



Configuring Frame Relay

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 4.0.0	<p>Support for Frame Relay was added for the following SPAs:</p> <ul style="list-style-type: none"> • Cisco 2-Port Channelized OC-12c/DS0 SPA • Cisco 1-Port Channelized OC-48/STM-16 SPA • Cisco 8-Port OC-12c/STM-4 POS SPA • Cisco 2-Port OC-48c/STM-16 POS/RPR SPA • Cisco 1-Port OC-192c/STM-64 POS/RPR XFP SPA <p>Support for the following Frame Relay features was added for the Cisco 2-Port Channelized OC-12c/DSO SPA:</p> <ul style="list-style-type: none"> • Multilink Frame Relay (FRF.16) • End-to-End Fragmentation (FRF.12)
Release 4.0.1	<p>Support for Frame Relay was added for the following SPAs:</p> <ul style="list-style-type: none"> • Cisco 1-Port Channelized OC-3/STM-1 SPA • Cisco 2-Port and 4-Port Clear Channel T3/E3 SPA • Cisco 4-Port OC-3c/STM-1 POS SPA • Cisco 8-Port OC-3c/STM-1 POS SPA

Release 4.1.0	<p>Support for Frame Relay was added for the following SPAs:</p> <ul style="list-style-type: none"> • Cisco 4-Port Channelized T3/DS0 SPA • Cisco 8-Port Channelized T1/E1 SPA
---------------	--

- [Configuring Frame Relay, on page 2](#)
- [Prerequisites for Configuring Frame Relay, on page 3](#)
- [Information About Frame Relay Interfaces, on page 3](#)
- [Configuring Frame Relay, on page 11](#)
- [Configuration Examples for Frame Relay, on page 27](#)

Configuring Frame Relay

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 4.0.0	<p>Support for Frame Relay was added for the following SPAs:</p> <ul style="list-style-type: none"> • Cisco 2-Port Channelized OC-12c/DS0 SPA • Cisco 1-Port Channelized OC-48/STM-16 SPA • Cisco 8-Port OC-12c/STM-4 POS SPA • Cisco 2-Port OC-48c/STM-16 POS/RPR SPA • Cisco 1-Port OC-192c/STM-64 POS/RPR XFP SPA <p>Support for the following Frame Relay features was added for the Cisco 2-Port Channelized OC-12c/DSO SPA:</p> <ul style="list-style-type: none"> • Multilink Frame Relay (FRF.16) • End-to-End Fragmentation (FRF.12)

Release 4.0.1	<p>Support for Frame Relay was added for the following SPAs:</p> <ul style="list-style-type: none"> • Cisco 1-Port Channelized OC-3/STM-1 SPA • Cisco 2-Port and 4-Port Clear Channel T3/E3 SPA • Cisco 4-Port OC-3c/STM-1 POS SPA • Cisco 8-Port OC-3c/STM-1 POS SPA
Release 4.1.0	<p>Support for Frame Relay was added for the following SPAs:</p> <ul style="list-style-type: none"> • Cisco 4-Port Channelized T3/DS0 SPA • Cisco 8-Port Channelized T1/E1 SPA

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, ensure that these 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](#).
 - To enable Frame Relay encapsulation on a POS interface, see the module in this manual.
 - To enable Frame Relay encapsulation on a serial interface, see the module in this manual.

Information About Frame Relay Interfaces

These sections explain the various aspects of configuring Frame Relay interfaces:

Frame Relay Encapsulation

On the Cisco ASR 9000 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 module in this manual.
- To enable Frame Relay encapsulation on a serial interface, see the 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 1: Frame Relay Encapsulation Default Settings](#) are present. These default settings can be changed by configuration as described in this module.

Table 1: 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

Parameter	Configuration File Entry	Default Settings	Command Mode
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 reenabling 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 2 on page 667](#) and [Table 2: LMI Polling Configuration Commands for DTE](#).

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:

$$(MTU - 13) / 8 = \text{maximum number of PVCs}$$

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.

