Carrier Ethernet Configuration Guide, Cisco IOS XE Release 3S (Cisco ASR 920 Series)

First Published: 2014-07-29

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

CONTENTS

CHAPTER 1

Using Ethernet Operations Administration and Maintenance

Ethernet Operations, Administration, and Maintenance (OAM) is a protocol for installing, monitoring, and troubleshooting Ethernet metropolitan-area networks (MANs) and Ethernet WANs. It relies on a new, optional sublayer in the data link layer of the Open Systems Interconnection (OSI) model. The OAM features covered by this protocol are Discovery, Link Monitoring, Remote Fault Detection, Remote Loopback, and Cisco Proprietary Extensions.

The advent of Ethernet as a MAN and WAN technology has emphasized the necessity for integrated management for larger deployments. For Ethernet to extend into public MANs and WANs, it must be equipped with a new set of requirements on Ethernet's traditional operations, which had been centered on enterprise networks only. The expansion of Ethernet technology into the domain of service providers, where networks are substantially larger and more complex than enterprise networks and the user-base is wider, makes operational management of link uptime crucial.

Information About Using Ethernet Operations Administration and Maintenance

Ethernet OAM

Ethernet OAM is a protocol for installing, monitoring, and troubleshooting metro Ethernet networks and Ethernet WANs. It relies on a new, optional sublayer in the data link layer of the OSI model. Ethernet OAM can be implemented on any full-duplex point-to-point or emulated point-to-point Ethernet link. A system-wide implementation is not required; OAM can be deployed for part of a system; that is, on particular interfaces.

Normal link operation does not require Ethernet OAM. OAM frames, called OAM protocol data units (PDUs), use the slow protocol destination MAC address 0180.c200.0002. They are intercepted by the MAC sublayer and cannot propagate beyond a single hop within an Ethernet network.

Ethernet OAM is a relatively slow protocol with modest bandwidth requirements. The frame transmission rate is limited to a maximum of 10 frames per second; therefore, the impact of OAM on normal operations is negligible. However, when link monitoring is enabled, the CPU must poll error counters frequently. In this case, the required CPU cycles will be proportional to the number of interfaces that have to be polled.

Two major components, the OAM client and the OAM sublayer, make up Ethernet OAM. The following two sections describe these components.

OAM Client

The OAM client is responsible for establishing and managing Ethernet OAM on a link. The OAM client also enables and configures the OAM sublayer. During the OAM discovery phase, the OAM client monitors OAM PDUs received from the remote peer and enables OAM functionality on the link based on local and remote state as well as configuration settings. Beyond the discovery phase (at steady state), the OAM client is responsible for managing the rules of response to OAM PDUs and managing the OAM remote loopback mode.

OAM Sublayer

The OAM sublayer presents two standard IEEE 802.3 MAC service interfaces: one facing toward the superior sublayers, which include the MAC client (or link aggregation), and the other interface facing toward the subordinate MAC control sublayer. The OAM sublayer provides a dedicated interface for passing OAM control information and OAM PDUs to and from a client.

The OAM sublayer is made up of three components: control block, multiplexer, and packet parser (p-parser). Each component is described in the following sections.

Control Block

The control block provides the interface between the OAM client and other blocks internal to the OAM sublayer. The control block incorporates the discovery process, which detects the existence and capabilities of remote OAM peers. It also includes the transmit process that governs the transmission of OAM PDUs to the multiplexer and a set of rules that govern the receipt of OAM PDUs from the p-parser.

Multiplexer

The multiplexer manages frames generated (or relayed) from the MAC client, control block, and p-parser. The multiplexer passes through frames generated by the MAC client untouched. It passes OAM PDUs generated by the control block to the subordinate sublayer; for example, the MAC sublayer. Similarly, the multiplexer passes loopback frames from the p-parser to the same subordinate sublayer when the interface is in OAM remote loopback mode.

P-Parser

The p-parser classifies frames as OAM PDUs, MAC client frames, or loopback frames and then dispatches each class to the appropriate entity. OAM PDUs are sent to the control block. MAC client frames are passed to the superior sublayer. Loopback frames are dispatched to the multiplexer.

Benefits of Ethernet OAM

Ethernet OAM provides the following benefits:

- Competitive advantage for service providers.
- Standardized mechanism to monitor the health of a link and perform diagnostics.

Note REP traps are prioritized when both Ethernet OAM and REP traps are configured on the same port. If you want to view Ethernet OAM logs, then you must disable REP configurations.

Cisco Implementation of Ethernet OAM

The Cisco implementation of Ethernet OAM consists of the Ethernet OAM shim and the Ethernet OAM module.

The Ethernet OAM shim is a thin layer that connects the Ethernet OAM module and the platform code. It is implemented in the platform code (driver). The shim also communicates port state and error conditions to the Ethernet OAM module via control signals.

The Ethernet OAM module, implemented within the control plane, handles the OAM client as well as control block functionality of the OAM sublayer. This module interacts with the CLI and Simple Network Management Protocol (SNMP)/programmatic interface via control signals. In addition, this module interacts with the Ethernet OAM shim through OAM PDU flows.

OAM Features

The OAM features as defined by IEEE 802.3ah, *Ethernet in the First Mile*, are discovery, Link Monitoring, Remote Fault Detection, Remote Loopback, and Cisco Proprietary Extensions.

Discovery

Discovery is the first phase of Ethernet OAM and it identifies the devices in the network and their OAM capabilities. Discovery uses information OAM PDUs. During the discovery phase, the following information is advertised within periodic information OAM PDUs:

- OAM mode—Conveyed to the remote OAM entity. The mode can be either active or passive and can be used to determine device functionality.
- OAM configuration (capabilities)—Advertises the capabilities of the local OAM entity. With this information a peer can determine what functions are supported and accessible; for example, loopback capability.
- OAM PDU configuration—Includes the maximum OAM PDU size for receipt and delivery. This information along with the rate limiting of 10 frames per second can be used to limit the bandwidth allocated to OAM traffic.
- Platform identity—A combination of an organization unique identifier (OUI) and 32-bits of vendor-specific information. OUI allocation, controlled by the IEEE, is typically the first three bytes of a MAC address.

Discovery includes an optional phase in which the local station can accept or reject the configuration of the peer OAM entity. For example, a node may require that its partner support loopback capability to be accepted into the management network. These policy decisions may be implemented as vendor-specific extensions.

If you configure an error-disable action on an Ethernet OAM monitor command, it triggers the interface admin down when error frames exceed the threshold. Un-shut the interface manually to recover the data.

Link Monitoring

Link monitoring in Ethernet OAM detects and indicates link faults under a variety of conditions. Link monitoring uses the event notification OAM PDU and sends events to the remote OAM entity when there are problems detected on the link. The error events include the following:

- Error Symbol Period (error symbols per second)—The number of symbol errors that occurred during a specified period exceeded a threshold. These errors are coding symbol errors.
- Error Frame (error frames per second)—The number of frame errors detected during a specified period exceeded a threshold.
- Error Frame Period (error frames per *n* frames)—The number of frame errors within the last n frames has exceeded a threshold.
- Error Frame Seconds Summary (error seconds per *m* seconds)—The number of error seconds (1-second intervals with at least one frame error) within the last m seconds has exceeded a threshold.

Since IEEE 802.3ah OAM does not provide a guaranteed delivery of any OAM PDU, the event notification OAM PDU may be sent multiple times to reduce the probability of a lost notification. A sequence number is used to recognize duplicate events.

Remote Failure Indication

Faults in Ethernet connectivity that are caused by slowly deteriorating quality are difficult to detect. Ethernet OAM provides a mechanism for an OAM entity to convey these failure conditions to its peer via specific flags in the OAM PDU. The following failure conditions can be communicated:

- Link Fault—Loss of signal is detected by the receiver; for instance, the peer's laser is malfunctioning. A link fault is sent once per second in the information OAM PDU. Link fault applies only when the physical sublayer is capable of independently transmitting and receiving signals.
- Dying Gasp—An unrecoverable condition has occurred; for example, when an interface is shut down. This type of condition is vendor specific. A notification about the condition may be sent immediately and continuously.

  For more information on Dying Gasp, see the Dying Gasp Support for Loss of Power Supply Through SNMP, Syslog and Ethernet OAM chapter in the Cisco NCS 520 Series Router Configuration Guide.
- Critical Event—An unspecified critical event has occurred. This type of event is vendor specific. A critical event may be sent immediately and continuously.

Remote Loopback

An OAM entity can put its remote peer into loopback mode using the loopback control OAM PDU. Loopback mode helps an administrator ensure the quality of links during installation or when troubleshooting. In loopback mode, every frame received is transmitted back on the same port except for OAM PDUs and pause frames. The periodic exchange of OAM PDUs must continue during the loopback state to maintain the OAM session.

The loopback command is acknowledged by responding with an information OAM PDU with the loopback state indicated in the state field. This acknowledgement allows an administrator, for example, to estimate if a network segment can satisfy a service-level agreement. Acknowledgement makes it possible to test delay, jitter, and throughput.

When an interface is set to the remote loopback mode the interface no longer participates in any other Layer 2 or Layer 3 protocols; for example Spanning Tree Protocol (STP) or Open Shortest Path First (OSPF). The reason is that when two connected ports are in a loopback session, no frames other than the OAM PDUs are sent to the CPU for software processing. The non-OAM PDU frames are either looped back at the MAC level or discarded at the MAC level.

From a user's perspective, an interface in loopback mode is in a link-up state.

Cisco Vendor-Specific Extensions

Ethernet OAM allows vendors to extend the protocol by allowing them to create their own type-length-value (TLV) fields.

OAM Messages

Ethernet OAM messages or OAM PDUs are standard length, untagged Ethernet frames within the normal frame length bounds of 64 to 1518 bytes. The maximum OAM PDU frame size exchanged between two peers is negotiated during the discovery phase.

OAM PDUs always have the destination address of slow protocols (0180.c200.0002) and an Ethertype of 8809. OAM PDUs do not go beyond a single hop and have a hard-set maximum transmission rate of 10 OAM PDUs per second. Some OAM PDU types may be transmitted multiple times to increase the likelihood that they will be successfully received on a deteriorating link.

Four types of OAM messages are supported:

- Information OAM PDU--A variable-length OAM PDU that is used for discovery. This OAM PDU includes local, remote, and organization-specific information.
- Event notification OAM PDU--A variable-length OAM PDU that is used for link monitoring. This type of OAM PDU may be transmitted multiple times to increase the chance of a successful receipt; for example, in the case of high-bit errors. Event notification OAM PDUs also may include a time stamp when generated.
- Loopback control OAM PDU--An OAM PDU fixed at 64 bytes in length that is used to enable or disable the remote loopback command.
- Vendor-specific OAM PDU--A variable-length OAM PDU that allows the addition of vendor-specific extensions to OAM.

IEEE 802.3ah Link Fault RFI Support

The IEEE 802.3ah Link Fault RFI Support feature provides a per-port configurable option that moves a port into a blocking state when an OAM PDU control request packet is received with the Link Fault Status flag set. In the blocking state, the port can continue to receive OAM PDUs, detect remote link status, and automatically recover when the remote link becomes operational. When an OAM PDU is received with the Link Fault Status flag set to zero or FALSE, the port is enabled and all VLANs configured on the port are set to "forwarding."

Note If you configure the Ethernet OAM timeout period to be the minimum allowable value of 2 seconds, the Ethernet OAM session may be dropped briefly when the port transitions from blocked to unblocked. This action will not occur by default; the default timeout value is 5 seconds.

Before the release of the IEEE 802.3ah Link Fault RFI Support feature, when an OAM PDU control request packet was received with the Link Fault Status flag set, one of three actions was taken:

- A warning message was displayed or logged, and the port remained operational.
- The Link Fault Status flag was ignored.

A new keyword, **error-block-interface**, for the CLI command **ethernet oam remote-failure action** is introduced with the IEEE 802.3ah Link Fault RFI Support feature. For detailed information about this command, see the *Cisco IOS Carrier Ethernet Command Reference*.

Ethernet Connectivity Fault Management

Ethernet connectivity fault management (CFM) is an end-to-end per-service-instance Ethernet layer OAM protocol that includes proactive connectivity monitoring, fault verification, and fault isolation. End to end can be provider edge (PE) to PE or customer edge (CE) to CE. Per service instance means per VLAN.

For more information about Ethernet CFM, see Ethernet Connectivity Fault Management .

How to Set Up and Configure Ethernet Operations Administration and Maintenance

Enabling Ethernet OAM on an Interface

Ethernet OAM is by default disabled on an interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ethernet oam** [**max-rate** *oampdus* | **min-rate** *num-seconds*| **mode** {**active** | **passive**} | **timeout** *seconds*]
5. **exit**

DETAILED STEPS

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

	Command or Action	Purpose
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 3/8`	Specifies an interface and enters interface configuration mode.
Step 4	**ethernet oam** [**max-rate** *oampdus* | **min-rate** *num-seconds*| **mode** {**active** | **passive**} | **timeout** *seconds*] **Example:** `Device(config-if)# ethernet oam`	Enables Ethernet OAM.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.

Disabling and Enabling a Link Monitoring Session

Link monitoring is enabled by default when you enable Ethernet OAM. Perform these tasks to disable and enable link monitoring sessions:

Disabling a Link Monitoring Session

Perform this task to disable a link monitoring session.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ethernet oam** [**max-rate** *oampdus* | **min-rate** *num-seconds*| **mode** {**active** | **passive**} | **timeout** *seconds*]
5. **no ethernet oam link-monitor supported**
6. **exit**

DETAILED STEPS

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

	Command or Action	Purpose
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitEthernet 3/8`	Specifies an interface and enters interface configuration mode.
Step 4	**ethernet oam** [**max-rate** *oampdus* | **min-rate** *num-seconds*| **mode** {**active** | **passive**} | **timeout** *seconds*] **Example:** `Device(config-if)# ethernet oam`	Enables Ethernet OAM.
Step 5	**no ethernet oam link-monitor supported** **Example:** `Device(config-if)# no ethernet oam link-monitor supported`	Disables link monitoring on the interface.
Step 6	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.

Enabling a Link Monitoring Session

Perform this task to reenable a link monitoring session after it was previously disabled.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ethernet oam link-monitor supported**
5. **exit**

DETAILED STEPS

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

	Command or Action	Purpose
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitEthernet 3/8`	Specifies an interface and enters interface configuration mode.
Step 4	**ethernet oam link-monitor supported** **Example:** `Device(config-if)# ethernet oam link-monitor supported`	Enables link monitoring on the interface.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.

Stopping and Starting Link Monitoring Operations

Link monitoring operations start automatically when Ethernet OAM is enabled on an interface. When link monitoring operations are stopped, the interface does not actively send or receive event notification OAM PDUs. The tasks in this section describe how to stop and start link monitoring operations.

Stopping Link Monitoring Operations

Perform this task to stop link monitoring operations.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ethernet oam** [**max-rate** *oampdus* | **min-rate** *num-seconds*| **mode** {**active** | **passive**} | **timeout** *seconds*]
5. **no ethernet oam link-monitor on**
6. **exit**

DETAILED STEPS

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

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 3/8`	Specifies an interface and enters interface configuration mode.
Step 4	**ethernet oam** [**max-rate** *oampdus* | **min-rate** *num-seconds*| **mode** {**active** | **passive**} | **timeout** *seconds*] **Example:** `Device(config-if)# ethernet oam`	Enables Ethernet OAM.
Step 5	**no ethernet oam link-monitor on** **Example:** `Device(config-if)# no ethernet oam link-monitor on`	Stops link monitoring operations.
Step 6	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.

Starting Link Monitoring Operations

Perform this task to start link monitoring operations.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ethernet oam link-monitor on**
5. **exit**

DETAILED STEPS

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

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 3/8`	Specifies an interface and enters interface configuration mode.
Step 4	**ethernet oam link-monitor on** **Example:** `Device(config-if)# ethernet oam link-monitor on`	Starts link monitoring operations.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.

Configuring Link Monitoring Options

Perform this optional task to specify link monitoring options. Steps 4 through 10 can be performed in any sequence.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ethernet oam** [**max-rate** *oampdus* | **min-rate** *num-seconds*| **mode** {**active** | **passive**} | **timeout** *seconds*]
5. **ethernet oam link-monitor frame** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *milliseconds*}
6. **ethernet oam link-monitor frame-period** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *frames*}
7. **ethernet oam link-monitor frame-seconds** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *milliseconds*}
8. **ethernet oam link-monitor receive-crc** {**threshold** {**high** {*high-frames* | **none**} | **low** *low-frames*} | **window** *milliseconds*}
9. **ethernet oam link-monitor transmit-crc** {**threshold** {**high** {*high-frames* | **none**} | **low** *low-frames*} | **window** *milliseconds*}
10. **ethernet oam link-monitor symbol-period** {**threshold** {**high** {**none** | *high-symbols*} | **low** *low-symbols*} | **window** *symbols*}
11. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitEthernet 3/8`	Identifies the interface and enters interface configuration mode.
Step 4	**ethernet oam** [**max-rate** *oampdus* | **min-rate** *num-seconds*| **mode** {**active** | **passive**} | **timeout** *seconds*] **Example:** `Device(config-if)# ethernet oam`	Enables Ethernet OAM.

Step 5	**ethernet oam link-monitor frame** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *milliseconds*}
	Example:
	`Device(config-if)# ethernet oam link-monitor frame window 399`
		Configures a number for error frames that when reached triggers an action.

Step 6	**ethernet oam link-monitor frame-period** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *frames*}
	Example:
	`Device(config-if)# ethernet oam link-monitor frame-period threshold high 599`
		Configures a number of frames to be polled.
		Frame period is a user-defined parameter.

