Configuring Serial Interfaces on Cisco IOS XR Sotware

This module describes the configuration of serial interfaces on the Cisco XR 12000 Series Router.

Before you configure a serial interface, you must configure the clear channel T3/E3 controller or channelized T1/E1controller (DS0 channel) that is associated with that interface.

Feature History for Configuring Serial Controller Interfaces

Release	Modification
Release 3.3.0	This feature was introduced on the Cisco XR 12000 Series Router.
	Support was added on the Cisco XR 12000 Series Router for the following hardware:
	• Cisco XR 12000 SIP-401
	• Cisco XR 12000 SIP-501
	• Cisco XR 12000 SIP-601
	Support was added on the Cisco XR 12000 Series Router for the following SPAs:
	• Cisco 2-Port and 4-Port Channelized T3/DS0 SPA
	• Cisco 2-Port and 4-Port T3/E3 Serial SPA
Release 3.4.0	Support for the following features was introduced:
	• Subinterfaces with permanent virtual circuits (PVCs)
	• Frame Relay encapsulation on serial main interfaces and PVCs on the following hardware:
	– Cisco 8-port Channelized T1/E1 SPA
	– Cisco 2-Port and 4-Port Channelized T3/DS0 SPA
	– Cisco 2-Port and 4-Port T3/E3 Serial SPA
	– Cisco 1-Port Channelized OC-3 SPA
	– Cisco 1-Port Channelized OC-12 SPA
	– Cisco 1-Port Channelized OC-48 SPA
	– Cisco 1-Port Channelized OC-12/STM-4 ISE Line Card
Release 3.4.1	Multilink PPP was supported on serial interfaces on the Cisco XR 12000 Series Router.

Release 3.5.0	Support was added on the Cisco XR 12000 Series Router for the following SPAs: • Cisco 1-Port Channelized OC-12/DS0 SPA • Cisco 1-Port Channelized OC-48/STM-16 SPA
Release 3.7.0	Support was added on the Cisco XR 12000 Series Router for the 1-Port Channelized OC-48/DS3 Line Card.
Release 3.8.0	Support was added on the Cisco XR 12000 Series Router for quality of service (QoS) on Layer 2 subinterfaces and the following line cards: • Cisco 1-Port Channelized OC-12/DS0 Line Card • Cisco 4-Port Channelized OC-12/DS3 Line Card
Release 4.0.0	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 Serial Interfaces

Before configuring serial interfaces, ensure that the following tasks and conditions are met:

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

- Your hardware must support T3/E3 or T1/E1 controllers and serial interfaces.

  The following hardware supports T3/E3 controllers and serial interfaces:

  – Cisco 2-Port and 4-Port Clear Channel T3/E3 SPAs

  – Cisco 2-Port and 4-Port Channelized T3 SPAs

  – Cisco 4-Port Channelized OC-12/DS3 line cards

  – Cisco 1-Port Channelized OC-48/STM-16 SPA and line cards

  The following hardware supports T1/E1 controllers and DS0 channels:

  – Cisco 2-Port and 4-Port Channelized T3 SPAs

  – Cisco 4-Port Channelized OC-12/DS3 line cards

  – Cisco 1-Port Channelized OC-12/DS0 SPAs and line cards

– Cisco 1-Port Channelized OC-48/DS3 SPAs and line cards

– Cisco 1-Port Channelized OC3/STM-1 SPA

– Cisco 8-Port Channelized T1/E1 SPA

The following hardware supports serial interfaces:

- Cisco 2-Port and 4-Port Clear Channel T3/E3 SPAs

- Cisco 2-Port and 4-Port Channelized T3 SPAs

- Cisco 4-Port Channelized OC-12/DS3 line cards

- Cisco 1-Port Channelized OC-12/DS0 SPAs and line cards

- Cisco 1-Port Channelized OC-48/DS3 SPAs and line cards

- Cisco 1-Port Channelized OC3/STM-1 SPA

- Cisco 8-Port Channelized T1/E1 SPA

Note The Cisco 2-Port and 4-Port Channelized T3 SPAs can run in clear channel mode, or they can be channelized into 28 T1 or 21 E1 controllers.

- You should have configured the clear channel T3/E3 controller or channelized T3 to T1/E1 controller that is associated with the serial interface you want to configure, as described in the *"Configuring Clear Channel T3/E3 and Channelized T3 and T1/E1 Controllers on Cisco IOS XR Software"* module in this manual.

Note On channelized T3 to T1/E1 controllers, serial interfaces are automatically created when users configure individual DS0 channel groups on the T1/E1 controllers.

Information About Configuring Serial Interfaces

To configure serial interfaces, study the following concepts:

On the , a single serial interface carries data over a single interface using PPP, Cisco HDLC, or Frame Relay encapsulation.

High-Level Overview: Serial Interface Configuration on Clear-Channel SPAs

Table 17 provides a high-level overview of the tasks required to configure a T3 serial interface on the Cisco 2-Port and 4-Port Clear Channel T3/E3 SPA.

Table 17   Overview: Configuring a T3 Serial Interface on a Clear Channel SPA

Step	Task	Module	Section
1.	Use the **hw-module subslot** command to set serial mode for the SPA to be T3, if necessary. **Note** By default, the 2-Port and 4-Port Clear Channel T3/E3 SPA is set to run in T3 mode.	"Configuring Clear Channel T3/E3 and Channelized T3 and T1/E1 Controllers on Cisco IOS XR Software"	Setting the Card Type
2.	Configure the T3 controller.	"Configuring Clear Channel T3/E3 and Channelized T3 and T1/E1 Controllers on Cisco IOS XR Software"	Setting the Card Type
3.	Configure the serial interface that is associated with the T3 controller you configured in Step 2.	"Configuring Serial Interfaces on Cisco IOS XR Sotware"	"How to Configure Serial Interfaces"

Table 18 provides a high-level overview of the tasks required to configure an E3 serial interface on a 2-Port and 4-Port Clear Channel T3/E3 SPA.

Table 18   Overview: Configuring an E3 Serial Interface on a Clear Channel SPA

Step	Task	Module	Section
1.	Use the **hw-module subslot** command to set serial mode for the SPA to be E3.	"Configuring Clear Channel T3/E3 and Channelized T3 and T1/E1 Controllers on Cisco IOS XR Software"	Setting the Card Type
2.	Configure the E3 controller.	"Configuring Clear Channel T3/E3 and Channelized T3 and T1/E1 Controllers on Cisco IOS XR Software"	Setting the Card Type
3.	Configure the serial interface that is associated with the E3 controller you configured in Step 2.	Configuring Serial Interfaces on Cisco IOS XR Sotware	*How to Configure Serial Interfaces*

High-Level Overview: Serial Interface Configuration on Channelized SPAs

Table 19 provides a high-level overview of the tasks required to configure a T1 serial interface on the following SPAs and line cards.

• Cisco 2-Port and 4-Port Channelized T3 SPA

• Cisco 4-Port Channelized OC-12/DS3 line cards

• Cisco 1-Port Channelized OC-12/DS0 SPAs and line cards

• Cisco 1-Port Channelized OC-48/STM-16 SPA and line cards

Table 19   Overview: Configuring a Serial Interface on a T1 DS0 Channel

Step	Task	Module	Section
1.	Configure the T3 controller parameters and set the SPA mode to be T3. 28 T1 controllers are automatically created.	"Configuring Clear Channel T3/E3 and Channelized T3 and T1/E1 Controllers on Cisco IOS XR Software"	Setting the Card Type Configuring a Channelized T3 Controller
2.	Create and configure DS0 channel groups on the T1 controllers.	"Configuring Clear Channel T3/E3 and Channelized T3 and T1/E1 Controllers on Cisco IOS XR Software"	Configuring a T1 Controller
3.	Configure the Serial interfaces that are associated channel groups you created in Step 2.	"Configuring Serial Interfaces on Cisco IOS XR Sotware"	How to Configure Serial Interfaces

Table 20 provides a high-level overview of the tasks required to configure an E1 serial interface on the following SPAs and line cards.