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

LMI Polling Configuration Commands for DCE

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

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

Table 2: 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.	<code>lmi-n391dte <i>polling-cycles</i></code>	6
Sets the error threshold.	<code>lmi-n392dte <i>threshold</i></code>	3
Sets the monitored event count.	<code>lmi-n393dte <i>events</i></code>	4
Sets the polling interval (in seconds) between each status inquiry from the DTE end.	<code>frame-relay lmi-t391dte <i>seconds</i></code>	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
- Cisco 2-Port Channelized OC-12c/DSO 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 switch processor (RSP) 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, it is recommended that each bundle have a unique name.

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} *policy-map-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](#).

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_DEACTIVATE 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/RSP0/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/RSP0/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

These sections describe how to configure Frame Relay interfaces.

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.

Before you begin

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* module in this manual.
- To enable Frame Relay encapsulation on a serial interface, see the *Configuring Serial Interfaces* 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 | q933a | cisco]
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/RSP0/CPU0:router# configure	Enters global configuration mode.

	Command or Action	Purpose
Step 2	<p>interface <i>type interface-path-id</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config)# interface pos 0/4/0/1</pre>	Enters interface configuration mode.
Step 3	<p>frame-relay intf-type {dce dte}</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-if)# frame-relay intf-type dce</pre>	<p>Configures the type of support provided by the interface.</p> <ul style="list-style-type: none"> 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). <p>Note</p> <ul style="list-style-type: none"> The default interface type is DTE.
Step 4	<p>frame-relay lmi-type [ansi q933a cisco]</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-if)# frame-relay lmi-type ansi</pre>	<p>Selects the LMI type supported on the interface.</p> <ul style="list-style-type: none"> 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. <p>Note</p> <ul style="list-style-type: none"> The default LMI type is Cisco.
Step 5	<p>encap {cisco ietf}</p> <p>Example:</p> <pre>RP/0/RSP0RP0/CPU0:router (config-fr-vc)# encap ietf</pre>	<p>Configures the encapsulation for a Frame Relay PVC.</p> <p>Note</p> <ul style="list-style-type: none"> If the encapsulation type is not configured explicitly for a PVC, then that PVC inherits the encapsulation type from the main interface.
Step 6	<p>end or commit</p> <p>Example:</p> <pre>RP/0/RSP0RP0/CPU0:router(config-if)# end or RP/0/RSP0RP0/CPU0:router(config-if)# commit</pre>	<p>Saves configuration changes.</p> <ul style="list-style-type: none"> When you issue the end command, the system prompts you to commit changes: <pre>Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:</pre>

	Command or Action	Purpose
		<ul style="list-style-type: none"> • 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	<p>show interfaces [summary [<i>type interface-path-id</i>] [brief description detail accounting [rates]]] [location node-id]</p> <p>Example:</p> <pre>RP/0/RSP0RP0/CPU0:router# show interface pos 0/4/0/1</pre>	(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	<p>configure</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router# configure</pre>	Enters global configuration mode.
Step 2	<p>interface <i>type interface-path-id</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config)# interface POS 0/4/0/1</pre>	Enters interface configuration mode.
Step 3	<p>frame-relay lmi disable</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-if)# frame-relay lmi disable</pre>	Disables LMI on the specified interface.
Step 4	<p>end or commit</p> <p>Example:</p> <pre>RP/0/RSP0RP0/CPU0:router(config-if)# end</pre> <p>or</p> <pre>RP/0/RSP0RP0/CPU0:router(config-if)# commit</pre>	<p>Saves configuration changes.</p> <ul style="list-style-type: none"> When you issue the end command, the system prompts you to commit changes: <pre>Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:</pre> 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	<p>show interfaces [summary [<i>type interface-path-id</i>] [brief description detail accounting [rates]]] [location node-id]</p> <p>Example:</p> <pre>RP/0/RSP0RP0/CPU0:router# show interfaces POS 0/1/0/0</pre>	(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.