Step 7	**ethernet oam link-monitor frame-seconds** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *milliseconds*}
	Example:
	`Device(config-if)# ethernet oam link-monitor frame-seconds window 699`
		Configures a period of time in which error frames are counted.

	Command or Action	Purpose

Step 8	**ethernet oam link-monitor receive-crc** {**threshold** {**high** {*high-frames* | **none**} | **low** *low-frames*} | **window** *milliseconds*}
	Example:
	`Device(config-if)# ethernet oam link-monitor receive-crc window 99`
		Configures an Ethernet OAM interface to monitor ingress frames with cyclic redundancy check (CRC) errors for a period of time.

Step 9	**ethernet oam link-monitor transmit-crc** {**threshold** {**high** {*high-frames* | **none**} | **low** *low-frames*} | **window** *milliseconds*}
	Example:
	`Device(config-if)# ethernet oam link-monitor transmit-crc threshold low 199`
		Configures an Ethernet OAM interface to monitor egress frames with CRC errors for a period of time.

Step 10	**ethernet oam link-monitor symbol-period** {**threshold** {**high** {**none** | *high-symbols*} | **low** *low-symbols*} | **window** *symbols*}
	Example:
	`Device(config-if)# ethernet oam link-monitor symbol-period threshold high 299`
		Configures a threshold or window for error symbols, in number of symbols.

Step 11	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.

Example

```
Device# configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.

Device(config)# interface gigabitEthernet 3/8
Device(config-if)#
Device(config-if)# ethernet oam

Device(config-if)# ethernet oam link-monitor high-threshold action error-disable-interface
Device(config-if)# ethernet oam link-monitor frame window 399
Device(config-if)# ethernet oam link-monitor frame-period threshold high 599
Device(config-if)# ethernet oam link-monitor frame-seconds window 699
Device(config-if)# ethernet oam link-monitor receive-crc window 99
Device(config-if)# ethernet oam link-monitor transmit-crc threshold low 199
Device(config-if)# ethernet oam link-monitor symbol-period threshold high 299
Device(config-if)# exit
Device# show running-config

Building configuration...
Current configuration : 5613 bytes
```

```
!
!
version 12.2
!
!
.
.
.
!
!
interface GigabitEthernet3/8
 no ip address
 ethernet oam link-monitor high-threshold action error-disable-interface
 ethernet oam link-monitor frame window 399
 ethernet oam link-monitor frame-period threshold high 599
 ethernet oam link-monitor frame-seconds window 699
 ethernet oam link-monitor receive-crc window 99
 ethernet oam link-monitor transmit-crc threshold low 199
 ethernet oam link-monitor symbol-period threshold high 299
 ethernet oam
```

Configuring Global Ethernet OAM Options Using a Template

Perform this task to create a template to use for configuring a common set of options on multiple Ethernet OAM interfaces. Steps 4 through 10 are optional and can be performed in any sequence. These steps may also be repeated to configure different options.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **template** *template-name*
4. **ethernet oam link-monitor receive-crc** {**threshold** {**high** {*high-frames* | **none**} | **low** *low-frames*} | **window** *milliseconds*}
5. **ethernet oam link-monitor transmit-crc** {**threshold** {**high** {*high-frames* | **none**} | **low** *low-frames*} | **window** *milliseconds*}
6. **ethernet oam link-monitor symbol-period** {**threshold** {**high** {**none** | *high-symbols*} | **low** *low-symbols*} | **window** *symbols*}
7. **ethernet oam link-monitor frame** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *milliseconds*}
8. **ethernet oam link-monitor frame-period** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *frames*}
9. **ethernet oam link-monitor frame-seconds** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *milliseconds*}
10. **exit**
11. **interface** *type number*
12. **source template** *template-name*
13. **exit**
14. **exit**
15. **show running-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**template** *template-name* **Example:** `Device(config)# template oam-temp`	Configures a template and enters template configuration mode.

Step 4	**ethernet oam link-monitor receive-crc** {**threshold** {**high** {*high-frames* | **none**} | **low** *low-frames*} | **window** *milliseconds*}
	Example:
	`Device(config-template)# ethernet oam link-monitor receive-crc window 99`
		Configures an Ethernet OAM interface to monitor ingress frames with CRC errors for a period of time.

Step 5	**ethernet oam link-monitor transmit-crc** {**threshold** {**high** {*high-frames* | **none**} | **low** *low-frames*} | **window** *milliseconds*}
	Example:
	`Device(config-template)# ethernet oam link-monitor transmit-crc threshold low 199`
		Configures an Ethernet OAM interface to monitor egress frames with CRC errors for a period of time.

Step 6	**ethernet oam link-monitor symbol-period** {**threshold** {**high** {**none** | *high-symbols*} | **low** *low-symbols*} | **window** *symbols*}
	Example:
	`Device(config-template)# ethernet oam link-monitor symbol-period threshold high 299`
		Configures a threshold or window for error symbols, in number of symbols.

Step 7	**ethernet oam link-monitor frame** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *milliseconds*}
	Example:
	`Device(config-template)# ethernet oam link-monitor frame window 399`
		Configures a number for error frames that when reached triggers an action.

	Command or Action	Purpose
Step 8	**ethernet oam link-monitor frame-period** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *frames*} **Example:** `Device(config-template)# ethernet oam link-monitor frame-period threshold high 599`	Configures a number of frames to be polled. Frame period is a user-defined parameter.
Step 9	**ethernet oam link-monitor frame-seconds** {**threshold** {**high** {**none** | *high-frames*} | **low** *low-frames*} | **window** *milliseconds*} **Example:** `Device(config-template)# ethernet oam link-monitor frame-seconds window 699`	Configures a period of time in which error frames are counted.
Step 10	**exit** **Example:** `Device(config-template)# exit`	Returns to global configuration mode.
Step 11	**interface** *type number* **Example:** `Device(config)# interface gigabitEthernet 3/8`	Identifies the interface on which to use the template and enters interface configuration mode.
Step 12	**source template** *template-name* **Example:** `Device(config-if)# source template oam-temp`	Applies to the interface the options configured in the template.
Step 13	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
Step 14	**exit** **Example:** `Device(config)# exit`	Returns to privileged EXEC mode.
Step 15	**show running-config** **Example:** `Device# show running-config`	Displays the updated running configuration.

Configuring a Port for Link Fault RFI Support

Perform this task to put a port into a blocking state when an OAM PDU control request packet is received with the Link Fault Status flag set.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ethernet oam remote-failure** {**critical-event** | **dying-gasp** | **link-fault**} **action** {**error-block-interface** }
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface fastethernet 1/2`	Enters interface configuration mode.

Step 4	**ethernet oam remote-failure** {**critical-event** | **dying-gasp** | **link-fault**} **action** {**error-block-interface** }
	Example:
	`Device(config-if)# ethernet oam remote-failure critical-event action error-block-interface`
		Sets the interface to the blocking state when a critical event occurs.

Step 5	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.

Configuration Examples for Ethernet Operations Administration and Maintenance

The following example shows how to configure Ethernet OAM options using a template and overriding that configuration by configuring an interface. In this example, the network supports a Gigabit Ethernet interface between the customer edge device and provider edge device.

```
! Configure a global OAM template for both PE and CE configuration.
!
Device(config)# template oam
Device(config-template)# ethernet oam link-monitor symbol-period threshold low 10
Device(config-template)# ethernet oam link-monitor symbol-period threshold high 100
Device(config-template)# ethernet oam link-monitor frame window 100
Device(config-template)# ethernet oam link-monitor frame threshold low 10
Device(config-template)# ethernet oam link-monitor frame threshold high 100
Device(config-template)# ethernet oam link-monitor frame-period window 100
Device(config-template)# ethernet oam link-monitor frame-period threshold low 10
Device(config-template)# ethernet oam link-monitor frame-period threshold high 100
Device(config-template)# ethernet oam link-monitor frame-seconds window 1000
Device(config-template)# ethernet oam link-monitor frame-seconds threshold low 10
Device(config-template)# ethernet oam link-monitor frame-seconds threshold high 100
Device(config-template)# ethernet oam link-monitor receive-crc window 100
Device(config-template)# ethernet oam link-monitor receive-crc threshold high 100
Device(config-template)# ethernet oam link-monitor transmit-crc window 100
Device(config-template)# ethernet oam link-monitor transmit-crc threshold high 100

Device(config-template)# exit
!
! Enable Ethernet OAM on the CE interface
!
Device(config)# interface gigabitethernet 4/1/1
Device(config-if)# ethernet oam
!
! Apply the global OAM template named "oam" to the interface.
!
Device(config-if)# source template oam
!
! Configure any interface-specific link monitoring commands to override the template
configuration. The following example disables the high threshold link monitoring for receive
 CRC errors.
!
Device(config-if)# ethernet oam link-monitor receive-crc threshold high none
!
! Enable Ethernet OAM on the PE interface
!
Device(config)# interface gigabitethernet 8/1/1
Device(config-if)# ethernet oam
!
! Apply the global OAM template named "oam" to the interface.
!
Device(config-if)# source template oam
```

The following examples show how to verify various Ethernet OAM configurations and activities.

Verifying an OAM Session

The following example shows that the local OAM client, Gigabit Ethernet interface Gi6/1/1, is in session with a remote client with MAC address 0012.7fa6.a700 and OUI 00000C, which is the OUI for Cisco. The remote client is in active mode and has established capabilities for link monitoring and remote loopback for the OAM session.

```
Device# show ethernet oam summary
Symbols:          * - Master Loopback State,  # - Slave Loopback State
Capability codes: L - Link Monitor,  R - Remote Loopback
                  U - Unidirection,  V - Variable Retrieval
  Local                       Remote
Interface       MAC Address    OUI    Mode    Capability
 Gi6/1/1       0012.7fa6.a700 00000C active      L R
```

Verifying OAM Discovery Status

The following example shows how to verify OAM discovery status of a local client and a remote peer:

```
Device# show ethernet oam discovery interface gigabitethernet6/1/1
GigabitEthernet6/1/1
Local client
------------
  Administrative configurations:
    Mode:              active
    Unidirection:      not supported
    Link monitor:      supported (on)
    Remote loopback:   not supported
    MIB retrieval:     not supported
    Mtu size:          1500
  Operational status:
Port status:       operational
    Loopback status:   no loopback
    PDU permission:    any
    PDU revision:      1
Remote client
-------------
  MAC address: 0030.96fd.6bfa
  Vendor(oui): 0x00 0x00 0x0C (cisco)
  Administrative configurations:

   Mode:              active
   Unidirection:      not supported
   Link monitor:      supported
   Remote loopback:   not supported
   MIB retrieval:     not supported
   Mtu size:          1500
```

Verifying Information OAMPDU and Fault Statistics

The following example shows how to verify statistics for information OAM PDUs and local and remote faults:

```
Device# show ethernet oam statistics interface gigabitethernet6/1/1
GigabitEthernet6/1/1
Counters:
---------
Information OAMPDU Tx                    : 588806
Information OAMPDU Rx                    : 988
Unique Event Notification OAMPDU Tx      : 0
Unique Event Notification OAMPDU Rx      : 0
```

```
Duplicate Event Notification OAMPDU TX  : 0
Duplicate Event Notification OAMPDU RX  : 0
Loopback Control OAMPDU Tx              : 1
Loopback Control OAMPDU Rx              : 0
Variable Request OAMPDU Tx              : 0
Variable Request OAMPDU Rx              : 0
Variable Response OAMPDU Tx             : 0
Variable Response OAMPDU Rx             : 0
Cisco OAMPDU Tx                         : 4
Cisco OAMPDU Rx                         : 0
Unsupported OAMPDU Tx                   : 0
Unsupported OAMPDU Rx                   : 0
Frames Lost due to OAM                  : 0
Local Faults:
-------------
0 Link Fault records
2 Dying Gasp records
Total dying gasps       : 4
Time stamp              : 00:30:39
Total dying gasps       : 3
Time stamp              : 00:32:39
0 Critical Event records
Remote Faults:
--------------
0 Link Fault records
0 Dying Gasp records
0 Critical Event records
Local event logs:
-----------------
0 Errored Symbol Period records
0 Errored Frame records
0 Errored Frame Period records
0 Errored Frame Second records
Remote event logs:
------------------
0 Errored Symbol Period records
0 Errored Frame records
0 Errored Frame Period records
0 Errored Frame Second records
```

Verifying Link Monitoring Configuration and Status

The following example shows how to verify link monitoring configuration and status on the local client. The highlighted Status field in the example shows that link monitoring status is supported and enabled (on).

```
Device# show ethernet oam status interface gigabitethernet6/1/1
GigabitEthernet6/1/1
General
-------
  Mode:                  active
  PDU max rate:          10 packets per second
  PDU min rate:          1 packet per 1 second
  Link timeout:          5 seconds
  High threshold action: no action
Link Monitoring
---------------
  Status: supported (on)
  Symbol Period Error
    Window:              1 million symbols
    Low threshold:       1 error symbol(s)
    High threshold:      none
  Frame Error
    Window:              10 x 100 milliseconds
    Low threshold:       1 error frame(s)
    High threshold:      none
Frame Period Error
    Window:              1 x 100,000 frames
    Low threshold:       1 error frame(s)
    High threshold:      none
  Frame Seconds Error
    Window:              600 x 100 milliseconds
    Low threshold:       1 error second(s)
    High threshold:      none
```

Verifying Status of a Remote OAM Client

The following example shows that the local client interface Gi6/1/1 is connected to a remote client. Note the values in the Mode and Capability fields.

```
Device# show ethernet oam summary
Symbols:          * - Master Loopback State,  # - Slave Loopback State
Capability codes: L - Link Monitor,  R - Remote Loopback
                  U - Unidirection,  V - Variable Retrieval
  Local                       Remote
Interface       MAC Address    OUI    Mode    Capability
 Gi6/1/1       0012.7fa6.a700 00000C active      L R
```

CHAPTER 2

Trunk EFP Support

The Trunk EFP Support feature provides support for Ethernet flow points (EFPs) on trunk ports. A trunk port allows a range of VLANs to be forwarded on a given interface while still maintaining data-plane segmentation between the VLANs.

Restrictions for Trunk EFP Support

- The **rewrite ingress tag pop 1 symmetric** command is the only **rewrite** command that is supported for trunk EFP configurations. The **rewrite ingress tag pop 1 symmetric** command must be included in the configuration when the Trunk EFP Support feature is enabled.
- A bridge-domain number that is part of a trunk EFP configuration cannot be shared by other EFPs under the same port or interface.
- Only one trunk EFP can be configured under one port or interface.
- All features configured on a trunk EFP (other than encapsulations and bridge-domain assignments) are applied uniformly to all VLANs and bridge domains. If a feature requires VLAN-specific or bridge-domain-specific configuration values, the feature cannot be applied on the trunk EFP. Those special VLANs or bridge domains must be removed from the EFP trunk to form individual EFPs.
- Trunk EFP MET supports a maximum of 4078 VLANs and the maximum threshold supported is 20480.
- Untagged EFP should be added to the BDI when untagged packets are directed towards the interface to avoid packets punting to host queue.

Restrictions for Trunk EFP with Encapsulation from Bridge Domain

- When an EFP is created on an interface followed by a TEFP with encapsulation from bridge domain (BD), all the BDs in the switch gets added to the TEFP with encapsulation from BD except the ones present in the EFP configured .
- You cannot create an EFP or TEFP after configuring TEFP with encapsulation from BD. It is recommneded that TEFP with encapsulation from BD should be the last EFP created on an interface.
- You cannot make changes to EFP after you have configured TEFP with encapsulation from BD. If you need to edit the EFP, you must first remove the TEFP with encapsulation from BD and then edit the TEFP.
- You cannot convert a TEFP into a TEFP with encapsulation from BD or vice versa.
- It is recommended to have a service instance ID of the TEFP with encapsulation from BD greater than the ID of any other EFP configured on that interface.
- You must maintain some delay when detaching and attaching the scaled TEFP with encapsulation from BD configurations.
- On an access interface having both EFP and TEFP or TEFP with encapsulation from BD configured, any data traffic with VLAN ID equal to bridge domain of EFP is flooded if the VLAN ID present in the data traffic does not match the encapsulation values present in the EFP and TEFP with encapsulation from BD.

Information About Trunk EFP Support

Benefits of Trunk EFP Support

The Carrier Ethernet infrastructure supports the following types of Ethernet flow points (EFPs):

- Static EFPs that are user-configurable.
- Dynamic EFPs that are created and maintained during a Cisco Intelligent Services Gateway (ISG) session.

With this feature, a new EFP type has been added that is intended for use on a trunk port.

A trunk port allows a range of VLANs to be forwarded on a given interface while maintaining data-plane segmentation between the VLANs.

Like a static EFP, this new type of EFP is user-configurable via the **service instance trunk** command, the **encapsulation** command, and the **bridge-domain from-encapsulation** command when the Trunk EFP Support feature is enabled.

Ethernet Flow Points

An Ethernet flow point (EFP) is a forwarding decision point in the provider edge (PE) router, which gives network designers flexibility to make many Layer 2 flow decisions within the interface. Many EFPs can be configured on a single physical port. (The number varies from one device to another.) EFPs are the logical demarcation points of an Ethernet virtual connection (EVC) on an interface. An EVC that uses two or more user network interfaces (UNIs) requires an EFP on the associated ingress and egress interfaces of every device that the EVC passes through.

EFPs can be configured on any Layer 2 traffic port; however, they are usually configured on UNI ports. The following parameters (matching criteria) can be configured on the EFP:

- Frames of a specific VLAN, a VLAN range, or a list of VLANs (100-150 or 100,103,110)
- Frames with no tags (untagged)
- Frames with identical double-tags (VLAN tags) as specified
- Frames with identical Class of Service (CoS) values

A frame passes each configured match criterion until the correct matching point is found. If a frame does not fit any of the matching criteria, it is dropped. Default criteria can be configured to avoid dropping frames.