- 2-Port and 4-Port Channelized T3 SPA
- 4-Port Channelized OC-12/DS3 line cards
- 1-Port Channelized OC-12/DS0 SPAs and line cards
- 1-Port Channelized OC-48/DS3 SPAs and line cards

Table 20   Overview: Configuring a Serial Interface on an E1 DS0 Channel

Step	Task	Module	Section
1.	Configure the T3 controller parameters and set the SPA mode to be E3. 21 E1 controllers are automatically created.	"Configuring Clear Channel T3/E3 and Channelized T3 and T1/E1 Controllers on Cisco IOS XR Software"	Configuring a Channelized T3 Controller
2.	Create and configure DS0 channel groups on the E1 controllers.	"Configuring Clear Channel T3/E3 and Channelized T3 and T1/E1 Controllers on Cisco IOS XR Software"	Configuring an E1 Controller
3.	Configure the Serial interfaces that are associated channel groups you created in Step 2.	Configuring Serial Interfaces on Cisco IOS XR Sotware	How to Configure Serial Interfaces

Cisco HDLC Encapsulation

Cisco High-Level Data Link Controller (HDLC) is the Cisco proprietary protocol for sending data over synchronous serial links using HDLC. Cisco HDLC also provides a simple control protocol called Serial Line Address Resolution Protocol (SLARP) to maintain serial link keepalives. HDLC is the default

encapsulation type for serial interfaces under Cisco IOS XR software. Cisco HDLC is the default for data encapsulation at Layer 2 (data link) of the Open System Interconnection (OSI) stack for efficient packet delineation and error control.

Note   Cisco HDLC is the default encapsulation type for the serial interfaces.

Cisco HDLC uses keepalives to monitor the link state, as described in the "Keepalive Timer" section on page 449.

Note   Use the **debug chdlc slarp packet** command to display information about the Serial Line Address Resolution Protocol (SLARP) packets that are sent to the peer after the keepalive timer has been configured.

PPP Encapsulation

PPP is a standard protocol used to send data over synchronous serial links. PPP also provides a Link Control Protocol (LCP) for negotiating properties of the link. LCP uses echo requests and responses to monitor the continuing availability of the link.

Note   When an interface is configured with PPP encapsulation, a link is declared down, and full LCP negotiation is re-initiated after five ECHOREQ packets are sent without receiving an ECHOREP response.

PPP provides the following Network Control Protocols (NCPs) for negotiating properties of data protocols that will run on the link:

- IP Control Protocol (IPCP) to negotiate IP properties
- Multiprotocol Label Switching control processor (MPLSCP) to negotiate MPLS properties
- Cisco Discovery Protocol control processor (CDPCP) to negotiate CDP properties
- IPv6CP to negotiate IP Version 6 (IPv6) properties
- Open Systems Interconnection control processor (OSICP) to negotiate OSI properties

PPP uses keepalives to monitor the link state, as described in the "Keepalive Timer" section on page 449.

PPP supports the following authentication protocols, which require a remote device to prove its identity before allowing data traffic to flow over a connection:

- Challenge Handshake Authentication Protocol (CHAP)—CHAP authentication sends a challenge message to the remote device. The remote device encrypts the challenge value with a shared secret and returns the encrypted value and its name to the local router in a response message. The local router attempts to match the name of the remote device with an associated secret stored in the local username or remote security server database; it uses the stored secret to encrypt the original challenge and verify that the encrypted values match.
- Microsoft Challenge Handshake Authentication Protocol (MS-CHAP)—MS-CHAP is the Microsoft version of CHAP. Like the standard version of CHAP, MS-CHAP is used for PPP authentication; in this case, authentication occurs between a personal computer using Microsoft Windows NT or Microsoft Windows 95 and a Cisco router or access server acting as a network access server.

- Password Authentication Protocol (PAP)—PAP authentication requires the remote device to send a name and a password, which are checked against a matching entry in the local username database or in the remote security server database.

Note For more information on enabling and configuring PPP authentication protocols, see the *"Configuring PPP on Cisco IOS XR Software"* module in this manual.

Use the **ppp authentication** command in interface configuration mode to enable CHAP, MS-CHAP, and PAP on a serial interface.

Note Enabling or disabling PPP authentication does not effect the local router's willingness to authenticate itself to the remote device.

Multilink PPP

Multilink Point-to-Point Protocol (MLPPP) is supported on the following SPAs:

- 1-Port Channelized OC-12/DS0 SPAs and line cards
- 2-Port and 4-Port Channelized T3 SPAs
- 8-Port Channelized T1/E1 SPA

MLPPP provides a method for combining multiple physical links into one logical link. The implementation of MLPPP combines multiple PPP serial interfaces into one multilink interface. MLPPP performs the fragmenting, reassembling, and sequencing of datagrams across multiple PPP links.

MLPPP provides the same features that are supported on PPP Serial interfaces with the exception of QoS. It also provides the following additional features:

- Fragment sizes of 128, 256, and 512 bytes
- Long sequence numbers (24-bit)
- Lost fragment detection timeout period of 80 ms
- Minimum-active-links configuration option
- LCP echo request/reply support over multilink interface
- Full T1 and E1 framed and unframed links

For more information about configuring MLPPP on a serial interface, see the *"Configuring PPP on Cisco IOS XR Software"* module in this document.

Keepalive Timer

Cisco keepalives are useful for monitoring the link state. Periodic keepalives are sent to and received from the peer at a frequency determined by the value of the keepalive timer. If an acceptable keepalive response is not received from the peer, the link makes the transition to the down state. As soon as an acceptable keepalive response is obtained from the peer or if keepalives are disabled, the link makes the transition to the up state.

Note The **keepalive** command applies to serial interfaces using HDLC or PPP encapsulation. It does not apply to serial interfaces using Frame Relay encapsulation.

For each encapsulation type, a certain number of keepalives ignored by a peer triggers the serial interface to transition to the down state. For HDLC encapsulation, three ignored keepalives causes the interface to be brought down. For PPP encapsulation, five ignored keepalives causes the interface to be brought down. ECHOREQ packets are sent out only when LCP negotiation is complete (for example, when LCP is open).

Note   Use the **keepalive** command in interface configuration mode to set the frequency at which LCP sends ECHOREQ packets to its peer. To restore the system to the default keepalive interval of 10 seconds, use the **keepalive** command with **no** argument. To disable keepalives, use the **keepalive disable** command. For both PPP and Cisco HDLC, a keepalive of 0 disables keepalives and is reported in the **show running-config** command output as **keepalive disable**.Before performing a Minimal Disruptive Restart (MDR) upgrade, we recommend disabling keepalives on a Cisco XR 12000 Series Router.

When LCP is running on the peer and receives an ECHOREQ packet, it responds with an echo reply (ECHOREP) packet, regardless of whether keepalives are enabled on the peer.

Keepalives are independent between the two peers. One peer end can have keepalives enabled; the other end can have them disabled. Even if keepalives are disabled locally, LCP still responds with ECHOREP packets to the ECHOREQ packets it receives. Similarly, LCP also works if the period of keepalives at each end is different.

Note   Use the **debug chdlc slarp packet** command and other Cisco HDLC **debug** commands to display information about the Serial Line Address Resolution Protocol (SLARP) packets that are sent to the peer after the keepalive timer has been configured.

Frame Relay Encapsulation

When Frame Relay encapsulation is enabled on a serial interface, the interface configuration is hierarchical and comprises the following elements:

1. The serial main interface comprises the physical interface and port. If you are not using the serial interface to support Cisco HDLC and PPP encapsulated connections, then you must configure subinterfaces with permanent virtual circuits (PVCs) under the serial main interface. Frame Relay connections are supported on PVCs only.

2. Serial subinterfaces are configured under the serial main interface. A serial subinterface does not actively carry traffic until you configure a PVC under the serial subinterface. Layer 3 configuration typically takes place on the subinterface.

3. Point-to-point PVCs are configured under a serial subinterface. You cannot configure a PVC directly under a main interface. A single point-to-point PVC is allowed per subinterface. PVCs use a predefined circuit path and fail if the path is interrupted. PVCs remain active until the circuit is removed from either configuration. Connections on the serial PVC support Frame Relay encapsulation only.

