0
  out bytes 0           dropped pkts 0         in FECN packets 0
  in BECN pkts 0        out FECN pkts 0        out BECN pkts 0
  in DE pkts 0          out DE pkts 0
  out bcast pkts 0      out bcast bytes 0
  pvc create time 00:00:00      last time pvc status changed 00:00:00
```

The following example shows how to modify LMI polling options on PVCs configured for a DTE, and then use the **show frame-relay lmi** and **show frame-relay lmi-info** commands to display information for monitoring and troubleshooting the interface:

```
RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router(config)# interface pos 0/3/0/0
RP/0/0/CPU0:router(config-if)# frame-relay lmi-n391dte 10
RP/0/0/CPU0:router(config-if)# frame-relay lmi-n391dte 5
RP/0/0/CPU0:router(config-if)# frame-relay lmi-t391dte 15
RP/0/0/CPU0:router(config-subif)# commit

RP/0/0/CPU0:router# show frame-relay lmi interface pos 0/3/0/0

LMI Statistics for interface pos 0/3/0/0 (Frame Relay DTE) LMI TYPE = ANSI
Invalid Unnumbered Info 0             Invalid Prot Disc 0
Invalid Dummy Call Ref 0              Invalid Msg Type 0
Invalid Status Message 0              Invalid Lock Shift 9
Invalid Information ID 0              Invalid Report IE Len 0
Invalid Report Request 0              Invalid Keep IE Len 0
Num Status Enq. Rcvd 9444             Num Status Msgs Sent 9444
Num Full Status Sent 1578             Num St Enq. Timeouts 41
Num Link Timeouts 7

RP/0/0/CPU0:router# show frame-relay lmi-info interface pos 0/3/0/0

LMI IDB Info for interface POS0/3/0/0
  ifhandle:             0x6176840
  Interface type:       DTE
  Interface state:      UP
  Line Protocol:        UP
  LMI type (cnf/oper):  AUTO/CISCO
  LMI type autosense:   OFF
  Interface MTU:        1504
  -------------- DTE -------------
  T391:                 15s
  N391: (cnf/oper):     5/5
  N392: (cnf/oper):     3/0
  N393:                 4
  My seq#:              83
  My seq# seen:         83
  Your seq# seen:       82
  -------------- DCE -------------
  T392:                 15s
  N392: (cnf/oper):     3/0
  N393:                 4
  My seq#:              0
  My seq# seen:         0
  Your seq# seen:       0
```

Multilink Frame Relay: Example

The following example shows how to configure multilink Frame Relay with serial interfaces:

```
RP/0/0/CPU0:router# config
RP/0/0/CPU0:router(config)# controller MgmtMultilink 0/3/1/0
RP/0/0/CPU0:router(config-mgmtmultilink)# bundle 100
RP/0/0/CPU0:router(config-mgmtmultilink)# exit

RP/0/0/CPU0:router(config)# controller T3 0/3/1/0
RP/0/0/CPU0:router(config-t3)# mode t1
RP/0/0/CPU0:router(config-t3)# clock source internal
RP/0/0/CPU0:router(config-t3)# exit
```

```
RP/0/0/CPU0:router(config)# controller T1 0/3/1/0/0
RP/0/0/CPU0:router(config-t1)# channel-group 0
RP/0/0/CPU0:router(config-t1-channel_group)# timeslots 1-24
RP/0/0/CPU0:router(config-t1-channel_group)#  exit
RP/0/0/CPU0:router(config-t1-channel_group)# exit
RP/0/0/CPU0:router(config-t1)# exit

RP/0/0/CPU0:router(config)# interface Multilink 0/3/1/0/100
RP/0/0/CPU0:router(config-if)# encapsulation frame-relay
RP/0/0/CPU0:router(config-if)# exit

RP/0/0/CPU0:router(config)# interface Multilink 0/3/1/0/100.16 point-to-point
RP/0/0/CPU0:router(config-subif)# ipv4 address 3.1.100.16 255.255.255.0
RP/0/0/CPU0:router(config-subif)# pvc 16
RP/0/0/CPU0:router(config-fr-vc)# service-policy output policy-mapA
RP/0/0/CPU0:router(config-fr-vc)# exit
RP/0/0/CPU0:router(config-subif)# exit