You can configure a new type of TEFP called TEFP with encapsulation from bridge domain (BD). All the BDs configured on the switch are part of the VLAN list of the encapsulated TEFP. The TEFP is encapsulated using the **encapsulation dot1q from-bd** command. The feature brings about the following interaction between the Ethernet-EFP and Layer2-bridge domain components:

- If BDs exist in the system and a TEFP with encapsulation from bridge domain is created, then all the BDs get added to the VLAN list of TEFP with encapsulation from bridge domain.
- If TEFP with encapsulation from bridge domain exists in the system and a new BD is created, then the BD is added to the VLAN list of all the TEFP with encapsulation from bridge domain in the system.
- If TEFP with encapsulation from bridge domain exists in the system and a BD gets deleted, and if the deleted BD is not part of an existing TEFP or EFP then it gets deleted from all the TEFP with encapsulation from bridge domain in the system.

The following types of commands can be used in an EFP:

- Rewrite commands—In each EFP, VLAN tag management can be specified with the following actions:
    - Pop—1) pops out a tag; 2) pops out two tags
    - Push—1) pushes in a tag; 2) pushes in two tags
    - Translate—1 to 1) changes a tag value; 1 to 2) pops one tag and pushes two tags; 2 to 1) pops two tags and pushes one tag; 2 to 2) changes the value for two tags
- Forwarding commands—Each EFP specifies the forwarding command for the frames that enter the EFP. Only one forwarding command can be configured per EFP. The forwarding options are as follows:
    - Layer 2 point-to-point forwarding to a pseudowire tunnel
    - Multipoint bridge forwarding to a bridge domain entity
    - Local switch-to-switch forwarding between two different interfaces

- Feature commands—In each EFP, the QoS features or parameters can be changed and the ACL can be updated.

Trunk Ports

An Ethernet interface can be configured as a trunk port (interface). A trunk port, also known as a trunk, is a point-to-point link between a networking device and another networking device. Trunks carry the traffic of multiple VLANs over a single link and allow you to extend VLANs across an entire network. A trunk port configured on the interface with two or more VLANs can carry traffic for several VLANs simultaneously.

To correctly deliver the traffic on a trunk port with several VLANs, the device uses the IEEE 802.1Q encapsulation or tagging method.

How to Enable Trunk EFP Support

Enabling Trunk EFP Support

To enable Ethernet flow point (EFP) support on a trunk port or trunk interface, complete the following steps.

Note TEFP is supported on a PC interface and on a Gigabit interface. The procedure listed below is for TEFP configuration on a PC interface. Similar procedure is used for TEFP configuration on a gigabit interface.

Note When configuring TEFP on a port-channel interface, ensure that the port interface is always up.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *port-channel number*
4. **service instance trunk** *id* **ethernet**
5. **encapsulation dot1q** {**from-bd** |*vlan-id* [**,** *vlan-id* [- *vlan-d*]]}
6. **rewrite ingress tag pop 1 symmetric**
7. **bridge-domain from-encapsulation**
8. **no shutdown**
9. **end**

DETAILED STEPS

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

	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *port-channel number* **Example:** `Device(config)# interface port-channel 1`	Configures the interface and enters interface configuration mode.
Step 4	**service instance trunk** *id* **ethernet** **Example:** `Device(config-if)# service instance trunk 1 ethernet`	Configures an Ethernet service instance on an interface and enters Ethernet service configuration mode.

Step 5	**encapsulation dot1q** {**from-bd** |*vlan-id* [**,** *vlan-id* [- *vlan-d*]]}
	Example:
	`Device(config-if-srv)# encapsulation dot1q 1-5, 7, 9-12`
	`Device(config-if-srv)# encapsulation dot1q from-bd`
		Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance.

Step 6	**rewrite ingress tag pop 1 symmetric** **Example:** `Device(config-if-srv)# rewrite ingress tag pop 1 symmetric`	Specifies the encapsulation adjustment to be performed on a frame that is entering a service instance.
Step 7	**bridge-domain from-encapsulation** **Example:** `Device(config-if-srv)# bridge-domain from-encapsulation`	Creates a list of bridge domains for an EFP trunk port using the bridge-domain IDs derived from the encapsulation VLAN numbers.
Step 8	**no shutdown** **Example:** `Device(config-if-srv)# no shutdown`	Disables shutdown and keeps the interface or port active.

Step 9	**end**
	Example:
	`Device(config-if-srv)# end`
		Returns to privileged EXEC mode.

Verifying the Trunk EFP Support Configuration

Use one or more of the commands listed below to verify the Trunk EFP Support feature configuration.

SUMMARY STEPS

1. **enable**
2. **show ethernet service instance**
3. **show ethernet service instance interface port-channel** [*number*]
4. **show bridge-domain**
5. **exit**

DETAILED STEPS

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show ethernet service instance**

Example:

```
Device# show ethernet service instance
```

Displays information about Ethernet service instances.

Step 3 **show ethernet service instance interface port-channel** [*number*]

Example:

```
Device# show ethernet service instance interface port-channel 1
```

Displays interface-only information about Ethernet service instances for all port-channel interfaces or for a specified port-channel interface.

Step 4 **show bridge-domain**

Example:

```
Device# show bridge-domain
```

Displays bridge-domain information.

Step 5 **exit**

Example:

```
Device# exit
```

Exits privileged EXEC mode.

Configuration Examples

Example: Configuring Trunk EFP Support

In the following example, EFP support has been configured on a trunk interface.

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 1
Device(config-if)# service instance trunk 1 ethernet
Device(config-if-srv)# encapsulation dot1q 1 - 5, 7, 9 - 12
Device(config-if-srv)# rewrite ingress tag pop 1 symmetric
Device(config-if-srv)# bridge-domain from-encapsulation
Device(config-if-srv)# no shutdown
Device(config-if-srv)# end
```

Example: Configure the Trunk EFP with Encapsulation from Bridge Domain

```
Device> enable
Device# configure terminal
Device(config)#interface gigabitEthernet 0/0/0
Device(config-if)#service instance trunk 4000 eth
Device(config-if-srv)#encapsulation dot1q from-bd
Device(config-if-srv)#rewrite ingress tag pop 1 symmetric
Device(config-if-srv)#bridge-domain from-encapsulation
Device(config-if-srv)#end
```

Example: Verifying the Trunk EFP Support Configuration

The following is sample output from the **show ethernet service instance** command. The output displays trunk as the service instance type and indicates that a bridge domain for VLANs in the range of 12 to 1900 (as specified by the encapsulation parameters) has been created for service instance 4000 on a trunk port (interface).

```
Device# show ethernet service instance id 4000 interface port-channel 1

Service Instance ID: 4000
Service Instance Type: Trunk
Associated Interface Port-channel: 1
Associated EVC:
L2protocol drop
CE-Vlans:
Encapsulation: dot1q 12-1900 vlan protocol type 0x8100
Rewrite: ingress tag pop 1 symmetric
Interface Port-channel Dot1q Tunnel Ethertype: 0x8100
State: Up
EFP Statistics:
   Pkts In   Bytes In   Pkts Out  Bytes Out
168729725 10798985220  160246675 10255787200
EFP Microblocks:
```

```
****************
Microblock type: Bridge-domain
Bridge-domain: 12-1900
```

Example: Verify the Trunk EFP with Encapsulation from Bridge Domain

```
Device#show ethernet service  instance id 4000 int GigabitEthernet 0/0/0 detail
Service Instance ID: 4000
Service Instance Type: Trunk
Associated Interface: GigabitEthernet0/0/0
Associated EVC:
L2protocol drop
CE-Vlans:
Encapsulation: dot1q 2-21 vlan protocol type 0x8100
Rewrite: ingress tag pop 1 symmetric
Interface Dot1q Tunnel Ethertype: 0x8100
State: Up
EFP Statistics:
   Pkts In   Bytes In   Pkts Out  Bytes Out
2810511074 191114753032          0          0
EFP Microblocks:
****************
Microblock type: Bridge-domain
Bridge-domain: 2-21

Microblock type: L2Mcast
L2 Multicast GID: 9

Microblock type: dhcp_snoop
L2 Multicast GID: 9

Microblock type: PPPoE IA UBLOCK
PPPoE IA info
Enable: 0
Format Type: 0
cricuit id:
remote id:
```

CHAPTER 3

Ethernet Virtual Connections Configuration

An Ethernet Virtual Connection (EVC) is defined by the Metro-Ethernet Forum (MEF) as an association between two or more user network interfaces that identifies a point-to-point or multipoint-to-multipoint path within the service provider network. An EVC is a conceptual *service pipe* within the service provider network. A *bridge domain* is a local broadcast domain that is VLAN-ID-agnostic. An Ethernet flow point (EFP) service instance is a logical interface that connects a bridge domain to a physical port or to an EtherChannel group.

An EVC broadcast domain is determined by a bridge domain and the EFPs that are connected to it. You can connect multiple EFPs to the same bridge domain on the same physical interface, and each EFP can have its own matching criteria and rewrite operation. An incoming frame is matched against EFP matching criteria on the interface, learned on the matching EFP, and forwarded to one or more EFPs in the bridge domain. If there are no matching EFPs, the frame is dropped.

You can use EFPs to configure VLAN translation. For example, if there are two EFPs egressing the same interface, each EFP can have a different VLAN rewrite operation, which is more flexible than the traditional switchport VLAN translation model.

Effective Cisco IOS-XE Release 3.15.0S, QoS policies on EFPs are supported with ingress rewrite type as push. In the ingress direction with one VLAN tag is pushed and in the egress direction one VLAN tag is popped.

This document describes how to configure EVC features.

For detailed information about the commands, see:

- The Cisco IOS XE 3S Carrier Ethernet Command Reference: http://www.cisco.com/en/US/docs/ios/cether/command/reference/ce_book.html
- Master Command Index for Cisco IOS XE Release 3S: http://www.cisco.com/en/US/docs/ios/mcl/allreleasemcl/all_book.html

Supported EVC Features

- Service instance—you create, delete, and modify EFP service instances on Ethernet interfaces.
- Encapsulation—you can map traffic to EFPs based on:
    - 802.1Q VLANs (a single VLAN or a list or range of VLANs)
    - 802.1Q tunneling (QinQ) VLANs (a single outer VLAN and a list or range of inner VLANs)
    - Double-tagged frames mapped to EVC based on C-tags (wildcard S-Tags)
- Bridge domains—you can configure EFPs as members of a bridge domain (up to 64 EFPs per bridge domain for bridge domain with BDIs.).
- Rewrite (VLAN translation)
    - Pop symmetric

      pop 1 removes the outermost tag

      pop 2 removes the two outermost tags

      pop symmetric adds a tag (or 2 tags for **pop 2 symmetric**) on egress for a *push* operation
    - Ingress push—The **rewrite ingress tag push dot1q** *vlan-id* **symmetric** command adds a tag to an ingress packet
    - QinQ with rewrite

  rewrite ingress tag push is supported on QoS with CoS Marking for EVCs on RSP2 module.

Note Ingress push on Qos on for EVC is *not* supported on RSP1 module.

Note EVC push is also supported on 802.1ad.

- EVC forwarding
- MAC address learning and aging
- EVCs on EtherChannels
- Hairpinning
- Split horizon
- Layer 2 protocol tunneling and QinQ
- Bridging between EFPs
- MSTP (MST on EVC bridge domain)
- EFP statistics (packets and bytes)

- QoS aware EVC/EFP per service instance
- Static MAC Addresses

These Layer 2 port-based features can run with EVC configured on the port:

- LACP
- CDP
- MSTP
- EVC egress filtering

Restrictions for Ethernet Virtual Connections Configuration

- Translate operations are *not* supported.
- You can create a maximum of 128 EFPs per bridge-domain.
- The **no mac address-table learning bridge-domain** *bridge-id* global configuration command is *not* currently supported.
- Only dot1q encapsulation is supported on trunk EFPs.

  Note Effective Cisco IOS-XE Release 3.14.0S, dot1ad encapsulation is also supported on the trunk EFPs.

- Effective Cisco IOS-XE Release 3.15.0S, ingress mapping of Differentiated Services Code Point (DSCP) or Class of Service (CoS) to the C-CoS or S-CoS is mandatory.

  Also, mapping profiles between DSCP or CoS and C-CoS or S-CoS should be the same across all ethernet flow points (EFPs).

- Egress classification and queuing is based on DSCP or CoS.
- There is a traffic drop while traversing from scaled TEFP (2000 VLANs) to EFP configurations.
- You can create a maximum of 64 EFPs per bridge-domain.
- Ingress mapping of Differentiated Services Code Point (DSCP) or Class of Service (CoS) to the C-CoS or S-CoS is supported.
- Egress classification and queuing is based on DSCP or CoS.
- Remote MEPs are not learnt when SH group 1 and SH group 2 are configured in the access EVC BD.
- TCAM exhaustion message is displayed even when TEFP with 900 VLAN has a QoS policy configured to match a single VLAN. As a result, the TEFP scale is affected.
- When EFP is configured with rewrite ingress push, user's CFI is not preserved.

Caution Ensure that you set the EVC to its default mode before reconfiguring the scale to avoid error and timing issues.

Ethernet Virtual Connections

You use the **ethernet evc** *evc-id* global configuration command to create an Ethernet virtual connection (EVC). The *evc-id* or name is a text string from 1 to 100 bytes. Entering this command puts the device into service configuration mode (config-srv) where you configure all parameters that are common to an EVC.

In this mode you can enter these commands:

- **default**—Sets a command to its defaults
- **exit**—Exits EVC configuration mode
- **no**— Negates a command or sets its defaults
- **oam**—Specifies the OAM Protocol
- **uni**—Configures a count UNI under EVC

Service Instances and EFPs

Configuring a service instance on a Layer 2 port or EtherChannel creates a pseudoport or Ethernet flow point (EFP) on which you configure EVC features. Each service instance has a unique number per interface, but you can use the same number on different interfaces because service instances on different ports are not related.

If you have defined an EVC by entering the **ethernet evc** *evc-id* global configuration command, you can associate the EVC with the service instance (optional). There is no default behavior for a service instance.

Use the **service instance** *number* **ethernet** [*name*] interface configuration command to create an EFP on a Layer 2 interface or EtherChannel and to enter service instance configuration mode. You use service instance configuration mode to configure all management and control date plane attributes and parameters that apply to the service instance on a per-interface basis.

- The **service instance** *number* is the EFP identifier, an integer from 1 to 4000.
- The optional **ethernet** *name* is the name of a previously configured EVC. You do not need to enter an EVC name, but you must enter **ethernet**. Different EFPs can share the same name when they correspond to the same EVC. EFPs are tied to a global EVC through the common name.

When you enter service instance configuration mode, you can configure these options:

- **default**—Sets a command to its defaults
- **description**—Adds a service instance specific description
- **encapsulation**—Configures Ethernet frame match criteria
- **ethernet**—Configures Ethernet-lmi parameters
- **exit**— Exits from service instance configuration mode
- **ip**—Interface Internet Protocol config commands

- **ipv6**—IPv6 interface subcommands
- **l2protocol**—Configures Layer 2 control protocol processing
- **mac**—Commands for MAC address-based features
- **no**—Negates a command or sets its defaults
- **service-policy** —Attaches a policy-map to an EFP
- **shutdown**—Takes the service instance out of service

  Enter the [**no**] **shutdown** service-instance configuration mode to shut down or bring up a service instance.
- **snmp**—Modify SNMP service instance parameters

Encapsulation

Encapsulation defines the matching criteria that maps a VLAN, a range of VLANs, class of service (CoS) bits, Ethertype, or a combination of these to a service instance. You configure encapsulation in service instance configuration mode. You must configure one encapsulation command per EFP (service instance).

Use the **encapsulation** service-instance configuration mode command to set encapsulation criteria. Different types of encapsulations are default, dot1q, dot1ad, priority-tagged and untagged. Supported Ethertypes include ipv4, ipv6, pppoe-all, pppoe-discovery, and pppoe-session.

Encapsulation classification options also include:

- outer tag VLAN
- outer tag CoS
- inner tag VLAN
- inner tag CoS
- payload ethertype

After you have entered an encapsulation method, these keyword options are available in service instance configuration mode:

- **bridge-domain**—Configures a bridge domain
- **rewrite**—Configures Ethernet rewrite criteria

Table 1: Supported Encapsulation Types

	Description
encapsulation dot1q *vlan-id* [*vlan-id*[-*vlan-id*]]	Defines the matching criteria to be used to map 802.1Q frames ingress on an interface to the appropriate EFP. The options are a single VLAN, a range of VLANs, or lists of VLANs or VLAN ranges. VLAN IDs are 1 to 4094. • Enter a single VLAN ID for an exact match of the outermost tag. • Enter a VLAN range for a ranged outermost match.
encapsulation dot1q *vlan-id* **second-dot1q** *vlan-id* [*vlan-id*[-*vlan-id*]]	Double-tagged 802.1Q encapsulation. Matching criteria to be used to map QinQ frames ingress on an interface to the appropriate EFP. The outer tag is unique and the inner tag can be a single VLAN, a range of VLANs or lists of VLANs or VLAN ranges. • Enter a single VLAN ID in each instance for an exact match of the outermost two tags. • Enter a VLAN range for **second-dot1q** for an exact outermost tag and a ranged second tag.
encapsulation dot1q {**any** | *vlan-id* [*vlan-id*[-*vlan-id*]]} **etype** *ethertype*	Ethertype encapsulation is the payload encapsulation type after VLAN encapsulation. • ethertype—The etype string can have these values: ipv4, ipv6, pppoe-discovery, pppoe-session, or pppoe-all. • Matches any or an exact outermost VLAN or VLAN range and a payload ethertype.
encapsulation dot1q *vlan_id* **cos** *cos_value* **second-dot1q** *vlan-id cos cos_value*	CoS value encapsulation defines match criterion after including the CoS for the S-Tag and the C-Tag. The CoS value is a single digit between 1 and 7 for S-Tag and C-Tag. You cannot configure CoS encapsulation with **encapsulation untagged**, but you can configure it with **encapsulation priority-tag**. The result is an exact outermost VLAN and CoS match and second tag. You can also use VLAN ranges.
encapsulation dot1q any	**encapsulation**Matches any packet with one or more VLANs.

	Description
encapsulation dot1q add **encapsulation dot1q add inner** *vlan range* **encapsulation dot1ad add** **encapsulation dot1ad add inner** *vlan range*	Adds one or more VLAN tag values for matching criteria. This command is also used with **show run** command when the encapsulation configuration command is more than the terminal width and **ethernet service multi-line** command is configured or if the encapsulation command is more than 255 characters.
encapsulation dot1q remove **encapsulation dot1ad remove**	Removes one or more VLAN tag values for matching criteria.
ethernet service multi-line	Permits use of multi-line output based on the screen width. This is applicable to **encapsulation dot1q add** command. This is visible only when **show running config** command is executed when this command is enabled. Values are *on* or *off*
encapsulation untagged	Matching criteria to be used to map untagged (native) Ethernet frames entering an interface to the appropriate EFP. Only one EFP per port can have untagged encapsulation. However, a port that hosts EFP matching untagged traffic can also host other EFPs that match tagged frames.
encapsulation default	Configures the default EFP on an interface, acting as a catch-all encapsulation. All packets are seen as native. If you enter the **rewrite** command with encapsulation default, the command is rejected. If the default EFP is the only one configured on a port, it matches all ingress frames on that port. If you configure the default EFP on a port, you cannot configure any other EFP on the same port with the same bridge domain. You can configure only one default EFP per interface. If you try to configure more than one, the command is rejected.
encapsulation priority-tagged	Specifies priority-tagged frames. A priority-tagged packet has VLAN ID 0 and CoS value of 0 to 7.

If a packet entering or leaving a port does not match any of the encapsulations on that port, the packet is dropped, resulting in *filtering* on both ingress and egress. The encapsulation must match the packet *on the wire* to determine filtering criteria. *On the wire* refers to packets ingressing the router before any rewrites and to packets egressing the router after all rewrites.

Note The router does not allow overlapping encapsulation configurations.

Ethertype

The router uses the default ether types 0x8100 and 0x88a8 for dot1q and Q-in-Q encapsulations.

The ethertypes 0x9100 and 0x9200 are supported using the custom ethertype feature by configuring the **dot1q tunneling ethertype** command on a physical port.

Custom ethertype allows configuration of the ethertype per port. The 0x9100 and 0x9200 ethertypes are supported in the custom ethertype model. 802.1q (0x8100) ethertype is the default ethertype, and is configured under each service instance.

Custom Ethertype

With the custom dot1q ethertype, you can select a non-standard (0x9100 and 0x9200) 2-byte ethertype in order to identify 802.1Q tagged frames. The router is allowed to interoperate with third party vendors' switches that do not use the standard 0x8100 ethertype to identify 802.1Q-tagged frames. For instance, if 0x9100 ethertype is used as the custom dot1q ethertype on a particular port, incoming frames containing the ethertype are assigned to the VLAN contained in the tag, immediately following the ethertype. Frames that arrive on that same port containing ethertypes other than 0x9100 and 0x8100 are forwarded to service instance with untagged encapsulation, if present.

The interface can be configured with the following ethertypes:

- 0x9100
- 0x9200

Restrictions for Custom Ethertypes

- If a custom ethertype is configured under a physical port, all tagged service instances under the physical port are forced to use that particular ethertype.
- Rewrite push is not supported on CET interfaces.
- Custom ethertype is *not* supported on IP configured/routed interfaces.
- Custom ethertype config 0x88a8 is *not* supported. Only 0x9100 and 0x9200 are supported.
- Custom Ethertype dynamic update from Dot1q to Tunneling or Tunneling to Dot1q is *not* supported.
- Outer 0x8100 packets are supported.
- Dot1q Tunneling Ethertype CFI preservation is *not* supported.
- CFM with Custom Ethertype is *not* supported.
- Mac-learning limit is *not* supported.
- 802.1ad not supported for CET.
- DHCP snooping not supported for CET.

Configuration Example