Before you begin

Before configuring MFR bundles, be sure you have the following SPA installed:

- 1-Port Channelized STM-1/OC-3 SPA
- 2-Port Channelized OC-12c/DS0 SPA

Restrictions

- 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.
- The Cisco 1-Port Channelized OC-3/STM-1 SPA and 2-Port Channelized OC-12c/DS0 SPA have the following additional guidelines:
 - A maximum of 700 MFR bundles per line card is supported.
 - A maximum of 2600 MFR bundles per system is supported.
 - A maximum of 4000 Frame Relay Layer 3 subinterfaces per line card is supported.
 - A maximum of 8000 Frame Relay Layer 3 subinterfaces per system is supported.
- Fragmentation on a Frame Relay subinterface that is part of an MLFR bundle is not supported.
- 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* Router(config-if)# **frame-relay multilink bid** MFRBundle
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** *dldi*
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/RSP0/CPU0:router# config	Enters global configuration mode.
Step 2	controller MgmtMultilink <i>rack/slot/bay/controller-id</i> Example: RP/0/RSP0/CPU0:router(config)# controller MgmtMultilink 0/1/0/0	Creates the controller for a generic multilink bundle in the <i>rack/slot/bay/controller-id</i> 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

	Command or Action	Purpose
		have only one controller per bay; therefore, the controller ID number is always zero (0).
Step 3	exit Example: <pre>RP/0/RSP0/CPU0:router(config-mgmtmultilink)# exit</pre>	Exits the multilink management configuration mode.
Step 4	controller t3 interface-path-id Example: <pre>RP/0/RSP0/CPU0:router(config)# controller t3 0/1/0/0</pre>	Specifies the T3 controller name in the <i>rack/slot/module/port</i> notation and enters T3 configuration mode.
Step 5	mode type Example: <pre>RP/0/RSP0/CPU0:router(config-t3)# mode t1</pre>	Configures the type of multilinks to channelize; for example, 28 T1s.
Step 6	clock source {internal line} Example: <pre>RP/0/RSP0/CPU0:router(config-t3)# clock source internal</pre>	(Optional) Sets the clocking for individual E3 links. Note <ul style="list-style-type: none"> • The default clock source is internal. • 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: <pre>RP/0/RSP0/CPU0:router(config-t3)# exit</pre>	Exits T3/E3 controller configuration mode.
Step 8	controller {t1 e1} interface-path-id Example: <pre>RP/0/RSP0/CPU0:router(config)# controller t1 0/1/0/0/0</pre>	Enters T1 or E1 configuration mode.
Step 9	channel-group channel-group-number Example: <pre>RP/0/RSP0/CPU0:router(config-t1)# channel-group 0</pre>	Creates a T1 channel group and enters channel group configuration mode for that channel group.

	Command or Action	Purpose
Step 10	<p>timeslots <i>range</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-t1-channel_group)# timeslots 1-24</pre>	<p>Associates one or more DS0 time slots to a channel group and creates an associated serial subinterface on that channel group.</p> <ul style="list-style-type: none"> • 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	<p>exit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-t1-channel_group)# exit</pre>	Exits channel group configuration mode.
Step 12	<p>exit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-t1)# exit</pre>	Exits T1 configuration mode.
Step 13	<p>interface multilink <i>interface-path-id</i> [<i>.subinterface</i> {l2transport point-to-point}]</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config)# interface Multilink 0/1/0/0/100</pre>	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	<p>encapsulation frame-relay</p> <p>Example:</p> <pre>Router(config-if)# encapsulation frame-relay</pre>	Specifies the Frame Relay encapsulation type.
Step 15	<p>frame-relay multilink bid <i>bundle-id-name</i></p> <pre>Router(config-if)# frame-relay multilink bid MFRBundle</pre>	(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.
Step 16	<p>frame-relay multilink bandwidth-class {a b c [<i>threshold</i>]}</p> <p>Example:</p> <pre>Router(config-if)# frame-relay multilink bandwidth-class a</pre>	<p>Configures one of three types of bandwidth classes on a multilink Frame Relay interface:</p> <ul style="list-style-type: none"> • a—Bandwidth Class A • b—Bandwidth Class B • c—Bandwidth Class C <p>The default is a (Bandwidth Class A).</p>