Note   The administrative state of a parent interface drives the state of the subinterface and its PVC. When the administrative state of a parent interface or subinterface changes, so does the administrative state of any child PVC configured under that parent interface or subinterface.

To configure Frame Relay encapsulation on serial interfaces, use the **encapsulation frame-relay** command.

Frame Relay interfaces support two types of encapsulated frames:

- Cisco (default)
- IETF

Use the **encap** command in PVC configuration mode to configure Cisco or IETF encapsulation on a PVC. If the encapsulation type is not configured explicitly for a PVC, then that PVC inherits the encapsulation type from the main serial interface.

Note   Cisco encapsulation is required on serial main interfaces that are configured for MPLS. IETF encapsulation is not supported for MPLS.

Before you configure Frame Relay encapsulation on an interface, you must verify that all prior Layer 3 configuration is removed from that interface. For example, you must ensure that there is no IP address configured directly under the main interface; otherwise, any Frame Relay configuration done under the main interface will not be viable.

LMI on Frame Relay Interfaces

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 UNI interface. By default, **cisco** LMI is enabled on all PVCs. However, you can modify the default LMI type to be ANSI or Q.933, as described in the *"Modifying the Default Frame Relay Configuration on an Interface"* section of the *"Configuring Frame Relay Cisco IOS XR Software"* module in this manual.

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
```

Note   The default setting of the **mtu** command for a serial interface is 1504 bytes. Therefore, the default numbers of PVCs supported on a serial interface configured with **cisco** LMI is 186.

Layer 2 Tunnel Protocol Version 3-Based Layer 2 VPN on Frame Relay

The Layer 2 Tunnel Protocol Version 3 (L2TPv3) feature defines the L2TP protocol for tunneling Layer 2 payloads over an IP core network using Layer 2 virtual private networks (VPNs).

L2TPv3 is a tunneling protocol used for transporting Layer 2 protocols. It can operate in a number of different configurations and tunnel a number of different Layer 2 protocols and connections over a packet-switched network.

Before you can configure L2TPv3, you need to configure a connection between the two attachment circuits (ACs) that will host the L2TPv3 psuedowire. Cisco IOS XR software supports a point-to-point, end-to-end service, where two ACs are connected together.

This module describes how to configure a Layer 2 AC on a Frame Relay encapsulated serial interface.

Note   Serial interfaces support DLCI mode layer 2 ACs only; layer 2 port mode ACs are not supported on serial interfaces.

For detailed information about configuring L2TPv3 in your network, see the *"Implementing Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR Software"* module of the *Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco XR 12000 Series Router.* For detailed information about configuring L2VPNs, see the *"Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software"* module of the *Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco XR 12000 Series Router.*

Default Settings for Serial Interface Configurations

When an interface is enabled on a T3/E3 SPA, and no additional configuration commands are applied, the default interface settings shown in Table 21 are present. These default settings can be changed by configuration.

Table 21   Serial Interface Default Settings

Parameter	Configuration File Entry	Default Settings
Keepalive **Note** The **keepalive** command applies to serial interfaces using HDLC or PPP encapsulation. It does not apply to serial interfaces using Frame Relay encapsulation.	**keepalive** [**disable**] **no keepalive**	keepalive 10 seconds
Encapsulation	**encapsulation** [**hdlc** | **ppp** | **frame-relay** [**ietf**]]	hdlc
Maximum transmission unit (MTU)	**mtu** *bytes*	1504 bytes
Cyclic redundancy check (CRC)	**crc** [**16** | **32**]	16
Data stream inversion on a serial interface	**invert**	Data stream is not inverted
Payload scrambling (encryption)	**scramble**	Scrambling is disabled.
Number of High-Level Data Link Control (HDLC) flag sequences to be inserted between the packets	**transmit-delay**	Default is 0 (disabled).

Note   Default settings do not appear in the output of the **show running-config** command.

Serial Interface Naming Notation

The naming notation for serial interfaces on a clear channel SPA is *rack/slot/module/port*, as shown in the following example:

```
interface serial 0/0/1/2
```

The naming notation for T1, E1, and DS0 interfaces on a channelized SPA is
rack/*slot*/*module*/*port*/*channel-num:channel-group-number*, as shown in the following example:

```
interface serial 0/0/1/2/4:3
```

If a subinterface and PVC are configured under the serial interface, then the router includes the
subinterface number at the end of the serial interface address. In this case, the naming notation is
rack/*slot*/*module*/*port*[/*channel-num:channel-group-number*].*subinterface,* as shown in the following
examples:

```
interface serial 0/0/1/2.1
interface serial 0/0/1/2/4:3.1
```

Note   A slash between values is required as part of the notation.

The naming notation syntax for serial interfaces is as follows:

- *rack*: Chassis number of the rack.
- *slot*: Physical slot number of the modular services card or line card.
- *module*: Module number. Shared port adapters (SPAs) are referenced by their subslot number.
- *port*: Physical port number of the controller.
- *channel-num*: T1 or E1 channel number. T1 channels range from 0 to 23; E1 channels range from 0 to 31.
- *channel-group-number*: Time slot number. T1 time slots range from 1 to 24; E1 time slots range from 1 to 31. The *channel-group-number* is preceded by a colon and not a slash.
- *subinterface*: Subinterface number.

Use the question mark (**?**) online help function following the **serial** keyword to view a list of all valid
interface choices.

IPHC Overview

IP header compression (IPHC) is based on the premise that most of the headers in the packets of a
particular transmission remain constant throughout the flow. Only a few fields in the headers of related
packets change during a flow.

IPHC compresses these headers so that the compressed header contains only the fields that change from
packet to packet. All fields that remain the same from packet to packet are eliminated in the compressed
headers. Full headers are sent between compressed headers.

Full headers are uncompressed headers that contain all the original header fields along with additional
information (context ID) to identify the flow. The interval at which full headers are sent between
compressed packets is configurable using the **refresh max-period** and command.

IPHC contexts are used by the compressor (sender) and decompressor (receiver) of compressed packets
to encode and decode the packets in a flow. A context is stored on the compressor and decompressor and
is used in the delta calculation at both ends. The number of contexts allowed on a particular interface is
configurable. The maximum size of the header that can be compressed is also configurable.

IPHC supports the compression and decompression of RTP and UDP traffic and the decompression of
CN on TCP and CTCP traffic.

Users may choose one of the following types of compression formats:

- Internet Engineering Task Force (IETF) standard format.
  Uses RFC2507 and RFC2508 compression schemes.

- IPHC format.
  Provides options similar to IETF.

Table 22 shows the IPHC features, the values of the features, and their defaults:

Table 22   IPHC features and default setttings

IPHC Feature	Values	Defaults
TCP contexts	0 to 255	1
Non-TCP contexts	1 to 6000	16
Compression Format Options	IETF or IPHC	—
Feedback Messages	Enable or Disable	Enabled
Maximum Refresh Period Size	1 to 65535 packets	256
Maximum Header Size	20 to 40 bytes	40
Real Time Protocol (RTP)	Enable or Disable	Enabled
Refresh RTP	Enable or Disable	Disable

Currently, only IPv4 unicast packets with UDP in the protocol field of the IP header are compressed.

IPHC is configured on an interface as follows:

- Configure the IPHC slot level command

- Create an IPHC profile

- Configure IPHC attributes in the profile

- Attach the profile to an interface

IPHC profiles must contain the **rtp** command to enable Real Time Protocol (RTP) on the interface, or the profile is not enabled. The **refresh rtp** command must be used to enable the configured refresh settings for RTP packets. By default, refresh RTP is disabled and only the first packet in the flow is sent as a 'full-header' packet.

If some attributes, such as feedback messages, maximum refresh period size and maximum header size, are not configured in the profile, the default values for those attributes apply when the profile is enabled on the interface.

Currently, IPHC is supported only on serial interfaces with PPP encapsulation and on multilink with PPP encapsulation interfaces.

IPHC is typically configured between the Customer Edge (CE) and Provide Edge (PE) ends of an interface and must be configured at both ends of the interface to work. The PPP protocol negotiates the IPHC specific parameters between the two ends of the interface and settles on the lowest value configured between the two ends.

QoS and IPHC

An IPHC profile can be enabled on an interface so that the IPHC profile applies only to packets that match a Quality of Service (QoS) service policy. In this case, the QoS service-policy class attributes determine which packets are compressed. This allows users to fine tune IPHC with greater granularity.

Policy maps are attached to an interface using the **service-policy** command. IPHC action applies only to output service policies. IPHC is not supported on input service policies.

The user can configure IPHC using QoS as follows:

- Create a QoS **policy-map** with the **compress header ip** action.

- Attach the IPHC profile to the interface using the **ipv4 iphc profile** *profile_name* **mode service-policy** command.

- Attach the QoS **policy-map** with **compress header ip** action using the **service-policy output** command.

See "IPHC on a Serial Interface with MLPPP/LFI and QoS Configuration: Example" section on page 484 for an example of how to configure IPHC using QoS.

For complete information on configuring QoS, refer to the *Cisco XR 12000 Series Router Modular Quality of Service Configuration Guide* and the *Cisco XR 12000 Series Router Modular Quality of Service Command Reference*.

How to Configure Serial Interfaces

After you have configured a channelized or clear channel T3/E3 controller, as described in the "Configuring Clear Channel T3/E3 and Channelized T3 and T1/E1 Controllers on Cisco IOS XR Software" module in this document, you can configure the serial interfaces associated with that controller.

The following tasks describe how to configure a serial interface:

Bringing Up a Serial Interface

This task describes the commands used to bring up a serial interface.

Prerequisites

The Cisco XR 12000 Series Router must have at least one of the SIPs and one of the SPAs or line cards installed and be running Cisco IOS XR software:

- Cisco XR 12000 SIP-401
- Cisco XR 12000 SIP-501
- Cisco XR 12000 SIP-601
- 2-Port and 4-Port T3/E3 Serial SPA
- 2-Port and 4-Port Channelized T3/DS0 Serial SPA
- 4-Port Channelized OC-12/DS3 line cards
- 1-Port Channelized OC-12/DS0 SPAs and line cards
- 1-Port Channelized OC-48/DS3 SPAs and line cards
- 1-Port Channelized OC3/STM-1 SPA
- 8-Port Channelized T1/E1 SPA

Restrictions

The configuration on both ends of the serial connection must match for the interface to be active.

SUMMARY STEPS

1. **show interfaces**
2. **configure**
3. **interface serial** *interface-path-id*
4. **ipv4 address** *ip-address*
5. **no shutdown**
6. **end**
   or
   commit
7. **exit**
8. **exit**
9. Repeat Step 1 through Step 8 to bring up the interface at the other end of the connection.
10. **show ipv4 interface brief**
11. **show interfaces serial** *interface-path-id*

DETAILED STEPS

	Command or Action	Purpose

Step 1	`show interfaces`
	
	Example:
	`RP/0/0/CPU0:router# show interfaces`
		(Optional) Displays configured interfaces.
		
		• Use this command to also confirm that the router recognizes the PLIM card.

Step 2	`configure`
	
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 3	`interface serial` *interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface serial 0/1/0/0`	Specifies the serial interface name and notation *rack/slot/module/port,* and enters interface configuration mode.
Step 4	`ipv4 address` *ip-address* **Example:** `RP/0/0/CPU0:router(config-if)# ipv4 address 10.1.2.1 255.255.255.224`	Assigns an IP address and subnet mask to the interface. **Note** Skip this step if you are configuring Frame Relay encapsulation on this interface. For Frame Relay, the IP address and subnet mask are configured under the subinterface.

Step 5	`no shutdown`
	
	Example:
	`RP/0/0/CPU0:router (config-if)# no shutdown`
		Removes the shutdown configuration.
		
		Note Removal of the shutdown configuration eliminates the forced administrative down on the interface, enabling it to move to an up or down state (assuming the parent SONET layer is not configured administratively down).

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.

	Command or Action	Purpose
Step 7	**exit** **Example:** RP/0/0/CPU0:router (config-if)# exit	Exits interface configuration mode and enters global configuration mode.
Step 8	**exit** **Example:** RP/0/0/CPU0:router (config)# exit	Exits global configuration mode and enters EXEC mode.
Step 9	**show interfaces** **configure** **interface serial** *interface-path-id* **no shut** **exit** **exit** **Example:** RP/0/0/CPU0:router# show interfaces RP/0/0/CPU0:router# configure RP/0/0/CPU0:router (config)# interface serial 0/1/0/1 RP/0/0/CPU0:router(config-if)# ipv4 address 10.1.2.2 255.255.255.224 RP/0/0/CPU0:router (config-if)# no shutdown RP/0/0/CPU0:router (config-if)# commit RP/0/0/CPU0:router (config-if)# exit RP/0/0/CPU0:router (config)# exit	Repeat Step 1 through Step 8 to bring up the interface at the other end of the connection. **Note**　The configuration on both ends of the serial connection must match.