```
interface GigabitEthernet 0/1
    dot1q tunneling ethertype [0x9100 | 0x9200]
    service instance 1 ethernet
        encapsulation dot1q vlan 1  [second-dot1q vlan 2]
        rewrite ingress tag pop 1 symmetric
```

Bridge Domains

A service instance must be attached to a bridge domain. Flooding and communication behavior of a bridge domain is similar to that of a VLAN domain. Bridge-domain membership is determined by which service instances have joined it, while VLAN domain membership is determined by the VLAN tag in the packet.

Note You must configure encapsulation before you can configure the bridge domain.

Use the **bridge-domain** *bridge-id* service-instance configuration mode command to bind the EFP to a bridge domain instance. The *bridge-id* is the identifier for the bridge domain instance, an integer from 1 to 4000.

You can enable BDI MTU using the **enable_bdi_mtu sdm** template.

Setting Bandwidth

We recommend you set the bandwidth of the BDI associated with 1-Gigabit Ethernet or 10-Gigabit Ethernet interfaces. The default BDI value is 1 Gigabit for both, 1-Gigabit Ethernet and 10-Gigabit Ethernet interfaces.

Use the **bandwidth** command to set the bandwidth on the interfaces.

The following example shows the bandwidth configuration for BDI interface 120:

```
Router(config)# interface bdi 120
Router(config-if)# bandwidth 10000000
Router(config-if)# end
```

The following example displays the configured bandwidth for BDI interface 120:

```
Router# show interface bdi 120
BDI120 is up, line protocol is up
  Hardware is BDI, address is 7426.acf7.2ebf (bia 7426.acf7.2ebf)
  Internet address is 192.168.1.1/24
  MTU 1500 bytes, BW 10000000 Kbit/sec, DLY 10 usec
```

Split-Horizon

The split-horizon feature allows service instances in a bridge domain to join groups. Service instances in the same bridge domain and split-horizon group cannot forward data between each other, but can forward data between other service instances that are in the same bridge domain, but not in the same split-horizon group.

Service instances do not have to be in a split-horizon group. If a service instance does not belong to a group, it can send and receive from all ports within the bridge domain. A service instance cannot join more than one split-horizon group.

EFPs that are not configured with an explicit *group_id* do not belong to any group.

You can configure no more than 128 service instances per bridge domain. These 128 EFPs can be distributed among split-horizon groups 0 to 13.

VPLS uses a reserved internal split group 15 which does not overlap with manually configured split groups. In case VPLS is configured on same bridge domain, scale of 128 EFPs can still be achieved on the same bridge domain.

128 EFPs and 64 VFIs are supported on the same bridge domain.

Restrictions

- Maximum number of EFPs per bridge-domain is 128. The EFPs can be distributed among split-horizon groups 0 to 13.
- VPLS uses the reserved internal Split-horizon group 15 which *does not* overlap with manually configured split groups. In case VPLS is configured on the same bridge-domain, the scale of 128 EFPs is achieved on the same bridge-domain.
- 64 VFIs are supported on the same bridge-domain.

Rewrite Operations

You can use the **rewrite** command to modify packet VLAN tags. You can use this command to emulate traditional 802.1Q tagging, where packets enter a router on the native VLAN and VLAN tagging properties are added on egress. You can also use the **rewrite** command to facilitate VLAN translation and QinQ.

The supported **rewrite** commands:

- **rewrite ingress tag pop 1 symmetric**
- **rewrite ingress tag pop 2 symmetric**
- **rewrite ingress tag push dot1q** *vlan-id* **symmetric**

Enter the **rewrite ingress tag pop** {**1** | **2**} **symmetric** service-instance configuration mode command to specify the encapsulation adjustment to be performed on the frame ingress to the EFP. Entering **pop 1** pops (removes) the outermost tag; entering **pop 2** removes two outermost tags.

Note The **symmetric** keyword is required to complete **rewrite** to configuration.

When you enter the **symmetric** keyword, the egress counterpart performs the inverse action and pushes (adds) the encapsulation VLAN. You can use the **symmetric** keyword only with ingress rewrites and only when single VLANs are configured in encapsulation. If you configure a list of VLANs or a VLAN range or **encapsulation default** or **encapsulation any**, the **symmetric** keyword is not accepted for rewrite operations.

Static MAC Addresses

The router supports multicast static MAC addresses, which allow you to enable multicast at the layer 2 level. You can use multicast static MAC addresses to forward multicast packets to specific EFPs on a network.

Layer 2 Protocol Features

Layer 2 protocol peering, forwarding, and tunneling on CDP, LACP, LLDP, PAGP, STP, UDLD, and VTP traffic is supported. For more information about these features, see:

- Layer 2 Protocol Peering
- Layer 2 Protocol Software Forwarding

Layer 2 Control Protocol Enhancements

Starting with Cisco IOS XE Release 16.7.1, you can forward, tunnel, or discard Multiple Registration Protocol (MRP), Multiple VLAN Registration Protocol (MMRP) or Multiple MAC Registration Protocol (MVRP) for a service instance on an ethernet interface.

Layer 2 Control Protocol Restrictions

- The **l2protocol {discard | forward | tunnel} {mmrp | mvrp}** command should be applied on each Ethernet interface.
- The layer 2 protocol options such as discard, forward, or tunnel for MMRP or MVRP are supported on dot1q ports. The default action for these protocols on dot1q port is drop.
- The layer 2 protocol forward option is supported for MMRP or MVRP on dot1ad ports. The default action for these protocols on dot1ad port is drop.

Configuring Layer 2 Control Protocol Tunnel

To configure the Layer 2 control protocol options such as discard, forward, or tunnel on dot1q port, use the following commands:

```
interface GigabitEthernet 0/0/1
ethernet dot1ad uni s-port
service instance 2 ethernet
[no] l2protocol discard mmrp mvrp
[no] l2protocol forward mmrp mvrp
[no] l2protocol tunnel mmrp mvrp
```

The following example shows the configuration example to forward on the dot1ad port:

```
interface GigabitEthernet 0/0/2
description connected to Tester A.1
no ip address
ethernet dot1ad uni s-port
service instance 2 ethernet
encapsulation default
[no] l2protocol forward mmrp|mvrp
```

Configuring EFPs

Default EVC Configuration

No EFPs are configured. No service instances or bridge domains are configured.

Configuration Guidelines

The following guidelines apply when you configure EVCs on the router.

Note For information about supported EVC scale, see the Cisco ASR 920 Series Aggregation Services Router Configuration Guide.

- To configure a service instance on an interface, these commands are prerequisites:

```
Router (config)# interface gigabitethernet0/0/1
Router (config-if)# service instance 22 Ethernet ether
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# bridge-domain 10
```

- You must configure encapsulation on a service instance before configuring bridge domain.
- ISL trunk encapsulation is not supported.
- The router does not support overlapping configurations on the same interface and same bridge domain. If you have configured a VLAN range encapsulation, or encapsulation default, or encapsulation any on service instance 1, you cannot configure any other encapsulations that also match previous encapsulations in the same interface and bridge domain.
- QinQ is not supported on Trunk EFP interfaces.
- Trunk EFPs should be configured with the **rewrite ingress tag pop 1 symmetric** command.
- In MST instance, when you add or remove VLAN from Trunk EFP, the BDI interface goes down which results in loss of packets.
- On an access interface configured with EFP untagged and TEFP, when a tagged packet with encapsulation equal to bridge-domain of untagged EFP passes through the access interface, the packet passes through TEFP and returns to the source through EFP untagged configuration and the packet is not dropped.

Creating Service Instances

Beginning in privileged EXEC mode, follow these steps to create an EFP service instance:

SUMMARY STEPS

1. **configure terminal**
2. **interface** *interface-id*

3. **service instance** *number* **ethernet** [*name*]
4. **encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}
5. **rewrite ingress tag pop** {**1** | **2**} **symmetric**
6. **bridge-domain** *bridge-id* [**split-horizon group** *group-id*]
7. **end**
8. **show ethernet service instance show bridge-domain** [*n* | **split-horizon]**
9. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal**	Enter global configuration mode.
Step 2	**interface** *interface-id*	Specify the port to attach to the policy map, and enter interface configuration mode. Valid interfaces are physical ports.
Step 3	**service instance** *number* **ethernet** [*name*]	Configure an EFP (service instance) and enter service instance configuration) mode. • The number is the EFP identifier, an integer from 1 to 4000. • (Optional) **ethernet** name is the name of a previously configured EVC. You do not need to use an EVC name in a service instance.
Step 4	**encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}	Configure encapsulation type for the service instance. • **default**—Configure to match all unmatched packets. • **dot1q**—Configure 802.1Q encapsulation. See for details about options for this keyword. • **priority-tagged**—Specify priority-tagged frames, VLAN-ID 0 and CoS value of 0 to 7. • **untagged**—Map to untagged VLANs. Only one EFP per port can have untagged encapsulation.
Step 5	**rewrite ingress tag pop** {**1** | **2**} **symmetric**	(Optional) Specify that encapsulation modification to occur on packets at ingress. • **pop 1**—Pop (remove) the outermost tag. • **pop 2**—Pop (remove) the two outermost tags. • **symmetric—**Configure the packet to undergo the reverse of the ingress action at egress. If a tag is popped at ingress, it is pushed (added) at egress. This keyword is required for **rewrite** to function properly.

	Command or Action	Purpose
Step 6	**bridge-domain** *bridge-id* [**split-horizon group** *group-id*]	Configure the bridge domain ID. The range is from 1 to 4000. You can use the **split-horizon** keyword to configure the port as a member of a split horizon group. The *group-id* range is from 0 to 2.
Step 7	**end**	Return to privileged EXEC mode.
Step 8	**show ethernet service instance show bridge-domain** [*n* | **split-horizon]**	Verify your entries.
Step 9	**copy running-config startup-config**	(Optional) Save your entries in the configuration file. Use the **no** forms of the commands to remove the service instance, encapsulation type, or bridge domain or to disable the rewrite operation.

Creating a Trunk EFP

Beginning in privileged EXEC mode, follow these steps to create an EFP service instance:

Note Use the no forms of the commands to remove the service instance, encapsulation type, or bridge domain or to disable the rewrite operation.

Note Trunk EFPs on port-channel interfaces is supported. Traffic may *not* flow to the TEFP when the port-channel or its member links are in down state.

SUMMARY STEPS

1. **configure terminal**
2. **interface** *interface-id*
3. **service instance** [**trunk**] *number* **ethernet**
4. **encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}
5. **rewrite ingress tag pop** {**1** | **2**} **symmetric**
6. **bridge-domain** *bridge-id*
7. **end**
8. Use one of the following commands
   - **show ethernetservice instance**
   - **show bridge-domain** *[n* | **split-horizon**]
9. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal**	Enter global configuration mode.
Step 2	**interface** *interface-id*	Specify the port to attach to the policy map, and enter interface configuration mode. Valid interfaces are physical ports.
Step 3	**service instance** [**trunk**] *number* **ethernet**	Configure an EFP (service instance) and enter service instance configuration) mode. • The number is the EFP identifier, an integer from 1 to 4000. • The trunk keyword identifies the trunk ID to which the service instance is assigned. **Note** Trunk EFP (without port channel) supports encapsulation of up to 1000 VLANS.
Step 4	**encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}	**Note** Only dot1q encapsulation is supported on trunk EFPs. Configure encapsulation type for the service instance. • **default** —Configure to match all unmatched packets. • **dot1q** —Configure 802.1Q encapsulation. See Table 1 for details about options for this keyword. • **priority-tagged** —Specify priority-tagged frames, VLAN-ID 0 and CoS value of 0 to 7. • **untagged** —Map to untagged VLANs. Only one EFP per port can have untagged encapsulation.
Step 5	**rewrite ingress tag pop** {**1** | **2**} **symmetric**	(Optional) Specify that encapsulation modification to occur on packets at ingress. • **pop 1** —Pop (remove) the outermost tag. • **pop 2** —Pop (remove) the two outermost tags. **Caution** The **pop2** option is not currently supported on Trunk EFPs. • **symmetric**—Configure the packet to undergo the reverse of the ingress action at egress. If a tag is popped at ingress, it is pushed (added) at egress. This keyword is required for rewrite to function properly.
Step 6	**bridge-domain** *bridge-id*	Configures the router to derive bridge domains from the encapsulation VLAN list.
Step 7	**end**	Return to privileged EXEC mode.

	Command or Action	Purpose
Step 8	Use one of the following commands • **show ethernetservice instance** • **show bridge-domain** *[n* | **split-horizon**]	Verify your entries.
Step 9	**copy running-config startup-config**	(Optional) Save your entries in the configuration file.

Configuring Asymmetric EFPs

You can configure asymmetric rewrite rules in both ingress and egress directions of the EFP.

Encapsulation (EVC filtering) is verified at the egress for these rewrite rules:

- No rewrite rule
- Rewrite rule is **rewrite ingress tag push dot1q <value> symmetric**
- Rewrite rule is **rewrite ingress tag push dot1q <value>**
- Rewrite rule is **rewrite egress tag pop 1**

Pre-requisites

- Ensure that split-horizon groups are configured to avoid flooding between EFPs of the same Bridge Domain (BD).

Restrictions

- Transparent CFM is not supported with Asymmetric EFP.
- Q-in-Q encapsulation type in the EFP is not supported. Frames with dot1q greater than value 1 is supported. Dot1ad is not supported.
- Trunk-EFPs usage is not supported
- 2 Tag push or pop is not supported.
- Translate option, in VLAN Translation, is not supported with asymmetric rewrite rules.
- External Loopback operations are not supported.
- Ignoring MLD reports (IPv6) is not supported
- When the encapsulation is untagged in one of the EFPs, for example if **rewrite egress tag pop 1** is configured on the EFP, then **rewrite ingress tag pop 1** will cancel the rewrite rule and the packet is sent without rewrites.
- If there are different EFPs in the same BD that are carrying unicast and multicast traffic, then MAC learning should be disabled on the multicast EFP using **disable-learning** command.
- Asymmetric rewrite configuration fails for the priority-tagged encapsulation.

- When the encapsulation is untagged in one of the EFPs, for example if **rewrite egress tag pop 1** is configured on the EFP, then single tagged frames will cancel the rewrite rule and the packet is sent without rewrites.
- When two EFPs are configured at the egress under the same bridge-domain such that one of the EFPs matches the tag pushed at the egress and the other EFP does not check for encapsulation match, MAC movement can happen between the EFPs which would lead the VLAN tagging output based on the EFP on which the MAC address is learnt at the given point in time. This is an expected behavior by design. Split-horizon can be used to isolate the EFPs to avoid this behavior.

Procedure