	Command or Action	Purpose
Step 17	multilink fragment-size <i>size</i> [fragment-counter] Example: <pre>RP/0/0/CPU0:router(config-if)# multilink fragment-size 256 fragment-counter</pre>	(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: <pre>RP/0/RSP0/CPU0:router(config-if)# exit</pre>	Exits interface configuration mode.
Step 19	interface multilink <i>interface-path-id</i> [<i>.subinterface</i> { l2transport point-to-point }] Example: <pre>RP/0/RSP0/CPU0:router(config)# interface Multilink 0/1/0/0/100.16 point-to-point</pre>	<p>Creates a multilink subinterface in the <i>rack/slot/bay/controller-id bundleId.subinterface</i> [point-to-point l2transport] notation and enters the subinterface configuration mode.</p> <ul style="list-style-type: none"> • l2transport—Treat as an attachment circuit • point-to-point—Treat as a point-to-point link <p>You can configure up to 992 subinterfaces on a multilink bundle interface. The DLCIs are 16 to 1007.</p>
Step 20	ipv4 address <i>ip-address</i> Example: <pre>RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 3.1.100.16 255.255.255.0</pre>	<p>Assigns an IP address and subnet mask to the interface in the format:</p> <p><i>A.B.C.D/prefix</i> or <i>A.B.C.D/mask</i></p>
Step 21	pvc <i>dldci</i> Example: <pre>RP/0/RSP0/CPU0:router (config-subif)# pvc 16</pre>	<p>Creates a POS permanent virtual circuit (PVC) and enters Frame Relay PVC configuration submode.</p> <p>Replace <i>dldci</i> with a PVC identifier, in the range from 16 to 1007.</p> <p>Note • Only one PVC is allowed per subinterface.</p>
Step 22	service-policy { input output } <i>policy-map</i> Example: <pre>RP/0/RSP0/CPU0:router(config-fr-vc)# service-policy output policy-mapA</pre>	<p>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.</p> <p>Note • For information on creating and configuring policy maps, refer to <i>Cisco IOS XR Modular Quality of Service Configuration Guide</i>.</p>
Step 23	exit Example: <pre>RP/0/RSP0/CPU0:router(config-fr-vc)# exit</pre>	Exits the Frame-Relay virtual circuit mode.

	Command or Action	Purpose
Step 24	exit Example: RP/0/RSP0/CPU0:router(config-subif)# exit	Exits the subinterface configuration mode.
Step 25	interface serial <i>interface-path-id</i> Example: RP/0/RSP0/CPU0:router(config)# interface serial 0/1/0/0/0/0:0	Specifies the complete interface number with the <i>rack/slot/module/port/T3Num/T1num:instance</i> notation.
Step 26	encapsulation mfr Example: RP/0/RSP0/CPU0:router(config)# encapsulation mfr	Enables multilink Frame Relay on the serial interface.
Step 27	multilink group <i>group-id</i> Example: RP/0/RSP0/CPU0:router(config-if)# multilink group 100	Specifies the multilink group ID for this interface.
Step 28	frame-relay multilink lid <i>link-id name</i> Example: RP/0/RSP0/CPU0:router(config-if)# frame-relay multilink lid sj1	Configures a name for the Frame Relay multilink bundle link.
Step 29	frame-relay multilink ack <i>ack-timeout</i> Example: RP/0/RSP0/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 <i>hello-interval</i> Example: RP/0/RSP0/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 <i>retry-count</i> Example: RP/0/RSP0/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:	Exits interface configuration mode.