Step 10	**show ipv4 interface brief** **Example:** RP/0/0/CPU0:router # show ipv4 interface brief	Verifies that the interface is active and properly configured. If you have brought up a serial interface properly, the "Status" field for that interface in the **show ipv4 interface brief** command output displays "Up."
Step 11	**show interfaces serial** *interface-path-id* **Example:** RP/0/0/CPU0:router# show interfaces serial 0/1/0/0	(Optional) Displays the interface configuration.

What to Do Next

To modify the default configuration of the serial interface you just brought up, see the .

Configuring Optional Serial Interface Parameters

This task describes the commands used to modify the default configuration on a serial interface.

Prerequisites

Before you modify the default serial interface configuration, you must bring up the serial interface and remove the shutdown configuration, as described in the "Bringing Up a Serial Interface" section on page 455.

Restrictions

The configuration on both ends of the serial connection must match for the interface to be active.

SUMMARY STEPS

1. **configure**

2. **interface serial** *interface-path-id*

3. **encapsulation** [**hdlc** | **ppp** | **frame-relay** [**IETF**]

4. **serial**

5. **crc** *length*

6. **invert**

7. **scramble**

8. **transmit-delay** *hdlc-flags*

9. **end**
   or
   commit

10. **exit**

11. **exit**

12. **exit**

13. **show interfaces serial** [*interface-path-id*]

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface serial** *interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface serial 0/1/0/0`	Specifies the serial interface name and notation *rack/slot/module/port,* and enters interface configuration mode.
Step 3	**encapsulation** [**hdlc** | **ppp** | **frame-relay** [**IETF**] **Example:** `RP/0/0/CPU0:router(config-if)# encapsulation hdlc`	(Optional) Configures the interface encapsulation parameters and details such as HDLC, PPP or Frame Relay. **Note** The default encapsulation is **hdlc**.

	Command or Action	Purpose
Step 4	**serial** **Example:** RP/0/0/CPU0:router(config-if)# serial	(Optional) Enters serial submode to configure the serial parameters.
Step 5	**crc** *length* **Example:** RP/0/0/CPU0:ios(config-if-serial)# crc 32	(Optional) Specifies the length of the cyclic redundancy check (CRC) for the interface. Enter the **16** keyword to specify 16-bit CRC mode, or enter the **32** keyword to specify 32-bit CRC mode. **Note** The default is CRC length is 16.
Step 6	**invert** **Example:** RP/0/0/CPU0:ios(config-if-serial)# inverts	(Optional) Inverts the data stream.
Step 7	**scramble** **Example:** RP/0/0/CPU0:ios(config-if-serial)# scramble	(Optional) Enables payload scrambling on the interface. **Note** Payload scrambling is disabled on the interface.
Step 8	**transmit-delay** *hdlc-flags* **Example:** RP/0/0/CPU0:ios(config-if-serial)# transmit-delay 10	(Optional) Specifies a transmit delay on the interface. Values can be from 0 to 128. **Note** Transmit delay is disabled by default (the transmit delay is set to **0**).
Step 9	**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 10	**exit** **Example:** RP/0/0/CPU0:router(config-if-serial)# exit	Exits serial configuration mode.

	Command or Action	Purpose

Step 11	`exit`
	
	Example:
	`RP/0/0/CPU0:router (config-if)# exit`
		Exits interface configuration mode and enters global configuration mode.

Step 12	`exit`
	
	Example:
	`RP/0/0/CPU0:router (config)# exit`
		Exits global configuration mode and enters EXEC mode.

Step 13	`show interfaces serial` [*interface-path-id*] **Example:** `RP/0/0/CPU0:router# show interface serial 0/1/0/0`	(Optional) Displays general information for the specified serial interface.

What to Do Next

- To create a point-to-point Frame Relay subinterface with a PVC on the serial interface you just brought up, see the "Creating a Point-to-Point Serial Subinterface with a PVC" section on page 461.

- To configure PPP authentication on serial interfaces with PPP encapsulation, see the *"Configuring PPP on Cisco IOS XR Software"* module later in this manual.

- To modify the default keepalive configuration, see the "Modifying the Keepalive Interval on Serial Interfaces" section on page 466.

- To modify the default Frame Relay configuration on serial interfaces that have Frame Relay encapsulation enabled, see the "Modifying the Default Frame Relay Configuration on an Interface" section of the *"Configuring Frame Relay Cisco IOS XR Software"* module.

Creating a Point-to-Point Serial Subinterface with a PVC

The procedure in this section creates a point-to-point serial subinterface and configures a permanent virtual circuit (PVC) on that serial subinterface.

Note Subinterface and PVC creation is supported on interfaces with Frame Relay encapsulation only.

Prerequisites

Before you can create a subinterface on a serial interface, you must bring up the main serial interface with Frame Relay encapsulation, as described in the "Bringing Up a Serial Interface" section on page 455.

Restrictions

Only one PVC can be configured for each point-to-point serial subinterface.

SUMMARY STEPS

1. **configure**

2. **interface serial** *interface-path-id.subinterface* **point-to-point**

3. **ipv4 address** *ipv4_address/prefix*

4. **pvc** *dlci*

5. **end**
   or
   commit

6. Repeat Step 1 through Step 5 to bring up the serial subinterface and any associated PVC at the other end of the connection.

DETAILED STEPS

	Command or Action	Purpose

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

Step 2	`interface serial` *interface-path-id.subinterface* `point-to-point`
	
	Example:
	`RP/0/0/CPU0:router (config)# interface serial 0/1/0/0.1`
		Enters serial subinterface configuration mode.

Step 3	`ipv4 address` *ipv4_address/prefix* **Example:** `RP/0/0/CPU0:router (config-subif)#ipv4 address 10.46.8.6/24`	Assigns an IP address and subnet mask to the subinterface.
Step 4	`pvc` *dlci* **Example:** `RP/0/0/CPU0:router (config-subif)# pvc 20`	Creates a serial 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.

	Command or Action	Purpose

Step 5	**end**
	or
	commit
	
	Example:
	RP/0/0/CPU0:router (config-subif)# end
	or
	RP/0/0/CPU0:router(config-subif)# 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 6	**configure** **interface serial** *interface-path-id* **pvc** *dlci* **commit** **Example:** RP/0/0/CPU0:router# configure RP/0/0/CPU0:router (config)# interface serial 0/1/0/1.1 RP/0/0/CPU0:router (config-subif)#ipv4 address 10.46.8.5/24 RP/0/0/CPU0:router (config-subif)# pvc 20 RP/0/0/CPU0:router (config-fr-vc)# commit	Repeat Step 1 through Step 5 to bring up the serial subinterface and any associated PVC at the other end of the connection. **Note** The DLCI (or PVC identifier) must match on both ends of the subinterface connection. **Note** When assigning an IP address and subnet mask to the subinterface at the other end of the connection, keep in mind that the addresses at both ends of the connection must be in the same subnet.

What to Do Next

- To configure optional PVC parameters, see the "Configuring Optional Serial Interface Parameters" section on page 458.

- To modify the default Frame Relay configuration on serial interfaces that have Frame Relay encapsulation enabled, see the "Modifying the Default Frame Relay Configuration on an Interface" section of the *"Configuring Frame Relay Cisco IOS XR Software"* module.

- To attach a Layer 3 QOS service policy to the PVC under the PVC submode, refer to the appropriate Cisco IOS XR software configuration guide.

Configuring Optional PVC Parameters

This task describes the commands you can use to modify the default configuration on a serial PVC.

For additional information about Frame Relay options, see the "Configuring Frame Relay on Cisco IOS XR Software" module in the *Cisco IOS XR Interface and Hardware Component Configuration Guide for the Cisco XR 12000 Series Router.*

Prerequisites

Before you can modify the default PVC configuration, you must create the PVC on a serial subinterface, as described in the .

Restrictions

- The DLCI (or PVI identifier) must match on both ends of the PVC for the connection to be active.
- To change the PVC DLCI, you must delete the PVC and then add it back with the new DLCI.

SUMMARY STEPS

1. **configure**
2. **interface serial** *interface-path-id.subinterface*
3. **pvc** *dlci*
4. **encap** [**cisco** | **ietf**]
5. **service-policy** {**input** | **output**} *policy-map*
6. **end**
   or
   commit
7. Repeat Step 1 through Step 6 to configure the PVC at the other end of the connection.
8. **show frame-relay pvc** *dlci-number*
9. **show policy-map interface serial** *interface-path-id.subinterface* {**input** | **output**}
   or
   show policy-map type qos interface serial *interface-path-id.subinterface* {**input** | **output**}

	Command or Action	Purpose

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

Step 2	`interface serial` *interface-path-id.subinterface*
	
	Example:
	`RP/0/0/CPU0:router (config)# interface serial`
	`0/1/0/0.1`
		Enters serial subinterface configuration mode.