```
enable
configure terminal
interface TenGigabitEthernet0/0/26
no ip address
service instance 1 ethernet
encapsulation dot1q 30
rewrite ingress tag pop 1
igmp ingress ignore-rewrite
bridge-domain 30
end
```

Configuration Examples

Example for Configuring a Service Instance

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 22 Ethernet ether
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# bridge-domain 10
```

Example for Encapsulation Using a VLAN Range

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 22 Ethernet
Router (config-if-srv)# encapsulation dot1q 22-44
Router (config-if-srv)# bridge-domain 10
```

Configuration Example for Larger String VLAN in Encapsulation

Configuration Example

```
  show running config

ethernet service multi-line
  !
  interface GigabitEthernet0/0/0
   service instance 1 ethernet
    encapsulation dot1q 10,13,19-21,24,29,32-36,41,46-48,55,61,63-66
    encapsulation dot1q add 69-73,78,80,83-86
   !
   service instance 2 ethernet
    encapsulation dot1q 1 second-dot1q 10,13,19-21,24,29,32-36,41
    encapsulation dot1q add outer 2-5,7
    encapsulation dot1q add inner 46-48,55,61,63-66,69-73,78,80,83-86
    encapsulation dot1q add inner 91,95-99,101
   !
   interface GigabitEthernet0/0/0
    ethernet dot1ad nni
    service instance 3 ethernet
    encapsulation dot1ad 10,13,19-21,24,29,32-36,41,46-48,55,61,63-66
    encapsulation dot1ad add 69-73,78,80,83-86
   !
   service instance 4 ethernet
    encapsulation dot1ad 1 dot1q 10,13,19-21,24,29,32-36,41,46-48,55
    encapsulation dot1ad add inner 61,63-66,69-73,78,80,83-86
   !
  !
```

Example for Two Service Instances Joining the Same Bridge Domain

In this example, service instance 1 on interfaces Gigabit Ethernet 0/0/1 and 0/0/2 can bridge between each other.

```
Router (Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# bridge-domain 10

Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# bridge-domain 10
```

Example for Bridge Domains and VLAN Encapsulation

Unlike VLANs, the bridge-domain number does not need to match the VLAN encapsulation number.

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# bridge-domain 3000

Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 20
Router (config-if-srv)# bridge-domain 3000
```

However, when encapsulations do not match in the same bridge domain, traffic cannot be forwarded. In this example, the service instances on Gigabit Ethernet 0/0/1 and 0/0/2 can not forward between each other, since the encapsulations don’t match (filtering criteria). However, you can use the **rewrite** command to allow communication between these two.

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# bridge-domain 3000

Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 99
Router (config-if-srv)# bridge-domain 3000
```

Example for Rewrite

In this example, a packet that matches the encapsulation will have one tag removed (popped off). The **symmetric** keyword allows the reverse direction to have the inverse action: a packet that egresses out this service instance will have the encapsulation (VLAN 10) added (pushed on).

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# rewrite ingress tag pop 1 symmetric
Router (config-if-srv)# bridge-domain 3000
```

Example for Split Horizon

In this example, service instances 1 and 2 cannot forward and receive packets from each other. Service instance 3 can forward traffic to any service instance in bridge domain 3000 since no other service instance in bridge domain 3000 is in split-horizon group 2. Service instance 4 can forward traffic to any service instance in bridge domain 3000 since it has not joined any split-horizon groups.

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# rewrite ingress pop 1 symmetric
Router (config-if-srv)# bridge-domain 3000 split-horizon group 1
Router (config-if-srv)# exit
Router (config-if)# service instance 2 Ethernet
Router (config-if-srv)# encapsulation dot1q 99
Router (config-if-srv)# rewrite ingress pop 1 symmetric
Router (config-if-srv)# bridge-domain 3000 split-horizon group 1

Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance 3 Ethernet
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# rewrite ingress pop 1 symmetric
Router (config-if-srv)# bridge-domain 3000 split-horizon group 2
Router (config-if-srv)# exit
Router (config-if)# service instance 4 Ethernet
Router (config-if-srv)# encapsulation dot1q 99
Router (config-if-srv)# rewrite ingress pop 1 symmetric
Router (config-if-srv)# bridge-domain 3000
```

Example for Hairpinning

The switch supports *hairpinning*, which refers to traffic ingressing and egressing same interface. To achieve haripinning, configure two EFPs in the same bridge domain on the same physical interface, as in this example.

```
Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# rewrite ingress tag pop 1 symmetric
Router (config-if-srv)# bridge-domain 4000
Router (config-if-srv)# exit
Router (config-if)# service instance 2 Ethernet
Router (config-if-srv)# encapsulation dot1q 20
Router (config-if-srv)# rewrite ingress tag pop 1 symmetric
Router (config-if-srv)# bridge-domain 4000
```

Example for Egress Filtering

In EVC switching, egress filtering is performed before the frame is sent on the egress EFP. Egress filtering ensures that when a frame is sent, it conforms to the matching criteria of the service instance applied on the ingress direction. EFP does not require egress filtering if the number of pops is the same as the number of VLANs specified in the **encapsulation** command.

Egress Filtering is not supported on the RSP3 module.

Note Specifying the **cos** keyword in the encapsulation command is relevant only in the ingress direction. For egress filtering, **cos** is ignored.

For example, consider the following configuration.

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 20
Router (config-if-srv)# bridge-domain 19

Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance 2 Ethernet
Router (config-if-srv)# encapsulation dot1q 30
Router (config-if-srv)# bridge-domain 19

Router (config)# interface gigabitethernet0/3
Router (config-if)# service instance 3 Ethernet
Router (config-if-srv)# encapsulation dot1q 10 second-dot1q 20
Router (config-if-srv)# rewrite ingress pop 1 symmetric
Router (config-if-srv)# bridge-domain 19
```

If a packet with VLAN tag 10 or 20 is received on Gigabit Ethernet 0/0/3, the ingress logical port would be service instance 3. For the frame to be forwarded on a service instance, the egress frame must match the encapsulation defined on that service instance after the rewrite is done. Service instance 1 checks for outermost VLAN 20; service instance 2 checks for VLAN 30. In this example, the frame with VLAN tags 10 and 20 can be sent to service instance 1 but not to service instance 2.

Configuring Examples for Asymmetric EFPs

Configuring Asymmetric EFP with POP

```
enable
configure terminal
interface TenGigabitEthernet0/0/26
no ip address
service instance 1 ethernet
encapsulation untagged
rewrite egress tag pop 1
bridge-domain 30
end
```

Configuring Asymmetric EFP with Single Tag Push

```
enable
configure terminal
```

```
interface TenGigabitEthernet0/0/26
no ip address
service instance 1 ethernet
encapsulation untagged
rewrite ingress tag push dot1q 10
bridge-domain 30
end
```

Configuring Asymmetric EFP with Ingress VLAN Rewrite Disabled for IGMP Control Packets"

```
enable
configure terminal
interface TenGigabitEthernet0/0/26
no ip address
service instance 1 ethernet
encapsulation dot1q 30
rewrite ingress tag pop 1
igmp ingress ignore-rewrite
bridge-domain 30
end
```

Configuring Asymmetric EFP with Disabled MAC Address Learning

```
enable
configure terminal
interface TenGigabitEthernet0/0/26
no ip address
service instance 1 ethernet
encapsulation dot1q 30
rewrite egress tag push dotq 30
disable-learning
bridge-domain 30 split-horizon group 1
end
```

Configuring Other Features on EFPs

EFPs and EtherChannels

You can configure EFP service instances on EtherChannel port channels, but EtherChannels are not supported on ports configured with service instances. Load-balancing on port channels is based on the MAC address or IP address of the traffic flow on the EtherChannel interface.

This example configures a service instance on an EtherChannel port channel. Configuration on the ports in the port channel are independent from the service instance configuration.

```
Router (config)# interface port-channel 4
Router (config-if)# service instance 1 ethernet
Router (config-if-srv)# encapsulation untagged
Router (config-if-srv)# bridge-domain {any vlan}
Router (config-if-srv)# l2protocol peer {lacp | pagp}
```

Layer 2 Protocol Peering

For Layer 2 protocols (CDP, UDLD, LLDP, MSTP, LACP,PTP peer delay, ELMI, LOAM) to peer with a neighbor on a port that has an EFP service instance configured, you need to enter the **l2 protocol peer** *protocol* service-instance configuration command on the service instance.

This example shows how to configure CDP to peer with a neighbor on a service instance:

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation untagged
Router (config-if-srv)# l2protocol peer cdp
Router (config-if-srv)# bridge-domain 10
Router (config-if-srv)# end
```

Layer 2 Protocol Software Forwarding

Layer 2 protocol forwarding is based on the bridge domain ID and the destination MAC address.

Selecting the l2protocol forward option causes the router to flood interfaces in the same VLAN or bridge-domain with untagged or tagged BPDU packets. You can apply the l2protocol forward command to CDP, LACP, LLDP, PAGP, STP, UDLD, and VTP traffic. This is an example how to configure the l2protocol forward option:

```
interface GigabitEthernet0/9
ethernet uni id PRAV-PE2
service instance 1 ethernet
encapsulation untagged
l2protocol forward cdp
bridge-domain 500
!
service instance 10 ethernet xcon
  encapsulation dot1q 100
  l2protocol forward cdp
  xconnect 4.3.2.1 12 encapsulation mpls
 !
```

Configuring IEEE 802.1Q Tunneling and Layer 2 Protocol Tunneling Using EFPs

Tunneling is a feature used by service providers whose networks carry traffic of multiple customers and who are required to maintain the VLAN and Layer 2 protocol configurations of each customer without impacting the traffic of other customers. The router uses EFPs to support QinQ and Layer 2 protocol tunneling.

802.1Q Tunneling (QinQ)

Service provider customers often have specific requirements for VLAN IDs and the number of VLANs to be supported. The VLAN ranges required by different customers in the same service-provider network might overlap, and traffic of customers through the infrastructure might be mixed. Assigning a unique range of VLAN IDs to each customer would restrict customer configurations and could easily exceed the VLAN limit (4096) of the 802.1Q specification.

Using the EVCs, service providers can encapsulate packets that enter the service-provider network with multiple customer VLAN IDs (C-VLANs) and a single 0x8100 Ethertype VLAN tag with a service provider VLAN (S-VLAN). Within the service provider network, packets are switched based on the S-VLAN. When the packets egress the service provider network onto the customer network, the S-VLAN tag is decapsulated and the original customer packet is restored.

Figure below shows the tag structures of the double-tagged packets.

In figure below, Customer A was assigned VLAN 30, and Customer B was assigned VLAN 40. Packets entering the edge switches with 802.1Q tags are double-tagged when they enter the service-provider network, with the outer tag containing VLAN ID 30 or 40, appropriately, and the inner tag containing the original VLAN number, for example, VLAN 100. Even if both Customers A and B have VLAN 100 in their networks, the traffic remains segregated within the service-provider network because the outer tag is different. Each customer controls its own VLAN numbering space, which is independent of the VLAN numbering space used by other customers and the VLAN numbering space used by the service-provider network. At the outbound port, the original VLAN numbers on the customer's network are recovered.

Method 1

In this example, for Customer A, interface Gigabit Ethernet 0/1 is the customer-facing port, and Gigabit Ethernet 0/2 is a trunk port facing the service provider network. For Customer B, Gigabit Ethernet 0/3 is the customer-facing port, and Gigabit Ethernet 0/4 is the trunk port facing the service provider network.

Customer A

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 1-100
Router (config-if-srv)# bridge-domain 4000

Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance 2 Ethernet
Router (config-if-srv)# encapsulation dot1q 30
Router (config-if-srv)# rewrite ingress tag pop 1 symmetric
Router (config-if-srv)# bridge-domain 4000
```

For Customer A, service instance 1 on Gigabit Ethernet 0/1 is configured with the VLAN encapsulations used by the customer: C-VLANs 1–100. These are forwarded on bridge-domain 4000. The service provider facing port is configured with a service instance on the same bridge-domain and with an **encapsulation dot1q** command matching the S-VLAN. The **rewrite ingress pop 1 symmetric** command also implies a push of the configured encapsulation on egress packets. Therefore, the original packets with VLAN tags between 1 and 100 are encapsulated with another S-VLAN (VLAN 30) tag when exiting Gigabit Ethernet port 0/0/2.

Similarly, for double- tagged (S-VLAN = 30, C-VLAN = 1–100) packets coming from the provider network, the **rewrite ingress pop 1 symmetric** command causes the outer S-VLAN tag to be popped and the original C-VLAN tagged frame to be forwarded over bridge-domain 4000 out to Gigabit Ethernet 0/1.

The same scenario applies to Customer B.

Customer B

```
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 1-200
Router (config-if-srv)# bridge-domain 4001

Router (config)# interface gigabitethernet0/4
Router (config-if)# service instance 2 Ethernet
Router (config-if-srv)# encapsulation dot1q 40
Router (config-if-srv)# rewrite ingress tag pop 1 symmetric
Router (config-if-srv)# bridge-domain 4001
```

Method 2

QinQ is also supported when sending packets between an EFP and a trunk EFP. The same external behavior as Method 1 can be achieved with this configuration:

Customer A

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 1-100
Router (config-if-srv)# bridge-domain 30

Router (config)# interface gigabitethernet0/0/2
Router (config-if)# service instance trunk 1 ethernet
Router (config-if-srv)# encapsulation dot1q 30
Router (config-if-srv)# rewrite ingress tag pop 1 symmetric
Router (config-if-srv)# bridge-domain from-encapsulation
```

Again, service instance 1 on Gigabit Ethernet port 0/1 is configured with the VLAN encapsulations used by the customer. These are forwarded on bridge-domain 30. The service provider facing port is configured as a trunk port. The trunk port pushes a tag matching the bridge-domain that the packet is forwarded on (in this case S-VLAN 30).

For double tagged (S-VLAN = 30, C-VLAN = 1 to 100) packets coming in from the provider network, the trunk port pops the outer S-VLAN (30) and forwards the packet on that bridge-domain.

Customer B

```
Router (config)# interface gigabitethernet0/3
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 1-200
Router (config-if-srv)# bridge-domain 40

Router (config)# interface gigabitethernet0/0/4
Router (config-if)# service instance trunk 2 Ethernet
Router (config-if-srv)# encapsulation dot1q 40
Router (config-if-srv)# rewrite ingress tag pop 1 symmetric
Router (config-if-srv)# bridge-domain from-encapsulation
```

You can also combine the customer A and B configurations, as follows:

Customer A and B

```
Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance trunk 1 ethernet
Router (config-if-srv)# encapsulation dot1q 30,40
Router (config-if-srv)# rewrite ingress tag pop 1 symmetric
Router (config-if-srv)# bridge-domain from-encapsulation
```

For information about the effect on cost of service (CoS) for different EFT tagging operations, see the Cisco ASR 920 Router Chassis Software Configuration Guide.

Example for VLAN Translation Configurations

- For 1-to-1 VLAN translation configuration:

```
Router (config)# interface gigabitethernet0/1
Router (config-if)# service instance 965 ethernet
Router (config-if-srv)# encapsulation dot1q 960
Router (config-if-srv)# rewrite ingress tag translate 1-to-1 dot1q 965 symmetric

Router (config-if-srv)#  bridge-domain 965
```

For 2-to-1 VLAN translation configuration:

```
Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance 967 ethernet
Router (config-if-srv)# encapsulation dot1q 962 second-dot1q 963

Router (config-if-srv)# rewrite ingress tag translate 2-to-1 dot1q 967 symmetric

Router (config-if-srv)# bridge-domain 967
```

Example for Ingress Mapping of C-CoS to S-CoS

S-CoS is marked with the S-CoS value when the packet is matched with the C-CoS value at ingress. In the following example, GigabitEthernet 0/1 is the ingress interface and GigabitEthernet 0/3 is the egress interface. This classification is done at the ingress interface and the S-CoS value is set at 4.