	Command or Action	Purpose
	RP/0/RSP0/CPU0:router(config-if)# exit	
Step 33	<p>end or commit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-if)# end</pre> <p>OR</p> <pre>RP/0/RSP0/CPU0:router(config-if)# commit</pre>	<p>Saves configuration changes.</p> <ul style="list-style-type: none"> When you issue the end command, the system prompts you to commit changes: <pre>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</pre> 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	<p>exit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config)# exit</pre>	Exits global configuration mode.
Step 35	<p>show frame-relay multilink interface type interface-path-id [detail verbose]</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router# show frame-relay multilink interface Multilink 0/5/1/0/1</pre>	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.

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* RP/0/RSP0/CPU0:router(config-t1-channel_group)# timeslots 1-24
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** *dldi*
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** [**dldi** | **interface** | **location**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	config Example: RP/0/RSP0/CPU0:router# config	Enters global configuration mode.
Step 2	controller t3 <i>interface-path-id</i> Example: RP/0/RSP0/CPU0:router(config)# controller t3 0/1/0/0	Specifies the T3 controller name in the <i>rack/slot/module/port</i> notation and enters T3 configuration mode.
Step 3	mode <i>type</i> Example: RP/0/RSP0/CPU0:router(config-t3)# mode t1	Configures the type of multilinks to channelize; for example, 28 T1s.
Step 4	clock source { internal line }	(Optional) Sets the clocking for individual E3 links.
	Example:	

	Command or Action	Purpose
	<pre>RP/0/RSP0/CPU0:router(config-t3)# clock source internal</pre>	<p>Note</p> <ul style="list-style-type: none"> The default clock source is internal. 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	<p>exit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-t3)# exit</pre>	Exits T3/E3 or T1/E1 controller configuration mode.
Step 6	<p>controller t1 <i>interface-path-id</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config)# controller t1 0/1/0/0/0</pre>	Enters T1 configuration mode.
Step 7	<p>channel-group <i>channel-group-number</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-t1)# channel-group 0</pre>	Creates a T1 channel group and enters channel group configuration mode for that channel group.
Step 8	<p>timeslots <i>range</i></p> <pre>RP/0/RSP0/CPU0:router(config-t1-channel_group)# timeslots 1-24</pre>	<p>Associates one or more DS0 time slots to a channel group and creates an associated serial subinterface on that channel group.</p> <ul style="list-style-type: none"> 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. <p>Note</p> <ul style="list-style-type: none"> Each individual T1 controller supports a total of 24 DS0 time slots.
Step 9	<p>exit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-t1-channel_group)# exit</pre>	Exits channel group configuration mode.
Step 10	<p>exit</p> <p>Example:</p>	Exits T1 configuration mode.

	Command or Action	Purpose
	<pre>RP/0/RSP0/CPU0:router(config-t1)# exit</pre>	
Step 11	<p>interface serial <i>interface-path-id</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config)# interface serial 0/1/0/0/0:0</pre>	Specifies the complete interface number with the <i>rack/slot/module/port/T3Num/T1num:instance</i> notation.
Step 12	<p>encapsulation frame-relay</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:Router(config-if)# encapsulation frame-relay</pre>	Specifies the Frame Relay encapsulation type.
Step 13	<p>exit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-if)# exit</pre>	Exits interface configuration mode.
Step 14	<p>interface serial <i>interface-path-id</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config)# interface serial 1/0/0/0/0:0.1</pre>	Specifies the complete subinterface number with the <i>rack/slot/module/port/[channel-num:channel-group-number].subinterface</i> notation.
Step 15	<p>ipv4 address <i>ip-address</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 3.1.100.16 255.255.255.0</pre>	<p>Assigns an IP address and subnet mask to the interface in the format:</p> <p><i>A.B.C.D/prefix</i> or <i>A.B.C.D/mask</i></p>
Step 16	<p>pvc <i>dldci</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router (config-subif)# pvc 100</pre>	<p>Creates a POS permanent virtual circuit (PVC) and enters Frame Relay PVC configuration submenu.</p> <p>Replace <i>dldci</i> with a PVC identifier, in the range from 16 to 1007.</p> <p>Note • Only one PVC is allowed per subinterface.</p>
Step 17	<p>service-policy {input output} <i>policy-map</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-fr-vc)# service-policy output policy-mapA</pre>	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.