	Command or Action	Purpose
Step 3	**pvc** *dlci* **Example:** RP/0/0/CPU0:router (config-subif)# pvc 20	Enters subinterface configuration mode for the PVC.
Step 4	**encap** [**cisco** | **ietf**] **Example:** RP/0/0/CPU0:router (config-fr-vc)# encap ietf	(Optional) 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 serial interface.
Step 5	**service-policy** {**input** | **output**} *policy-map* **Example:** RP/0/0/CPU0:router (config-fr-vc)# service-policy output policy1	Attaches a policy map to an input subinterface or output subinterface. Once attached, the policy map is used as the service policy for the subinterface.

Step 6	**end**
	or
	commit
	
	Example:
	RP/0/0/CPU0:router (config-fr-vc)# end
	or
	RP/0/0/CPU0:router(config-fr-vc)# 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	**configure** **interface serial** *interface-path-id.subinterface* **pvc** *dlci* **encap** [**cisco** | **ietf**] **commit** **Example:** RP/0/0/CPU0:router# configure RP/0/0/CPU0:router (config)# interface serial 0/1/0/1.1 RP/0/0/CPU0:router (config-subif)# pvc 20 RP/0/0/CPU0:router (config-fr-vc)# encap cisco RP/0/0/CPU0:router (config-fr-vc)# commit	Repeat Step 1 through Step 6 to bring up the serial subinterface and any associated PVC at the other end of the connection. **Note** The configuration on both ends of the subinterface connection must match.

Command or Action	Purpose
Step 8 **show frame-relay pvc** *dlci-number* **Example:** `RP/0/0/CPU0:router# show frame-relay pvc 20`	(Optional) Verifies the configuration of specified serial interface.

Step 9 **show policy-map interface serial** *interface-path-id.subinterface* {**input** | **output**}
or
show policy-map type qos interface serial *interface-path-id.subinterface* {**input** | **output**}
 	
Example:
`RP/0/0/CPU0:router# show policy-map interface serial 0/1/0/0.1 output`
or
`RP/0/0/CPU0:router# show policy-map type qos interface serial 0/1/0/0.1 output`
	(Optional) Displays the statistics and the configurations of the input and output policies that are attached to a subinterface.

What to Do Next

- To modify the default Frame Relay configuration on serial interfaces that have Frame Relay encapsulation enabled, see the *"Modifying the Default Frame Relay Configuration on an Interface"* section of the *"Configuring Frame Relay Cisco IOS XR Software"* module in this manual.

Modifying the Keepalive Interval on Serial Interfaces

Perform this task to modify the keepalive interval on serial interfaces that have Cisco HDLC or PPP encapsulation enabled.

Note   When you enable Cisco HDLC or PPP encapsulation on a serial interface, the default keepalive interval is 10 seconds. Use this procedure to modify that default keepalive interval.

Note   Cisco HDLC is enabled by default on serial interfaces.

Prerequisites

Before modifying the keepalive timer configuration, ensure that Cisco HDLC or PPP encapsulation is enabled on the interface. Use the **encapsulation** command to enable Cisco HDLC or PPP encapsulation on the interface, as described in the *"Configuring Optional Serial Interface Parameters"* section on page 458.

Restrictions

- Before performing a Minimal Disruptive Restart (MDR) upgrade, we recommend disabling keepalives on a Cisco XR 12000 Series Router.

SUMMARY STEPS

1. **configure**

2. **interface serial** *interface-path-id*

3. **keepalive** {*seconds* | **disable**}
   or
   no keepalive

4. **end**
   or
   commit

5. **show interfaces** *type interface-path-id*

DETAILED STEPS

	Command or Action	Purpose

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

Step 2	`interface serial` *interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface serial 0/1/0/0`	Specifies the serial interface name and notation *rack/slot/module/port* and enters interface configuration mode.

Step 3	`keepalive` {*seconds* | **disable**}
	or
	no keepalive
	
	Example:
	`RP/0/0/CPU0:router(config-if)# keepalive 3`
	or
	`RP/0/0/CPU0:router(config-if)# no keepalive`
		Specifies the number of seconds between keepalive messages.
		
		• Use the **keepalive disable** command, the **no keepalive**, or the **keepalive** command with an argument of 0 to disable the keepalive feature.
		
		• The range is from 1 to 30 seconds. The default is 10 seconds.
		
		• If keepalives are configured on an interface, use the **no keepalive** command to disable the keepalive feature before configuring Frame Relay encapsulation on that interface.

	Command or Action	Purpose

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 serial` *interface-path-id* **Example:** `RP/0/0/CPU0:router# show interfaces serial 0/1/0/0`	(Optional) Verifies the interface configuration.

How to Configure a Layer 2 Attachment Circuit

The Layer 2 AC configuration tasks are described in the following procedures:

- Creating a Serial Layer 2 Subinterface with a PVC
- Configuring Optional Serial Layer 2 PVC Parameters

Note After you configure an interface for Layer 2 switching, no routing commands such as **ipv4 address** are permissible. If any routing commands are configured on the interface, then the **l2transport** command is rejected.

Creating a Serial Layer 2 Subinterface with a PVC

The procedure in this section creates a Layer 2 subinterface with a PVC.

Prerequisites

Before you can create a subinterface on a serial interface, you must bring up a serial interface, as described in the "Bringing Up a Serial Interface" section on page 455.

Restrictions

Only one PVC can be configured for each serial subinterface.

SUMMARY STEPS

1. **configure**

2. **interface serial** *interface-path-id.subinterface* **l2transport**

3. **pvc** *vpi*/*vci*

4. **end**
   or
   commit

5. Repeat Step 1 through Step 4 to bring up the serial subinterface and any associated PVC at the other end of the AC.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface serial** *interface-path-id.subinterface* **l2transport** **Example:** `RP/0/0/CPU0:router(config)# interface serial 0/1/0/0.1 l2transport`	Creates a subinterface and enters serial subinterface configuration mode for that subinterface.
Step 3	**pvc** *vpi*/*vci* **Example:** `RP/0/0/CPU0:router(config-if)# pvc 5/20`	Creates a serial permanent virtual circuit (PVC) and enters serial Layer 2 transport PVC configuration mode. **Note** Only one PVC is allowed per subinterface.

	Command or Action	Purpose
Step 4	**end** or **commit** **Example:** RP/0/0/CPU0:router(config-fr-vc)# end or RP/0/0/CPU0:router(config-fr-vc)# 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	Repeat Step 1 through Step 4 to bring up the serial subinterface and any associated PVC at the other end of the AC.	Brings up the AC. **Note** The configuration on both ends of the AC must match.

What to Do Next

• To configure optional PVC parameters, see the "Configuring Optional Serial Layer 2 PVC Parameters" section on page 470.

• For detailed information about configuring L2TPv3 in your network, see the *"Implementing Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR Software"* module of the *Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco XR 12000 Series Router.* For detailed information about configuring L2VPNs, see the *"Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software"* module of the *Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco XR 12000 Series Router.*

Configuring Optional Serial Layer 2 PVC Parameters

This task describes the commands you can use to modify the default configuration on a serial Layer 2 PVC.

Prerequisites

Before you can modify the default PVC configuration, you must create the PVC on a Layer 2 subinterface, as described in the "Creating a Serial Layer 2 Subinterface with a PVC" section on page 468.

Restrictions

The configuration on both ends of the PVC must match for the connection to be active.

SUMMARY STEPS

1. **configure**
2. **interface serial** *interface-path-id.subinterface* **l2transport**
3. **pvc** *dlci*
4. **encap** [**cisco** | **ietf**]
5. **service-policy** {**input** | **output**} *policy-map*
6. **fragment end-to-end** *fragment-size*
7. fragment-counter
8. **end**
   or
   commit
9. Repeat Step 1 through Step 7 to configure the PVC at the other end of the AC.
10. **show policy-map interface serial** *interface-path-id.subinterface* {**input** | **output**}
    or
    show policy-map type qos interface serial *interface-path-id.subinterface* {**input** | **output**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/0/CPU0:router# configure	Enters global configuration mode.
Step 2	**interface serial** *interface-path-id.subinterface* **l2transport** **Example:** RP/0/0/CPU0:router(config)# interface serial 0/1/0/0.1 l2transport	Enters serial subinterface configuration mode for a Layer 2 serial subinterface.
Step 3	**pvc** *dlci* **Example:** RP/0/0/CPU0:router(config-if)# pvc 100	Enters serial Frame Relay PVC configuration mode for the specified PVC.
Step 4	**encap** {**cisco**	**ietf**} **Example:** RP/0/0/CPU0:router(config-fr-vc)# encapsulation aal5	Configures the encapsulation for a Frame Relay PVC.

	Command or Action	Purpose