RP/0/0/CPU0:router(config)# interface Serial 0/3/1/0/0:0
RP/0/0/CPU0:router(config-if)# encapsulation mfr
RP/0/0/CPU0:router(config-if)# multilink group 100
RP/0/0/CPU0:router(config-if)# frame-relay multilink lid sj1
RP/0/0/CPU0:router(config-if)# frame-relay multilink ack 5
RP/0/0/CPU0:router(config-if)# frame-relay multilink hello 60
RP/0/0/CPU0:router(config-if)# frame-relay multilink retry 2
RP/0/0/CPU0:router(config-if)# exit
RP/0/0/CPU0:router(config)#
```

End-to-End Fragmentation: Example

The following example shows how to configure FRF.12 end-to-end fragmentation on a channelized Frame Relay serial interface:

```
RP/0/0/CPU0:router# config
RP/0/0/CPU0:router(config)# controller T30/3/1/0
RP/0/0/CPU0:router(config-t3)# mode t1
RP/0/0/CPU0:router(config-t3)# clock source internal
RP/0/0/CPU0:router(config-t3)# exit
RP/0/0/CPU0:router(config-t3)# controller T10/3/1/0/0
RP/0/0/CPU0:router(config-t1)# channel-group 0
RP/0/0/CPU0:router(config-t1-channel_group)#  timeslots 1-24
RP/0/0/CPU0:router(config-t1-channel_group)# exit
RP/0/0/CPU0:router(config-t1-channel_group)# interface Serial 0/3/1/0/0:0
RP/0/0/CPU0:router(config-if)# encapsulation frame-relay
RP/0/0/CPU0:router(config-if)# exit
RP/0/0/CPU0:router(config-if)# interface Serial 0/3/1/0/0:0.100 point-to-point
RP/0/0/CPU0:router(config-subif)# ipv4 address 3.1.1.1 255.255.255.0
RP/0/0/CPU0:router(config-subif)#  pvc 100
RP/0/0/CPU0:router(config-fr-vc)#  service-policy output LFI
RP/0/0/CPU0:router(config-fr-vc)# fragment end-to-end 256
RP/0/0/CPU0:router(config-fr-vc-frag)# fragment-counter
```

Additional References

These sections provide references related to Frame Relay.

Related Documents

Related Topic	Document Title
Cisco IOS XR master command reference	*Cisco IOS XR Master Commands List*
Cisco IOS XR interface configuration commands	*Cisco IOS XR Interface and Hardware Component Command Reference*
Initial system bootup and configuration information for a router using Cisco IOS XR software	*Cisco IOS XR Getting Started Guide*
Cisco IOS XR AAA services configuration information	*Cisco IOS XR System Security Configuration Guide* and *Cisco IOS XR System Security Command Reference*

Standards

Standards	Title
FRF.12	*Frame Relay Forum .12*
FRF.16	*Frame Relay Forum .16*
ANSI T1.617 Annex D	*American National Standards Institute T1.617 Annex D*
ITU Q.933 Annex A	*International Telecommunication Union Q.933 Annex A*

MIBs

MIBs	MIBs Link
FRF.16 MIB Cisco Frame Relay MIB IF-MIB Management Information Base for Frame Relay DTEs Management Information Base for Frame Relay DTEs Using SMIv2	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
RFC 1294	*Multiprotocol Interconnect Over Frame Relay*
RFC 1315	*Management Information Base for Frame Relay DTEs*
RFC 1490	*Multiprotocol Interconnect Over Frame Relay*
RFC 1586	*Guidelines for Running OSPF Over Frame Relay Networks*
RFC 1604	*Definitions of Managed Objects for Frame Relay Service*
RFC 2115	*Management Information Base for Frame Relay DTEs Using SMIv2*
RFC 2390	*Inverse Address Resolution Protocol*

RFCs	Title
RFC 2427	*Multiprotocol Interconnect Over Frame Relay*
RFC 2954	*Definitions of Managed Objects for Frame Relay Service*
RFC 3020	*RFC for FRF.16 MIB*

Technical Assistance

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