	Command or Action	Purpose
		<p>Note</p> <ul style="list-style-type: none"> • For effective FRF.12 functionality (interleave specifically), you should configure an egress service policy with priority. • For information on creating and configuring policy maps, refer to <i>Cisco IOS XR Modular Quality of Service Configuration Guide</i>,
Step 18	<p>fragment end-to-end <i>fragment-size</i></p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-fr-vc)# fragment end-to-end 100</pre>	<p>(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.</p> <p>Valid values are from 64 to 512, depending on your hardware.</p>
Step 19	<p>fragment-counter</p> <p>Example:</p> <pre>RP/0/0/CPU0:router(config-fr-vc)# fragment-counter</pre>	<p>(Optional) Enables fragmentation counters for a Frame Relay subinterface and PVC.</p>
Step 20	<p>exit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-fr-vc)# exit</pre>	<p>Exits the Frame-Relay virtual circuit mode.</p>
Step 21	<p>exit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-subif)# exit</pre>	<p>Exits the subinterface configuration mode.</p>
Step 22	<p>exit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-if)# exit</pre>	<p>Exits interface configuration mode.</p>
Step 23	<p>end or commit</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:router(config-if)# end or RP/0/RSP0/CPU0:router(config-if)# commit</pre>	<p>Saves configuration changes.</p> <ul style="list-style-type: none"> • When you issue the end command, the system prompts you to commit changes: <pre>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</pre>

	Command or Action	Purpose
		<ul style="list-style-type: none"> • 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/RSP0/CPU0:router(config)# exit	Exits global configuration mode.
Step 25	show frame-relay pvc [dlci interface location] Example: RP/0/RSP0/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

This 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/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface POS 0/3/0/0
RP/0/RSP0/CPU0:router(config-if)# encapsulation frame-relay IETF
RP/0/RSP0/CPU0:router(config-if)# frame-relay intf-type dce
RP/0/RSP0/CPU0:router(config-if)# frame-relay lmi-type ansi
RP/0/RSP0/CPU0:router(config-if)# no shutdown
RP/0/RSP0/CPU0:router(config-if)# end
```

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

```
RP/0/RSP0/CPU0:router# configure
```

```

RP/0/RSP0RP0/CPU0:router (config)# interface pos 0/3/0/0.10 point-to-point

RP/0/RSP0/CPU0:router (config-subif)#ipv4 address 10.46.8.6/24

RP/0/RSP0RP0/CPU0:router (config-subif)# pvc 20

RP/0/RSP0RP0/CPU0:router (config-fr-vc)# encap ietf

RP/0/RSP0RP0/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/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router (config)# interface
RP/0/RSP0/CPU0:router (config)# interface pos 0/3/0/0
RP/0/RSP0/CPU0:router (config-if)# frame-relay lmi disable
RP/0/RSP0/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/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router (config)# interface
RP/0/RSP0/CPU0:router (config)# interface serial 0/3/0/0
RP/0/RSP0/CPU0:router (config-if)# no frame-relay lmi disable
RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface serial 0/3/0/0/0:0.10 point-to-point
RP/0/RSP0/CPU0:router (config-subif)#ipv4 address 10.46.8.6/24
RP/0/RSP0/CPU0:router (config-subif)# pvc 20
RP/0/RSP0/CPU0:router (config-fr-vc)# encapsulation ietf
RP/0/RSP0/CPU0:router(config-subif)# commit
```

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

```
RP/0/RSP0/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 = ACTIVE, 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 = ACTIVE, 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 = ACTIVE, 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 = ACTIVE, 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/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface pos 0/3/0/0
RP/0/RSP0/CPU0:router(config-if)# frame-relay lmi-n391dte 10
RP/0/RSP0/CPU0:router(config-if)# frame-relay lmi-n391dte 5
RP/0/RSP0/CPU0:router(config-if)# frame-relay lmi-t391dte 15