```
  policy-map policy-dscp
  class class-dscp/customer-cos
  set cos 4

interface GigabitEthernet0/1-> Input interface with EVC Push and policy on EVC
service instance 1 ethernet
 encapsulation untagged
 rewrite ingress tag push dot1q 10
 service-policy input policy-dscp
 bridge-domain 20


interface GigabitEthernet0/3
service instance 1 ethernet
 encapsulation dot1q 30
 bridge-domain 20
```

Example for Ingress Mapping of C-CoS to C-CoS

In the following example, both C-CoS and S-CoS are configured with CoS=4. The example also illustrates the ingress mapping of C-DSCP or C-CoS to C-CoS, where CoS is marked for both S-CoS and C-CoS.

```
policy-map policy-dscp
class class-dscp/customer-cos
set cos 4          -> This sets the value of S-CoS.

interface GigabitEthernet0/1    ->Input interface with EVC Push and EVC policy
service instance 1 ethernet
 encapsulation untagged
 rewrite ingress tag push dot1q 10
 service-policy input policy-dscp
 bridge-domain 20
```

```
interface GigabitEthernet0/3
service instance 1 ethernet
 encapsulation dot1q 30
 rewrite ingress tag pop1 symmetric
 bridge-domain 20
```

Example for Egress Classification Based on CoS

```
interface GigabitEthernet0/1   -> Input interface with EVC Push and EVC policy
service instance 1 ethernet
 encapsulation untagged
 rewrite ingress tag push dot1q 10
 service-policy input set-cos
 bridge-domain 20

interface GigabitEthernet0/3
service instance 1 ethernet
 encapsulation dot1q 30
 service-policy output policy-dscp
 bridge-domain 20
```

EFPs and Ethernet over Multiprotocol Layer Switching (EoMPLS)

When you configure a pseudowire under a VLAN interface (for example, VLAN 33), the pseudowire becomes a virtual Layer 2 port in that VLAN (VLAN 33), or bridge domain. In this bridge domain, you can configure other types of Layer 2 ports, such as EFP portss. Switching functionalities, such as MAC address learning, flooding, and forwarding to learned MAC addresses, apply to all the Layer 2 ports, including the pseudowire.

Note When a pseudowire is present in the same bridge domain as an EFP, you cannot configure the EFP with the **rewrite ingress tag pop 2 symmetric** service instance configuration command. Other restrictions about switching between EFPs or between EFPs also still apply.

For more information about configuring pseudowire, see the Cisco ASR 920 Router Chassis Software Configuration Guide.

Bridge Domain Routing

The switch supports IP routing and multicast routing for bridge domains, including Layer 3 and Layer 2 VPNs, using the BDI model. There are the limitations:

- You must configure BDIs for bridge-domain routing.
- The bridge domain must be in the range of 1 to 4094 to match the supported VLAN range.
- You can use bridge domain routing with only native packets.

  Bridge domain routing only works if proper tag popping is configured on the corresponding EFP BD. For example, if an EFP is configured with a single tag then **rewrite** should be **pop 1 symmetric**. If the EFP is configured with double tag then **rewrite** should be **pop 2 symmetric**. For double tag EFP, **pop 1 symmetric** and routing on the BDI is not supported.

  Note Traffic engineering is not supported for BDI Routing.

This is an example of configuring bridge-domain routing with a single tag EFP:

```
Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 10
Router (config-if-srv)# rewrite ingress tag pop 1 symmetric
Router (config-if-srv)# bridge-domain 100

Router (config)# interface bdi 100
Router (config-if)# ip address 20.1.1.1 255.255.255.255
```

This is an example of configuring bridge-domain routing with two tags:

```
Router (config)# interface gigabitethernet0/2
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# encapsulation dot1q 10 second-dot1q 20
Router (config-if-srv)# rewrite ingress tag pop 2 symmetric
Router (config-if-srv)# bridge-domain 100

Router (config)# interface bdi 100
Router (config-if)# ip address 20.1.1.1 255.255.255.255
```

EFPs and Trunk Port MAC Addresses

Because forwarding can occur between EFPs and trunk ports, MAC address movement can occur on learned addresses. Addresses learned on EFPs will have the format of interface + EFP ID, for example gigabitethernet 0/0/1 + EFP 1. When an address moves between a non-secured EFP and a trunk port, the behavior is similar to that of moving between trunk ports.

To see MAC address information for bridge domains, use the **show mac-address-table bdomain** *domain* command.

When an EFP property changes (bridge domain, rewrite, encapsulation, split-horizon, secured or unsecured, or a state change), the old dynamic MAC addresses are flushed from their existing tables. This is to prevent old invalid entries from lingering.

EFPs and MSTP

EFP bridge domains are supported by the Multiple Spanning Tree Protocol (MSTP). These restrictions apply when running STP with bridge domains.

- EVC supports only MSTP.
- All incoming VLANs (outer-most or single) mapped to a bridge domain must belong to the same MST instance or loops could occur.
- For all EFPs that are mapped to the same MST instance, you must configure backup EFPs on every redundant path to prevent loss of connectivity due to STP blocking a port.
- When STP mode is PVST+ or PVRST, EFP information is not passed to the protocol. EVC only supports only MSTP.
- Changing STP mode from MST to PVRST is not allowed.

L3 Unicast and Multicast Routing on a Bridge Domain with Multiple EFPs

L3 unicast routing and L3 multicast routing are supported on bridge domains with multiple EFPs. This feature provides the following functionality:

- Broadcast domains are determined through bridge-domains rather than VLANs
- Multiple EFPs on a single bridge domain and physical interface with L3 multicast routing enabled is supported
- Each EFP has its own match criteria and its own ingress and egress rewrite operations

Figure below shows an access-facing port with multiple EFPs configured to the route or bridge.

Example for Configuring L3 Multicast Routing on a Bridge Domain

The following example shows how to configure L3 multicast routing on a bridge domain using existing IOS commands.

```
ip routing
Ip multicast-routing
!
!
interface bdi 100
    ip address 1.1.1.1 255.255.255.0
    ip pim sparse-mode
    Igmp version v3
!
interface GigabitEthernet0/1
 service instance 1 ethernet
  encapsulation dot1q 33
rewrite ingress tag pop 1 symmetric
bridge-domain 100
!
service instance 2 ethernet
  encapsulation dot1q 55
rewrite ingress tag pop 1 symmetric
   bridge-domain 100
```

Cross-Connect on EFP Interfaces

Cross-connect provides the ability to match the encapsulation of received packets on the ingress side of an EFP interface and send them out with the same encapsulation through the egress side of the EFP interface. Cross-connect bridge-domain entries are provided, and encapsulation matching is achieved by matching bridge-domain entries for the EFPs on which cross-connect is configured.

The following types of encapsulation tags are supported:

- untagged
- rewrite tags with pop1

Restrictions

- A bridge-domain cannot be configured on an EFP if cross-connect is already configured.
- Cross-connect works only when the MPLS license is enabled.
- Priority-tagged encapsulation is not supported.

- L2VPN VC statistics are not supported on the RSP3 module.

Configuring Cross-Connect on an EFP Interface

Beginning in privileged EXEC mode, follow these steps to configure cross-Connect on an EFP Interface.

SUMMARY STEPS

1. **configure terminal**
2. **interface** *interface-id*
3. **service instance** *number* **ethernet** [*name*]
4. **encapsulation dot1q** *vlan_id* **cos** *cos_value* **second-dot1q** *vlan-id* **cos** *cos_value*
5. **xconnect** *peer-router-id vcid* **pw-class** *pw-class name*
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal**	Enter global configuration mode.
Step 2	**interface** *interface-id*	Specify an interface to configure, and enter interface configuration mode.
Step 3	**service instance** *number* **ethernet** [*name*]	Configure an EFP (service instance) and enter service instance configuration) mode. • The number is the EFP identifier, an integer from 1 to 4000. • (Optional) **ethernet** name is the name of a previously configured EVC. You do not need to use an EVC name in a service instance.
Step 4	**encapsulation dot1q** *vlan_id* **cos** *cos_value* **second-dot1q** *vlan-id* **cos** *cos_value*	CoS value encapsulation defines match criterion after including the CoS for the S-Tag and the C-Tag. The CoS value is a single digit between 1 and 7 for S-Tag and C-Tag. You cannot configure CoS encapsulation with **encapsulation** *untagged*. The result is an exact outermost VLAN and CoS match and second tag. You can also use VLAN ranges.
Step 5	**xconnect** *peer-router-id vcid* **pw-class** *pw-class name*	Bind the attachment circuit to a pseudowire virtual circuit (VC) and enter xconnect configuration mode.

Step 6	**end**	Return to privileged EXEC mode.
		This is an example configuration of cross-connect on an EFP interface:
		`interface gigabitethernet 0/0/3`
		` service instance 30 ethernet`

	Command or Action	Purpose

		`encap dot1q x second dot1q y`
		`xconnect <10.10.10.10> 123 encapsulation mpls`

MAC Address Forwarding, Learning and Aging on EFPs

- Layer 2 forwarding is based on the bridge domain ID and the destination MAC address. The frame is forwarded to an EFP if the binding between the bridge domain, destination MAC address, and EFP is known. Otherwise, the frame is flooded to all the EFPs or ports in the bridge domain.
- MAC address learning is based on bridge domain ID, source MAC addresses, and logical port number. MAC addresses are managed per bridge domain when the incoming packet is examined and matched against the EFPs configured on the interface. If there is no EFP configured, the bridge domain ID equal to the outer-most VLAN tag is used as forwarding and learning look-up key.

  If there is no matching entry in the Layer 2 forwarding table for the ingress frame, the frame is flooded to all the ports within the bridge domain. Flooding within the bridge domain occurs for unknown unicast, unknown multicast, and broadcast.
- Dynamic addresses are addresses learned from the source MAC address when the frame enters the router. All unknown source MAC addresses are sent to the CPU along with ingress logical port number and bridge domain ID for learning. Once the MAC address is learned, the subsequent frame with the destination MAC address is forwarded to the learned port. When a MAC address moves to a different port, the Layer 2 forwarding entry is updated with the corresponding port.

  Note The router does not currently support the **no mac address-table** learning bridge-domain *bridge-id* global configuration command.
- Dynamic addresses are aged out if there is no frame from the host with the MAC address. If the aged-out frame is received by the switch, it is flooded to the EFPs in the bridge domain and the Layer 2 forwarding entry is created again. The default for aging dynamic addresses is 5 minutes. However, when MST undergoes a topology change, the aging time is reduced to the *forward-delay* time configured by the spanning tree. The aging time reverts back to the last configured value when the topology change expires.

  You can configure a dynamic address aging time per bridge domain using the **mac aging-time** *time* command. The range is in seconds and valid values are 120-300. The default value is 300. An aging time of 0 means that the address aging is disabled.
- MAC address movement is detected when the host moves from one port to another. If a host moves to another port or EFP, the learning lookup for the installed entry fails because the ingress logical port number does not match and a new learning cache entry is created. The detection of MAC address movement is disabled for static MAC addresses where the forwarding behavior is configured by the user.

Configuring a Static MAC Address

This section describes how to configure a static MAC address on the router. For an overview of static MAC addresses, see Static MAC Addresses, on page 40.

Limitations

The following limitations apply when configuring static MAC addresses:

- Static MAC addresses are supported only on egress ports.
- You can configure up to 1024 multicast static MAC addresses
- You can assign up to 24 EFPs to a bridge domain configured with a multicast static MAC address.
- MAC entries configured across different bridge-domains are represented as separate entries in the router MAC table.
- Multicast static MAC addresses apply only to layer 2 traffic; layer 3 multicast traffic is not affected by a static MAC configuration and is forwarded to all EFPs in a bridge domain.

Configuring a Static MAC Address

SUMMARY STEPS

1. **configure terminal**
2. **interface** *interface-id*
3. **no ip address**
4. **no negotiation auto**
5. **service instance** *number* **ethernet** [*name*]
6. **encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}
7. **bridge-domain** *bridge-id* [**split-horizon group** *group-id*]
8. **mac static address** *address*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enter global configuration mode.
Step 2	**interface** *interface-id* **Example:** `Router(config)# interface gigabitethernet 0/0/0`	Specify the port to attach to the policy map, and enter interface configuration mode. Valid interfaces are physical ports.
Step 3	**no ip address** **Example:** `Router(config-if)# no ip address`	To set a primary or secondary IP address for an interface, use the **ip address** interface configuration command. To remove an IP address or disable IP processing, use the **no** form of this command.
Step 4	**no negotiation auto** **Example:** `Router(config-if)# no negotiation auto`	Disables autonegotiation on Gigabit Ethernet interfaces.

	Command or Action	Purpose
Step 5	**service instance** *number* **ethernet** [*name*] **Example:** `Rotuer(config-if)# service instance 1 ethernet`	Configure an EFP (service instance) and enter service instance configuration) mode. • The number is the EFP identifier, an integer from 1 to 4000. • (Optional) **ethernet** name is the name of a previously configured EVC. You do not need to use an EVC name in a service instance.

Step 6	**encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}
	Example:
	`Router(config-if-srv)# encapsulation dot1q 100`
		Configure encapsulation type for the service instance.
		• **default**—Configure to match all unmatched packets.
		• **dot1q**—Configure 802.1Q encapsulation. See for details about options for this keyword.
		• **priority-tagged**—Specify priority-tagged frames, VLAN-ID 0 and CoS value of 0 to 7.
		• **untagged**—Map to untagged VLANs. Only one EFP per port can have untagged encapsulation.

Step 7	**bridge-domain** *bridge-id* [**split-horizon group** *group-id*] **Example:** `Router(config-if-srv)# bridge-domain 100`	Configure the bridge domain ID. The range is from 1 to 4000. You can use the **split-horizon** keyword to configure the port as a member of a split horizon group. The *group-id* range is from 0 to 2.
Step 8	**mac static address** *address* **Example:** `Router(config-if-srv)# mac static address 0000.bbbb.cccc`	Specifies the multicast MAC address.

Step 9	**end**
	Example:
	`Router(config-if-srv)# end`
		Return to privileged EXEC mode.

Configuring a Multicast Static MAC Address

SUMMARY STEPS

1. **configure terminal**
2. **interface** *interface-id*
3. **service instance** *number* **ethernet** [*name*]
4. **encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}
5. **bridge-domain** *bridge-id* [**split-horizon group** *group-id*]
6. **mac static address** *address*

7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enter global configuration mode.
Step 2	**interface** *interface-id* **Example:** `Router(config)# interface gigabitethernet 0/3/6`	Specify the port to attach to the policy map, and enter interface configuration mode. Valid interfaces are physical ports.
Step 3	**service instance** *number* **ethernet** [*name*] **Example:** `Rotuer(config)# service instance 1 ethernet`	Configure an EFP (service instance) and enter service instance configuration) mode. • The number is the EFP identifier, an integer from 1 to 4000. • (Optional) **ethernet** name is the name of a previously configured EVC. You do not need to use an EVC name in a service instance.

Step 4	**encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}
	
	Example:
	
	`Router(config-if-srv)# encapsulation dot1q 1`
		Configure encapsulation type for the service instance.
		
		• **default**—Configure to match all unmatched packets.
		
		• **dot1q**—Configure 802.1Q encapsulation. See Table 1: Supported Encapsulation Types for details about options for this keyword.
		
		• **priority-tagged**—Specify priority-tagged frames, VLAN-ID 0 and CoS value of 0 to 7.
		
		• **untagged**—Map to untagged VLANs. Only one EFP per port can have untagged encapsulation.

Step 5	**bridge-domain** *bridge-id* [**split-horizon group** *group-id*] **Example:** `Router(config-if-srv)# bridge-domain 1`	Configure the bridge domain ID. The range is from 1 to 4000. You can use the **split-horizon** keyword to configure the port as a member of a split horizon group. The *group-id* range is from 0 to 2.
Step 6	**mac static address** *address* **Example:** `Router(config-if-srv)# mac static address 1302.4302.23c3`	Specifies the multicast MAC address.
Step 7	**end** **Example:**	Return to privileged EXEC mode.

	Command or Action	Purpose
	`Router(config-if-srv)# end`	

Configuration Example

This is an example configuration of a static MAC address on an EFP interface:

```
interface gigabitEthernet 0/0/3
 service instance 10 ethernet
 encapsulation dot1q 10
 bridge-domain 100
 mac static address 1302.4302.23c3
```

This configuration specifies that any layer 2 traffic sent to destination MAC address 1302.4302.23c3 is forwarded only to service instance 10 of bridge-domain interface Gigabit Ethernet 0/0/3.

To disable a static MAC configuration, apply the **mac static address** *address* command to the service instance:

```
Router (config)# interface gigabitethernet0/0/1
Router (config-if)# service instance 1 Ethernet
Router (config-if-srv)# mac static address 1302.4302.23c3
```

Monitoring EVC

Table 2: Supported show Commands

	Description
show ethernet service evc [**id** *evc-id* | **interface** *interface-id*] [**detail**]	Displays information about all EVCs, or a specific EVC when you enter an EVC ID, or all EVCs on an interface when you enter an interface ID. The **detail** option provides additional information about the EVC.
show ethernet service instance [**id** *instance-id* **interface** *interface-id* | **interface** *interface-id*] {[**detail**] | [*stats*]}	Displays information about one or more service instance (EFPs). If you specify an EFP ID and interface, only data pertaining to that particular EFP is displayed. If you specify only an interface ID, data is displayed for all EFPs on the interface.
show bridge-domain [*n*]	When you enter *n*, this command displays all the members of the specified bridge-domain, if a bridge-domain with the specified number exists. If you do not enter *n,* the command displays all the members of all bridge-domains in the system.

	Description
show bridge-domain *n* **split-horizon** [**group** {*group_id* | **all**}]	When you do not specify a **group** *group_id*, this command displays all the members of bridge-domain *n* that belong to split horizon group 0. If you specify a numerical *group_id*, this command displays all the members of the specified group id. When you enter **group all**, the command displays all members of any split horizon group.

show ethernet service instance detail	This command displays detailed service instance information, including Layer 2 protocol information. This is an example of the output:
	
	`Router# show ethernet service instance detail`
	
	`Service Instance ID: 1`
	`Associated Interface: Ethernet0/0`
	`Associated EVC:`
	`L2protocol tunnel pagp`
	`CE-Vlans:`
	
	`State: Up`
	`EFP Statistics:`
	`   Pkts In   Bytes In   Pkts Out  Bytes Out`
	`         0          0          0          0`

show mac address-table	This command displays dynamically learned or statically configured MAC security addresses.
show mac address-table bridge-domain *bridge-domain id*	This command displays MAC address table information for the specified bridge domain.
show mac address-table count bridge-domain *bridge-domain id*	This command displays the number of addresses present for the specified bridge domain.
show mac address-table learning bridge-domain *bridge-domain id*	This command displays the learning status for the specified bridge domain.

This is an example of output from the **show ethernet service instance detail** command:

```
Router# show ethernet service instance id 1 interface gigabitEthernet 0/1 detail
Service Instance ID: 1
Associated Interface: GigabitEthernet0/0/13
Associated EVC: EVC_P2P_10
L2protocol drop
CE-Vlans:
Encapsulation: dot1q 10 vlan protocol type 0x8100
Interface Dot1q Tunnel Ethertype: 0x8100
State: Up
EFP Statistics:
   Pkts In   Bytes In   Pkts Out  Bytes Out
       214      15408      97150    6994800
EFP Microblocks:
****************
```

```
Microblock type: Bridge-domain
Bridge-domain: 10
```

This is an example of output from the **show bridge-domain** command:

```
Router# show bridge-domain 100
Bridge-domain 100 (1 ports in all)
State: UP Mac learning: Enabled
Aging-Timer: 300 second(s)
Maximum address limit: 256000
GigabitEthernet0/0/0 service instance 1

Nile Mac Address Entries

BD mac addr type ports
-------------------------------------------------------------------------------------------
100 0000.bbbb.cccc STATIC Gi0/0/0.Efp1

sh mac-address-table bdomain 100

Nile Mac Address Entries

BD mac addr type ports
-------------------------------------------------------------------------------------------
100 0000.bbbb.cccc STATIC Gi0/0/0.Efp1
```

This is an example of output from the **show ethernet service instance** statistics command:

```
Router# show ethernet service instance id 1 interface gigabitEthernet 0/0/13 stats
Service Instance 1, Interface GigabitEthernet0/0/13
Pkts In   Bytes In   Pkts Out  Bytes Out
       214      15408      97150    6994800
```

This is an example of output from the **show mac-address table count** command:

```
Router# show mac address-table count bdomain 10

Mac Entries for BD   10:
---------------------------
Dynamic Address Count  : 20
Static  Address Count  : 0
Total Mac Addresses    : 20
```

CHAPTER 4

EVC Local Connect

Local connect (Layer 2 point to point service) is a point to point connection. It transparently transmits packet between two service instances which are configured on the same box. Local connect only connects two end points (service instances) without learning any Mac addresses. This is different from the traditional L2 bridging.

Note Packet is not forwarded based on MAC addresses.

Information About EVC Local Connect

Prerequisites for EVC Local Connect

- EVC Local connect and global bridge domain configurations requires Avanced Metro IPAccess license.
- Ensure EFPs are configured without any Bridge-Domain or Xconnect mapped to it.

Restrictions for EVC Local Connect

- EVC local connect is not supported on port-channel interfaces.
- CFM is *not* supported on EVC which contains local connect.
- L2 Protocol Tunnel are *not* supported.
- Ethernet Loopback is *not* supported.
- EVC local connect over Trunk is *not* supported.

- Port based local connect is *not* supported.
- Egress filtering based on encapulation, vlan translation, terminal and facility loopback are *not* supported
- Local Connect members without service instances will *not* work.
- On point-point connection storm control should not be applied. However, with local connect, broadcast storm control gets applied.
- For IP multicast, IGMP and PIM control packets get punted to CPU and then re-injected into the hardware path. The same thing applies to DHCP control packets too.

Scaling

- With the 8k SDM template enabled, there can be 4000 local connects configured. This total EFP scale can be divided among cross-connect and local- connect and there is no fixed limit on the division numbers.
- Local Connect does not share Internal Bridge-Domain space with L2VPN.
- Local Connect is scaled by half of total EFP scale. EFP scale is 8000 on RSP3-400 and 4000 on RSP3-200 modules.

How to Configure EVC Local Connect

Configuring EVC Local Connect

Before You Begin

Ensure that service instances are configured with proper encapsulations and rewrites as needed.

Procedure

Follow this procedure to establish an EVC local connection:

Configuring Service Instance 1

```
enable
configure terminal
interface GigbitEthernet0/1/6
service instance 1 ethernet
encapsulation dot1q 2
end
```

Configuring Service Instance 2

```
enable
configure terminal
interface GigabitEthernet0/1/7
service instance 2 ethernet
encapsulation dot1q 2
end
```

EVC Local Connect for Service Instances 1 and 2

```
enable
configure terminal
l2vpn xconnect context efp2
member GigabitEthernet0/1/6 service-instance 1
member GigabitEthernet0/1/7 service-instance 2
no shut
end
```

Configuring EVC Local Connect as Interworking VLAN

Interworking VLAN is configured in the **l2vpn xconnect** command for local connect, if the local connect both member EFPs has different encapsulation types as default or untagged or vlan range. Follow this procedure to configure EVC local connect using internetworking VLAN.

Before You Begin

Ensure that service instances are configured with proper encapsulations and rewrites as needed.

Procedure

```
enable
configure terminal
l2vpn xconnect context connect1
member GigbitEthernet0/3/4 service-instance 1
member GigbitEthernet0/3/7 service-instance 1
interworking vlan
end
```

Verifying EVC Local Connect Configuration

Verifying EVC Local Connect Configuration

```
show l2vpn service xconnect name efp2

Legend: St=State    XC St=State in the L2VPN Service      Prio=Priority
        UP=Up       DN=Down            AD=Admin Down      IA=Inactive
        SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware
        m=manually selected

  Interface          Group       Encapsulation                   Prio  St  XC St
  ---------                -----            ----
--  -----
VPWS name: efp2, State: UP
  Gi0/1/6                        Gi0/1/6:2(Eth VLAN)             0     UP  UP
  Gi0/1/7                        Gi0/1/7:2(Eth VLAN)             0     UP  UP
```

Verifying not Configured EVC Local Connect

```
show l2vpn service xconnect name efp2

Legend: St=State    XC St=State in the L2VPN Service      Prio=Priority
        UP=Up       DN=Down            AD=Admin Down      IA=Inactive
        SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware
        m=manually selected

  Interface          Group       Encapsulation                   Prio  St  XC St
  ---------          -----       -------------                   ----  --  -----
Xconnect entry does not exist
```

Verifying EVC Local Connect with Interworking VLAN

```
show l2vpn service name test1

Legend: St=State    XC St=State in the L2VPN Service      Prio=Priority
        UP=Up       DN=Down            AD=Admin Down      IA=Inactive
        SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware
        m=manually selected

  Interface          Group       Encapsulation                   Prio  St  XC St
  ---------          -----       -------------                   ----  --  -----
VPWS name: test1, State: UP
  Gi0/1/6                        Gi0/1/6:1(Ethernet)             0     UP  UP
  Gi0/1/7                        Gi0/1/7:10(Eth VLAN)            0     UP  UP
```

Verifying Traffic Statistics

```
show interface gig0/1/6 | in pack

  30 second input rate 43604000 bits/sec, 43955 packets/sec
  30 second output rate 0 bits/sec, 0 packets/sec
     1521946 packets input, 188721304 bytes, 0 no buffer
     0 packets output, 0 bytes, 0 underruns
```

```
show interface gig0/1/7 | in pack

  30 second input rate 0 bits/sec, 0 packets/sec
  30 second output rate 43131000 bits/sec, 43482 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     1523724 packets output, 188941776 bytes, 0 underruns
```

```
show ethernet service instance id 1 interface gig0/1/6 stats

Port maximum number of service instances: 4000
Service Instance 1, Interface GigabitEthernet0/1/6
   Pkts In   Bytes In   Pkts Out  Bytes Out
   1300224  161227776          0          0


show ethernet service instance id 2 interface gig0/1/7 stats

Port maximum number of service instances: 4000
Service Instance 2, Interface GigabitEthernet0/1/7
   Pkts In   Bytes In   Pkts Out  Bytes Out
         0          0    1300226  161228024
```

Configuration Examples

Example: Configuration Example for EVC Local Connect

Example: Configuration Example for EVC Local Connect

```
show run interface GigabitEthernet0/1/6

Building configuration...

Current configuration : 142 bytes
!
interface GigabitEthernet0/1/6
 no ip address
 negotiation auto
 no keepalive
 service instance 1 ethernet
  encapsulation dot1q 10
 !
end
```

```
show run interface GigabitEthernet0/1/7

Building configuration...

Current configuration : 142 bytes
!
interface GigabitEthernet0/1/7
 no ip address
 negotiation auto
 no keepalive
 service instance 1 ethernet
  encapsulation dot1q 10
 !
end
```

```
show run | sec localconnect1

l2vpn xconnect context localconnect1
member GigabitEthernet0/1/6 service-instance 1
member GigabitEthernet0/1/7 service-instance 1
```

Example: Configuration Example for EVC Local Connect as Interworking VLAN

Example: Configuration Example for EVC Local Connect as Interworking VLAN

```
show run interface GigabitEthernet0/3/4

Building configuration...

Current configuration : 165 bytes
!
interface GigabitEthernet0/3/4
no ip address
negotiation auto
```

```
service instance 1 ethernet
  encapsulation dot1q 1
!
end
```

```
show run interface GigabitEthernet0/3/7

Building configuration...

Current configuration : 127 bytes
!
interface GigabitEthernet0/3/7
no ip address
negotiation auto
service instance 1 ethernet
  encapsulation default
!
end
```

```
show run | sec localconnect2

l2vpn xconnect context localconnect2
interworking vlan
member GigabitEthernet0/3/4 service-instance 1
member GigabitEthernet0/3/7 service-instance 1
```

Use Cases or Deployment Scenarios

Ingress is VLAN list and Egress is fixed VLAN

If you have the configuration where, Ingress has encapsulations as a list of VLAN and Egress is a fixed VLAN. You need to configure interworking VLAN in **l2vpn xconnect** command to enable local connect and the state of the connection to be UP.

A notification is displayed when you have not configured interworking VLAN.

The following configuration describes the scenario:

```
enable
configure terminal
interface GigabitEthernet0/1/6
service instance 1 ethernet
encapsulation dot1q 2,4,5-8,10
end

enable
configure terminal
interface GigabitEthernet0/1/7
service instance 2 ethernet
encapsulation dot1q 5
end

enable
configure terminal
l2vpn xconnect context efp2
member gigabitEthernet 0/1/6 service-instance 1
member gigabitEthernet 0/1/7 service-instance 2
no shut
end

% Incomplete xconnect configuration. Please configure interworking.
```

You can verify the configuration using the **show l2vpn service xconnect name name**

```
show l2vpn service xconnect name efp2

Legend: St=State    XC St=State in the L2VPN Service      Prio=Priority
        UP=Up       DN=Down            AD=Admin Down      IA=Inactive
        SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware
        m=manually selected

  Interface          Group       Encapsulation                   Prio  St  XC St
  ---------                   -----       -------------                   ----  --  -----
VPWS name: efp2, State: UP
  Gi0/1/6                       Gi0/1/6:1(Ethernet)              0     UP  UP
  Gi0/1/7                       Gi0/1/7:5(Eth VLAN)              0     UP  UP
```

CHAPTER 5

Using Ethernet Fault Detection

Different interfaces on the routers nowadays support layer 1 failure detection where the signal loss between two peers is detected and the information is communicated to the control plane to allow a corrective action. POS interfaces detect complex failures in the routers at layer 2 level and use higher level protocols such as Point-to-Point Protocol (PPP) to negotiate the state between peers before the interface is brought up at an layer 2 level in the control plane.

With the proliferation of the pure layer 2 network, ethernet must also detect failures and perform corrective actions. Hence, the Ethernet Fault Detection (EFD) mechanism is used that allows Ethernet OAM protocols, such as CFM, to control the "line protocol" state of an interface so that if a CFM defect is detected, the corrective actions can be performed.

Previously, EFD was used as an event notifier for CFM only. Protocols like G8032 and REP could register to CFM events and could update the protocol (could bring it down or up) accordingly. But now, EFD allows CFM events and the OAM protocols to be used as an layer 2 BFD or line-protocol for Ethernet interfaces.

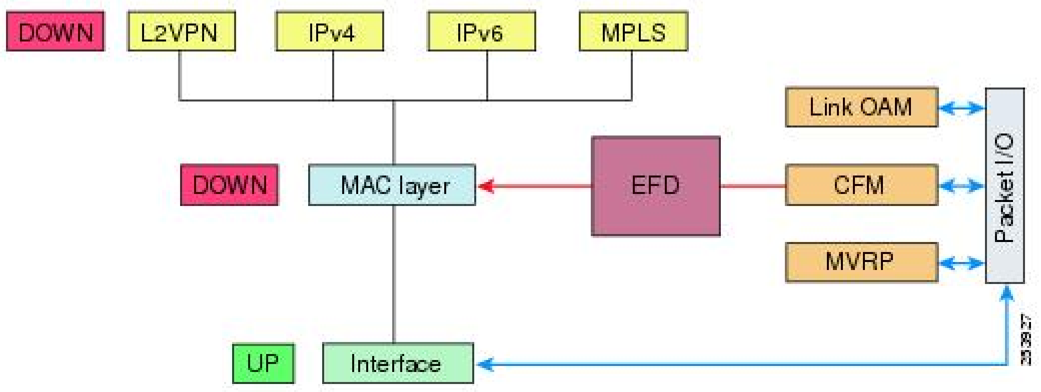
Information about Ethernet Fault Detection

Ethernet Fault Detection (EFD) is a mechanism that allows Ethernet OAM protocols, such as CFM, to control the “line protocol” state of an interface.Unlike many other interface types, Ethernet interfaces do not have a line protocol, whose state is independent from that of the interface. For Ethernet interfaces, this role is handled by the physical-layer Ethernet protocol itself, and therefore if the interface is physically up, then it is available and traffic can flow.

EFD changes this to allow CFM to act as the line protocol for Ethernet interfaces. This allows CFM to control the interface state so that if a CFM defect (such as AIS or loss of continuity) is detected with an expected peer MEP, the interface can be shut down. This not only stops any traffic flowing, but also triggers actions in any higher-level protocols to route around the problem. For example, in the case of Layer 2 interfaces, the MAC table would be cleared and MSTP would reconverge. For Layer 3 interfaces, the ARP cache would be cleared and potentially the IGP would reconverge.

Note EFD can only be used for down MEPs. When EFD is used to shut down the interface, the CFM frames continue to flow. This allows CFM to detect when the problem has been resolved, and thus bring the interface backup automatically.

Figure 1: CFM Error Detection and EFD Trigger



The figure shows CFM detection of an error on one of its sessions EFD signaling an error to the corresponding MAC layer for the interface. This triggers the MAC to go to a down state, which further triggers all higher level protocols (Layer 2 pseudowires, IP protocols, and so on) to go down and also trigger a reconvergence where possible. As soon as CFM detects there is no longer any error, it can signal to EFD and all protocols will once again go active.

Prerequisites for EFD

- EFD should be configured on only one side of the connection. When both sides go down because of EFD, CFM cannot be brought up as CFM frames are not sent by both the nodes.
- EFD is supported on EFP, port-channel, and port MEPs.

Limitations and Restrictions for EFD

- EFD is supported only on Down MEPs.
- When a EFD line-protocol enabled CFM service has down MEPs configured under different interfaces (EFP), EFD events such as CCM interval mismatch, MD level mismatch bring down all the interfaces containing MEPs created for this CFM service.
- EFD action is not applicable to Trunk EFPs.
- EFD actions are not successful with a forwarding-loop.

Enabling Ethernet Fault Detection for a Service

To enable Ethernet Fault Detection (EFD) for a service to achieve fast convergence, complete the following steps:

Note Link protection is not supported on the Cisco ASR 900 RSP3 Module.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ethernet cfm global**
4. **link-protection enable**
5. **ethernet cfm domain***domain-name* **level** *level-id* [**direction outward**]
6. **service** {*ma-name* | *ma-num* | **vlan-id** *vlan-id* | **vpn-id** *vpn-id*} [**port** | **vlan** *vlan-id* [**direction down**]]
7. **continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**]
8. **efd notify g8032**
9. **efd line-protocol**
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ethernet cfm global** **Example:** `Device(config)# ethernet cfm global`	Enables Ethernet CFM globally.
Step 4	**link-protection enable** **Example:** `Device(config)# link-protection enable`	Enables link protection globally on the router.
Step 5	**ethernet cfm domain***domain-name* **level** *level-id* [**direction outward**] **Example:** `Device(config)# ethernet cfm domain G8032 level 4`	Configures the CFM domain for ODU 1 and enters Ethernet CFM configuration mode.

	Command or Action	Purpose
Step 6	**service** {*ma-name* | *ma-num* | **vlan-id** *vlan-id* | **vpn-id** *vpn-id*} [**port** | **vlan** *vlan-id* [**direction down**]] **Example:** `Device(config-ecfm)# service 8032_service evc 8032-evc vlan 1001 direction down`	Defines a maintenance association for ODU 1 and enters Ethernet CFM service instance configuration mode.
Step 7	**continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**] **Example:** `Device(config-ecfm-srv)# continuity-check interval 3.3ms`	Enables the transmission of continuity check messages (CCMs).
Step 8	**efd notify g8032** **Example:** `Device(config-ecfm-srv)# efd notify g8032`	Enables CFM to notify registered protocols when a defect is detected or cleared, which matches the current fault alarm priority.
Step 9	**efd line-protocol** **Example:** `Device(config-ecfm-srv)# efd line-protocol`	Triggers line-protocol action when the CFM error-database is updated.

Step 10	**end**
	Example:
	`Device(config-ecfm-srv)# end`
		Returns to user EXEC mode.

Configuration Example for EFD

The following example shows a sample output of a service instance that is brought down by an EFD action.