Step 5	**service-policy** {**input** | **output**} *policy-map* **Example:** RP/0/0/CPU0:router (config-subif)# service-policy output policy1	Attaches a policy map to an input subinterface or output subinterface. Once attached, the policy map is used as the service policy for the subinterface.
Step 6	**fragment end-to-end** *fragment-size* **Example:** RP/0/0/CPU0:router(config-fr-vc)# fragment end-to-end 100	Enables fragmentation of Frame Relay frames on an interface. Replace *fragment-size* with the number of payload bytes from the original Frame Relay frame that will go into each fragment. This number excludes the Frame Relay header of the original frame. On the Cisco 8-Port Channelized T1/E1 SPA, valid values are 128, 256, and 512.
Step 7	**fragment-counter** **Example:** RP/0/0/CPU0:router(config-fr-vc)# fragment-counter	Enables fragmentation counters for a Frame Relay subinterface and PVC.
Step 8	**end** or **commit** **Example:** RP/0/0/CPU0:router(config-serial-l2transport-pvc)# end or RP/0/0/CPU0:router(config-serial-l2transport-pvc)# 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.

	Command or Action	Purpose
Step 9	Repeat Step 1 through Step 7 to configure the PVC at the other end of the AC.	Brings up the AC. **Note** The configuration on both ends of the connection must match.

Step 10	`show policy-map interface serial` *interface-path-id.subinterface* {**input** | **output**}
	 or
	 `show policy-map type qos interface serial` *interface-path-id.subinterface* {**input** | **output**}
	
	Example:
	`RP/0/0/CPU0:router# show policy-map interface pos 0/1/0/0.1 output`
	
	or
	
	`RP/0/0/CPU0:router# show policy-map type qos interface pos 0/1/0/0.1 output`
		(Optional) Displays the statistics and the configurations of the input and output policies that are attached to a subinterface.

What to Do Next

- To configure a point-to-point pseudowire XConnect on the AC you just created, see the "*Implementing Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR Software*" module of the *Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco XR 12000 Series Router.*

- To configure an L2VPN, see the "*Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software*" module of the *Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco XR 12000 Series Router.*

Configuring IPHC

This section contains the following step procedures:

Prerequisites for Configuring IPHC

IP header compression (IPHC) is supported on the following cards:

- Cisco 1-Port Channelized OC-12/STM-4
- Cisco 1-Port Channelized OC-48/STM-16
- Cisco 1-Port Channelized STM-1/OC-3
- Cisco 8-Port Channelized T1/E1 SPA
- Cisco 2-Port and 4-Port Clear Channel T3/E3 SPA
- Cisco 2-Port and 4-Port Channelized T3 SPA

- Cisco Multirate 10G IP Services Engines SIPs
  - Cisco 12000-SIP-600
  - Cisco 12000-SIP-401
  - Cisco 12000-SIP-501
  - Cisco 12000-SIP-601

Configuring the IPHC Slot Level Command

This section describes how to configure the IP header compression (IPHC) slot level command, which reserves the IPHC resources, enables IPHC on the line card, and defines the maximum number of TCP and non-TCP connections for the nodes. This configuration must be done before an IPHC profile can be created.

> **Note** IPHC slot level configuration is required on both the peer routers.

SUMMARY STEPS

To configure the IP header compression (IPHC) slot level, perform the following steps.

1. **config**
2. **iphc tcp connections** *max-number* **location** *node-id*
3. **iphc non-tcp connections** *max-number* **location** *node-id*
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**config** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**iphc tcp connections** *max-number* **location** *node-id* **Example:** `RP/0/0/CPU0:router(config)# iphc tcp connections 2000 location 0/1/cpu0`	Sets the maximum number of TCP connections that may be configured for IPHC on a line card. The range is 1 to 2000.
Step 3	**iphc non-tcp connections** *max-number* **location** *node-id* **Example:** `RP/0/0/CPU0:router(config)# iphc non-tcp connections 20000 location 0/1/cpu0`	Sets the maximum number of non-TCP connections that may be configured for IPHC on a line card. The range is 1 to 20000.

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.

Configuring an IPHC Profile

This section describes how to create and configure an IP header compression (IPHC) profile. This procedure is for TCP and non-TCP compression.

SUMMARY STEPS

To configure an IP header compression (IPHC) profile, perform the following steps.

1. **configure**

2. **iphc profile** *profile-name* **type** {**ietf** | **iphc**}

3. **tcp compression**

4. **tcp context absolute** *number-of-contexts*

5. **non-tcp compression**

6. **non-tcp context absolute** *number-of-contexts*

7. **rtp**

8. **refresh max-period** {*max-number* | *infinite*}

9. **refresh rtp**

10. **feedback disable**

11. **max-header** *number-of-bytes*

12. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** Example: RP/0/0/CPU0:router# configure	Enters global configuration mode.
Step 2	**iphc profile** *profile-name* **type** {**ietf** | **iphc**} Example: RP/0/0/CPU0:router(config)# iphc profile Profile_1 type iphc	Creates an IPHC profile, sets the compression format type. and enters the IPHC profile configuration mode.
Step 3	**tcp compression** Example: RP/0/0/CPU0:router(config-iphc-profile)# tcp compression	Enables TCP compression in an IPHC profile.
Step 4	**tcp context absolute** number-of-contexts Example: RP/0/0/CPU0:router(config-iphc-profile)# tcp context absolute 255	Configures the maximum number of TCP contexts that are allowed for IPHC on a line card.
Step 5	**non-tcp compression** Example: RP/0/0/CPU0:router(config-iphc-profile)# non-tcp compression	Enables non-TCP compression in an IPHC profile.
Step 6	**non-tcp context absolute** number-of-contexts Example: RP/0/0/CPU0:router(config-iphc-profile)# non-tcp context absolute 255	Configures the maximum number of non-TCP contexts that are allowed for IPHC on a line card.

	Command or Action	Purpose
Step 7	**rtp** **Example:** `RP/0/0/CPU0:router(config-iphc-profile)# rtp`	Configures Real Time Protocol (RTP) on the interface.
Step 8	**refresh max-period** {*max-number* | **infinite**} **Example:** `RP/0/0/CPU0:router(config-iphc-profile)# refresh max-period 50`	Configures the maximum number of compressed IP header packets that are exchanged on a link before the IPHC context is refreshed.
Step 9	**refresh rtp** **Example:** `RP/0/0/CPU0:router(config-iphc-profile)# refresh rtp`	Enables the configured context refresh settings for RTP packets.
Step 10	**feedback disable** **Example:** `RP/0/0/CPU0:router(config-iphc-profile)# feedback disable`	Disables the IPHC context status feedback messages on an interface.
Step 11	**max-header** *number-of-bytes* **Example:** `RP/0/0/CPU0:router(config-iphc-profile)# max-header 20`	Configures the maximum size (in bytes) of a compressed IP header.

Step 12	**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.

Enabling an IPHC Profile on an Interface

This section describes how to enable an IP header compression (IPHC) profile on an interface by attaching the profile directly to the interface.

SUMMARY STEPS

To configure to enable an IPHC profile on an interface, perform the following steps.

1. **config**
2. **interface** *type interface-path-id*
3. **encapsulation ppp**
4. **ipv4 iphc profile** *profile-name* [**mode service-policy**]
5. **service policy input** | **output** | **type** *service-policy-name*
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**config** **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 serial 0/1/0/1	Specifies the interface. **Note** Use the **show interfaces** command to see a list of all interfaces currently configured on the router. For more information about the syntax for the router, use the question mark (**?**) online help function.
Step 3	**encapsulation** {**hdlc** | **ppp** | **frame-relay** | **mfr**} **Example:** RP/0/0/CPU0:router(config-if)# encapsulation ppp	Specifies Layer 2 encapsulation for the interface.
Step 4	**ipv4 iphc profile** *profile-name* [**mode service-policy**] **Example:** RP/0/0/CPU0:router(config-if)# ipv4 iphc profile Profile_1 or RP/0/0/CPU0:router(config-if)# ipv4 iphc profile Profile_1 mode service-policy	Attaches an IPHC profile to the interface: • *profile-name*—Text name of the IPHC profile to attach to the interface. • **mode service-policy**—Specifies that the IPHC profile applies only to a QoS service policy.

Command or Action	Purpose
Step 5 **service policy output** *service-policy-name* **Example:** `RP/0/0/CPU0:router(config-if)# service policy input` `| output	type service-policy-name`	(Optional) Specifies the name of the QoS service policy to which the IPHC profile applies. Only output service policies are allowed. Used only when **mode service-policy** is specified in Step 2.

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.

Configuration Examples for Serial Interfaces

This section provides the following configuration examples:

Bringing Up and Configuring a Serial Interface with Cisco HDLC Encapsulation: Example

The following example shows how to bring up a basic serial interface with Cisco HDLC encapsulation:

```
RP/0/0/CPU0:Router#config
RP/0/0/CPU0:Router(config)# interface serial 0/3/0/0/0:0
RP/0/0/CPU0:Router(config-if)# ipv4 address 192.0.2.2 255.255.255.252
RP/0/0/CPU0:Router(config-if)# no shutdown
```

```
RP/0/0/CPU0:router(config-if)# end
Uncommitted changes found, commit them? [yes]: yes
```

The following example shows how to configure the interval between keepalive messages to be 10 seconds:

```
RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router(config)# interface serial 0/3/0/0/0:0
RP/0/0/CPU0:router(config-if)# keepalive 10
RP/0/0/CPU0:router(config-if)# commit
```

The following example shows how to modify the optional serial interface parameters:

```
RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router(config)# interface serial 0/3/0/0/0:0
RP/0/0/CPU0:Router(config-if)# serial
RP/0/0/CPU0:Router(config-if-serial)# crc 16
RP/0/0/CPU0:Router(config-if-serial)# invert
RP/0/0/CPU0:Router(config-if-serial)# scramble
RP/0/0/CPU0:Router(config-if-serial)# transmit-delay 3
RP/0/0/CPU0:Router(config-if-serial)# commit
```

The following is sample output from the **show interfaces serial** command:

```
RP/0/0/CPU0:Router# show interfaces serial 0/0/3/0/5:23
Serial0/0/3/0/5:23 is down, line protocol is down
  Hardware is Serial network interface(s)
  Internet address is Unknown
  MTU 1504 bytes, BW 64 Kbit
     reliability 143/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16,  loopback not set, keepalive set (10 sec)
  Last clearing of "show interface" counters 18:11:15
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     2764 packets input, 2816 bytes, 3046 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
             0 runts, 0 giants, 0 throttles, 0 parity
     3046 input errors, 1 CRC, 0 frame, 0 overrun, 2764 ignored, 281 abort
     2764 packets output, 60804 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

Configuring a Serial Interface with Frame Relay Encapsulation: Example

The following example shows how to create a serial interface on a SPA with Frame Relay encapsulation and a serial subinterface with a PVC on router 1:

```
RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router(config)# interface serial 0/1/0/0
RP/0/0/CPU0:router(config-if)# encapsulation frame-relay
RP/0/0/CPU0:router(config-if)#frame-relay intf-type dce
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 serial 0/1/0/0.1 point-to-point
```

```
RP/0/0/CPU0:router (config-subif)#ipv4 address 10.20.3.1/24
RP/0/0/CPU0:router (config-subif)# pvc 16
RP/0/0/CPU0:router (config-fr-vc)# encapsulation ietf
RP/0/0/CPU0:router (config-fr-vc)# commit
RP/0/0/CPU0:router(config-fr-vc)# exit
RP/0/0/CPU0:router(config-subif)# exit
RP/0/0/CPU0:router(config)# exit

RP/0/0/CPU0:router# show interface serial 0/1/0/0
Wed Oct  8 04:14:39.946 PST DST
Serial0/1/0/0 is up, line protocol is up
  Interface state transitions: 5
  Hardware is Serial network interface(s)
  Internet address is 10.20.3.1/24
  MTU 4474 bytes, BW 44210 Kbit
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation FRAME-RELAY, crc 16,
  Scrambling is disabled, Invert data is disabled
  LMI enq sent  0, LMI stat recvd 0, LMI upd recvd 0
  LMI enq recvd 880, LMI stat sent  880, LMI upd sent  0 , DCE LMI up
  LMI DLCI 1023  LMI type is CISCO  frame relay DCE
  Last clearing of "show interface" counters 02:23:04
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     858 packets input, 11154 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     858 packets output, 12226 bytes, 0 total output drops
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
```

The following example shows how to create a serial interface on a SPA with Frame Relay encapsulation and a serial subinterface with a PVC on router 2, which is connected to router 1:

```
RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router(config)# interface serial 0/1/0/1
RP/0/0/CPU0:router(config-if)# encapsulation frame-relay
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 serial 0/1/0/1.1 point-to-point
RP/0/0/CPU0:router (config-subif)#ipv4 address 10.20.3.2/24
RP/0/0/CPU0:router (config-subif)# pvc 16
RP/0/0/CPU0:router (config-fr-vc)# encapsulation ietf
RP/0/0/CPU0:router (config-fr-vc)# commit
```

```
RP/0/0/CPU0:router(config-fr-vc)# exit
RP/0/0/CPU0:router(config-subif)# exit
RP/0/0/CPU0:router(config)# exit

RP/0/0/CPU0:router# show interface serial 0/1/0/1
Wed Oct  8 04:13:45.046 PST DST
Serial0/1/0/1 is up, line protocol is up
  Interface state transitions: 7
  Hardware is Serial network interface(s)
  Internet address is Unknown
  MTU 4474 bytes, BW 44210 Kbit
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation FRAME-RELAY, crc 16,
  Scrambling is disabled, Invert data is disabled
  LMI enq sent  1110, LMI stat recvd 875, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  Last clearing of "show interface" counters 02:22:09
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     853 packets input, 12153 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     853 packets output, 11089 bytes, 0 total output drops
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
```

Configuring a Serial Interface with PPP Encapsulation: Example

The following example shows how to create and configure a serial interface with PPP encapsulation:

```
RP/0/0/CPU0:router# configure
RP/0/0/CPU0:router(config)# interface serial 0/3/0/0/0:0
RP/0/0/CPU0:router(config-if)# ipv4 address 172.18.189.38 255.255.255.224
RP/0/0/CPU0:router(config-if)# encapsulation ppp
RP/0/0/CPU0:router(config-if)# no shutdown
RP/0/0/CPU0:router(config-if)# ppp authentication chap MIS-access
RP/0/0/CPU0:router(config-if)# end
Uncommitted changes found, commit them? [yes]: yes
```

The following example shows how to configure serial interface 0/3/0/0/0:0 to allow two additional retries after an initial authentication failure (for a total of three failed authentication attempts):

```
RP/0/0/CPU0:router# configuration
RP/0/0/CPU0:router(config)# interface serial 0/3/0/0/0:0
RP/0/0/CPU0:router(config-if)# encapsulation ppp
RP/0/0/CPU0:router(config-if)# ppp authentication chap
RP/0/0/CPU0:router(config-if)# ppp max-bad-auth 3
RP/0/0/CPU0:router(config-if)# end
Uncommitted changes found, commit them? [yes]: yes
```

IPHC Configuration: Examples

This section provides the following examples:

IPHC Profile Configuration: Example

The following example shows how to configure an IPHC Profile:

```
config
  iphc tcp connections 6000 location 0/2/1
  iphc non-tcp connections 6000 location 0/2/1
  iphc profile Profile_1 type iphc
    tcp compression
    tcp context absolute 255
    non-tcp compression
    non-tcp context absolute 255
    rtp
    refresh max-period 50
    refresh rtp
    feedback disable
    max-header 20
commit
```

IPHC on a Serial Interface Configuration: Examples

Example 1

The following example shows how to enable an IP header compression (IPHC) profile on a serial interface by attaching the profile directly to the interface:

```
config
  interface serial 0/1/0/1
    encapsulation ppp
    ipv4 iphc profile Profile_1
commit
```

Example 2

The following example shows how to enable an IP header compression (IPHC) profile on an interface by specifying a QoS service policy that contains an IPHC profile:

```
config
  interface serial 0/1/0/1:1
    encapsulation ppp
    ipv4 iphc profile Profile_2 mode service-policy
    service-policy output ip_header_compression_policy_map
commit
```

IPHC on Multilink Configuration: Example

The following example shows how to configure an IP header compression (IPHC) on a multilink interface:

```
config
  interface multilink 0/4/3/0/4
    ipv4 address 10.10.10.10
    encapsulation ppp
    ipv4 iphc profile Profile_1
    commit
  interface serial 0/1/0/1:1
    encapsulation ppp
    multilink group 4
    commit
```

IPHC on a Serial Interface with MLPPP/LFI and QoS Configuration: Example

The following example shows how to configure IP header compression (IPHC) on a serial interface with LFI and by specifying a QoS service policy that contains an IPHC profile:

```
config
  interface multilink 0/4/3/0/4
    ipv4 address 10.10.10.10
      multilink
        fragment-size 128
        interleave
    ipv4 iphc profile Profile_2 mode service-policy
    service-policy output SP_2
    commit
  interface serial 0/1/0/1:2
    encapsulation ppp
    multilink group 4
    commit
```

Additional References

These sections provide references related to T3/E3 and T1/E1 controllers and serial interfaces.

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.1.2	*PVC User-to-Network Interface (UNI) Implementation Agreement - July 2000*
ANSI T1.617 Annex D	—
ITU Q.933 Annex A	—

MIBs

MIBs	MIBs Link
—	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*
RFC 2427	*Multiprotocol Interconnect Over Frame Relay*
RFC 2954	*Definitions of Managed Objects for Frame Relay Service*

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