RP/0/RSP0R0/CPU0:router(config-subif)# commit

RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router# config
RP/0/RSP0/CPU0:router(config)# controller MgmtMultilink 0/3/1/0
RP/0/RSP0/CPU0:router(config-mgmtmultilink)# bundle 100
RP/0/RSP0/CPU0:router(config-mgmtmultilink)# exit

RP/0/RSP0/CPU0:router(config)# controller T3 0/3/1/0
RP/0/RSP0/CPU0:router(config-t3)# mode t1

```

```

RP/0/RSP0/CPU0:router(config-t3)# clock source internal
RP/0/RSP0/CPU0:router(config-t3)# exit

RP/0/RSP0/CPU0:router(config)# controller T1 0/3/1/0/0
RP/0/RSP0/CPU0:router(config-t1)# channel-group 0
RP/0/RSP0/CPU0:router(config-t1-channel_group)# timeslots 1-24
RP/0/RSP0/CPU0:router(config-t1-channel_group)# exit
RP/0/RSP0/CPU0:router(config-t1-channel_group)# exit
RP/0/RSP0/CPU0:router(config-t1)# exit

RP/0/RSP0/CPU0:router(config)# interface Multilink 0/3/1/0/100
RP/0/RSP0/CPU0:router(config-if)# encapsulation frame-relay
RP/0/RSP0/CPU0:router(config-if)# exit

RP/0/RSP0/CPU0:router(config)# interface Multilink 0/3/1/0/100.16 point-to-point
RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 3.1.100.16 255.255.255.0
RP/0/RSP0/CPU0:router(config-subif)# pvc 16
RP/0/RSP0/CPU0:router(config-fr-vc)# service-policy output policy-mapA
RP/0/RSP0/CPU0:router(config-fr-vc)# exit
RP/0/RSP0/CPU0:router(config-subif)# exit

RP/0/RSP0/CPU0:router(config)# interface Serial 0/3/1/0/0:0
RP/0/RSP0/CPU0:router(config-if)# encapsulation mfr
RP/0/RSP0/CPU0:router(config-if)# multilink group 100
RP/0/RSP0/CPU0:router(config-if)# frame-relay multilink lid sj1
RP/0/RSP0/CPU0:router(config-if)# frame-relay multilink ack 5
RP/0/RSP0/CPU0:router(config-if)# frame-relay multilink hello 60
RP/0/RSP0/CPU0:router(config-if)# frame-relay multilink retry 2
RP/0/RSP0/CPU0:router(config-if)# exit
RP/0/RSP0/CPU0:router(config)#

```

End-to-End Fragmentation: Example

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

```

RP/0/RSP0/CPU0:router# config
RP/0/RSP0/CPU0:router(config)# controller T30/3/1/0
RP/0/RSP0/CPU0:router(config-t3)# mode t1
RP/0/RSP0/CPU0:router(config-t3)# clock source internal
RP/0/RSP0/CPU0:router(config-t3)# exit
RP/0/RSP0/CPU0:router(config-t3)# controller T10/3/1/0/0
RP/0/RSP0/CPU0:router(config-t1)# channel-group 0
RP/0/RSP0/CPU0:router(config-t1-channel_group)# timeslots 1-24
RP/0/RSP0/CPU0:router(config-t1-channel_group)# exit
RP/0/RSP0/CPU0:router(config-t1-channel_group)# interface Serial 0/3/1/0/0:0
RP/0/RSP0/CPU0:router(config-if)# encapsulation frame-relay
RP/0/RSP0/CPU0:router(config-if)# exit
RP/0/RSP0/CPU0:router(config-if)# interface Serial 0/3/1/0/0:0.100 point-to-point
RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 3.1.1.1 255.255.255.0
RP/0/RSP0/CPU0:router(config-subif)# pvc 100
RP/0/RSP0/CPU0:router(config-fr-vc)# service-policy output LFI
RP/0/RSP0/CPU0:router(config-fr-vc)# fragment end-to-end 256
fragment-counter

```