```
Device#show ethernet ser ins int g0/0/6
Identifier Type Interface State CE-Vlans
2 Static GigabitEthernet0/0/6 ErrorDis
```

The following example is a sample of configuration of EFD.

```
Device#show ethernet ser ins int g0/0/6
Identifier Type Interface State CE-Vlans
2 Static GigabitEthernet0/0/6 ErrorDis
```

CHAPTER 6

Configuring Ethernet Connectivity Fault Management in a Service Provider Network

Ethernet Connectivity Fault Management (CFM) is an end-to-end per-service-instance Ethernet layer operations, administration, and maintenance (OAM) protocol. It includes proactive connectivity monitoring, fault verification, and fault isolation for large Ethernet metropolitan-area networks (MANs) and WANs.

The advent of Ethernet as a MAN and WAN technology imposes a new set of OAM requirements on Ethernet's traditional operations, which were centered on enterprise networks only. The expansion of Ethernet technology into the domain of service providers, where networks are substantially larger and more complex than enterprise networks and the user base is wider, makes operational management of link uptime crucial. More importantly, the timeliness in isolating and responding to a failure becomes mandatory for normal day-to-day operations, and OAM translates directly to the competitiveness of the service provider.

Note As an alternative, CFM can be configured over an Ethernet flow point (EFP) interface by using the cross connect functionality. For more information about this alternative, see *Configuring the CFM over EFP Interface with Cross Connect Feature* .

Prerequisites for Configuring Ethernet CFM in a Service Provider Network

Business Requirements

- Network topology and network administration have been evaluated.

- Business and service policies have been established.

Restrictions for Configuring Ethernet CFM in a Service Provider Network

- CFM loopback messages will not be confined within a maintenance domain according to their maintenance level. The impact of not having CFM loopback messages confined to their maintenance levels occurs at these levels:
    - Architecture—CFM layering is violated for loopback messages.
    - Deployment—A user may potentially misconfigure a network and have loopback messages succeed.
    - Security—A malicious device that recognizes devices' MAC addresses and levels may potentially explore a network topology that should be transparent.
- CFM is not fully supported on a Multiprotocol Label Switching (MPLS) provider edge (PE) device. There is no interaction between CFM and an Ethernet over MPLS (EoMPLS) pseudowire.
- CFM configuration is not supported on an EtherChannel in FastEthernet Channel (FEC) mode.
- QinQ encapsulation is not supported on the Cisco ASR 1000 Series Aggregation Services Router for CFM for routed subinterfaces.
- The client hog message is seen with scaled hardware offloaded CFM sessions for both local and remote MEP statistics.

CFM Configuration over EFP Interface with Cross Connect Feature

Ethernet Connectivity Fault Management (CFM) is an end-to-end per-service-instance Ethernet layer OAM protocol that includes proactive connectivity monitoring, fault verification, and fault isolation. Currently, Ethernet CFM supports Up facing and Down facing Maintenance Endpoints (MEPs).

For information on Ethernet Connectivity Fault Management, see http://www.cisco.com/en/US/docs/ios/12_2sr/12_2sra/feature/guide/srethcfm.html.

The CFM over EFP Interface with xconnect feature allows you to:

- Forward continuity check messages (CCM) towards the core over cross connect pseudowires.
- Receive CFM messages from the core.
- Forward CFM messages to the access side (after Continuity Check Database [CCDB] based on maintenance point [MP] filtering rules).

Restrictions for CFM Configuration over EFP Interface with Cross Connect Feature

Route Switch Processor 1 (RSP1)

- Only a single down-facing MEP is allowed on the L2VFI.
- As the number of PEs in a VPLS instance scale up, the number of CFM CC messages processed increases. Accordingly, the configuration of the down-facing MEP on L2VFI for large fully meshed PW topologies should be considered for only premium valued networks.
- In the design of CFM domains, the maintenance level of a Down-facing MEP on the L2VFI interface must be lower than the level from the AC.
- Up MEP, Down MEP, and MIPs are supported.
- CFM over untagged EFP is *not* supported on RSP1 module.

RSP1 and RSP2 Module

- Configuration of CCM sampling rate for the offloaded sessions using **offload sampling** command is not supported.
- Parsing multiple organizational-specific Type Length Value (TLV) is not supported.
- Priority-tagged encapsulation type is not supported.
- Error-objects are seen on active and standby RSP after reboot when CFM is globally disabled and MIP filter is enabled.
- CFM Traceroute with (forwarding database) FDB only option is not supported on Up MEP.
- CFM CC/Ping/Traceroute for Down MEP, CFM Ping/Traceroute for Up MEP use the bypass EAID, so these packets cannot be mirrored in the egress direction. Only Up MEP CFM CC can be mirrored.
- CFM Traceroute to expired RMEPs are flooded only to port where it was last learned. CFM Traceroute for new RMEPs are not initiated on their own. However ping to both expired and new RMEPs are flooded to all EFPs in the BD.

RSP3 Module

- L2VPN VC statistics are not supported on the RSP3 module.

Information About Configuring Ethernet CFM in a Service Provider Network

Ethernet CFM

Ethernet CFM is an end-to-end per-service-instance Ethernet layer OAM protocol that includes proactive connectivity monitoring, fault verification, and fault isolation. End to end can be PE to PE or CE to CE. A service can be identified as a service provider VLAN (S-VLAN) or an EVC service.

Being an end-to-end technology is the distinction between CFM and other metro-Ethernet OAM protocols. For example, MPLS, ATM, and SONET OAM help in debugging Ethernet wires but are not always end-to-end. 802.3ah OAM is a single-hop and per-physical-wire protocol. It is not end to end or service aware.

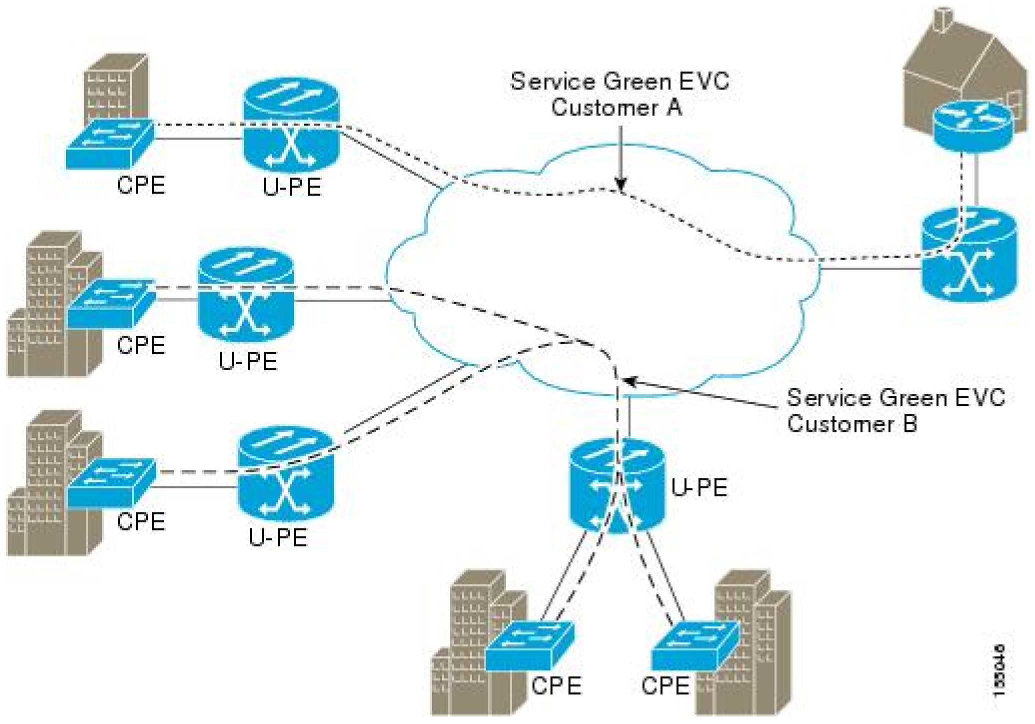
Troubleshooting carrier networks offering Ethernet Layer 2 services is challenging. Customers contract with service providers for end-to-end Ethernet service and service providers may subcontract with operators to provide equipment and networks. Compared to enterprise networks, where Ethernet traditionally has been implemented, these constituent networks belong to distinct organizations or departments, are substantially larger and more complex, and have a wider user base. Ethernet CFM provides a competitive advantage to service providers for which the operational management of link uptime and timeliness in isolating and responding to failures is crucial to daily operations.

Benefits of Ethernet CFM

- End-to-end service-level OAM technology
- Reduced operating expense for service provider Ethernet networks
- Competitive advantage for service providers
- Supports both distribution and access network environments with the outward facing MEPs enhancement

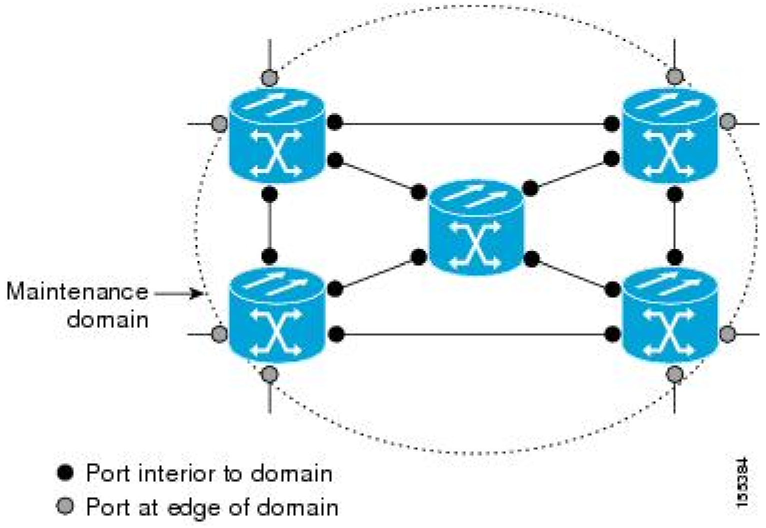
Customer Service Instance

A customer service instance is an Ethernet virtual connection (EVC), which is identified by an S-VLAN within an Ethernet island, and is identified by a globally unique service ID. A customer service instance can be point-to-point or multipoint-to-multipoint. The figure below shows two customer service instances. Service Instance Green is point to point; Service Instance Blue is multipoint to multipoint.



Maintenance Domain

A maintenance domain is a management space for the purpose of managing and administering a network. A domain is owned and operated by a single entity and defined by the set of ports internal to it and at its boundary. The figure below illustrates a typical maintenance domain.

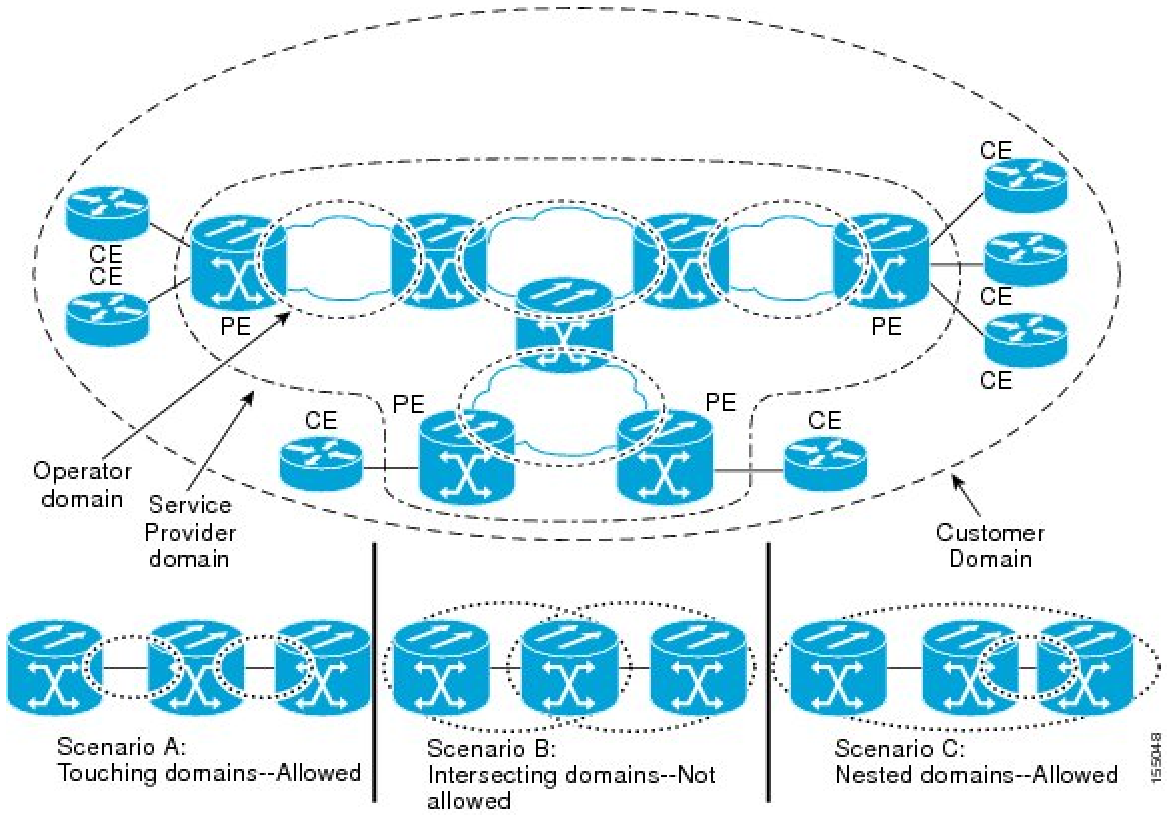


A unique maintenance level in the range of 0 to 7 is assigned to each domain by a network administrator. Levels and domain names are useful for defining the hierarchical relationship that exists among domains. The hierarchical relationship of domains parallels the structure of customer, service provider, and operator. The larger the domain, the higher the level value. For example, a customer domain would be larger than an operator domain. The customer domain may have a maintenance level of 7 and the operator domain may have a maintenance level of 0. Typically, operators would have the smallest domains and customers the largest domains, with service provider domains between them in size. All levels of the hierarchy must operate together.

Domains should not intersect because intersecting would mean management by more than one entity, which is not allowed. Domains may nest or touch but when two domains nest, the outer domain must have a higher maintenance level than the domain nested within it. Nesting maintenance domains is useful in the business model where a service provider contracts with one or more operators to provide Ethernet service to a customer. Each operator would have its own maintenance domain and the service provider would define its domain—a superset of the operator domains. Furthermore, the customer has its own end-to-end domain which is in turn a superset of the service provider domain. Maintenance levels of various nesting domains should be communicated among the administering organizations. For example, one approach would be to have the service provider assign maintenance levels to operators.

CFM exchanges messages and performs operations on a per-domain basis. For example, running CFM at the operator level does not allow discovery of the network by the higher provider and customer levels.

Network designers decide on domains and configurations. The figure below illustrates a hierarchy of operator, service provider, and customer domains and also illustrates touching, intersecting, and nested domains.


CE
CE
CE
CE
CE
CE
PE
PE
PE
PE
CE
CE
Operator domain
Service Provider domain
Customer Domain
Scenario A: Touching domains--Allowed
Scenario B: Intersecting domains--Not allowed
Scenario C: Nested domains--Allowed
155048

Maintenance Associations and Maintenance Points

A maintenance association (MA) identifies a service that can be uniquely identified within the maintenance domain. The CFM protocol runs within a maintenance association. A maintenance point is a demarcation point on an interface that participates in CFM within a maintenance domain. Maintenance points drop all lower-level frames and forward all higher-level frames. There are two types of maintenance points:

- Maintenance end points (MEPs) are points at the edge of the domain that define the boundaries and confine CFM messages within these boundaries. Outward facing or Down MEPs communicate through the wire side (connected to the port). Inward facing or Up MEPs communicate through the relay function side, not the wire side.

  CFM 802.1ag supports up and down per-VLAN MEPs, as well as port MEPs, which are untagged down MEPs that are not associated with a VLAN.

  Port MEPs are configured to protect a single hop and used to monitor link state through CFM. If a port MEP is not receiving continuity check messages from its peer (static remote MEP), for a specified interval, the port is put into an operational down state in which only CFM and OAM packets pass through, and all other data and control packets are dropped.

  - **Up MEP**—An up MEP sends and receives CFM frames through the relay function. It drops all CFM frames at its level or lower that come from the wire side, except traffic going to the down MEP. For CFM frames from the relay side, it processes the frames at its level and drops frames at a lower level. The MEP transparently forwards all CFM frames at a higher level, regardless of whether they are received from the relay or wire side. If the port on which MEP is configured is blocked by STP, the MEP can still send or receive CFM messages through the relay function. CFM runs at the provider maintenance level (UPE-to-UPE), specifically with up MEPs at the user network interface (UNI).

Note The device rate-limits all incoming CFM messages at a fixed rate of 500 frames per second.

- **Dowm MEP**—A down MEP sends and receives CFM frames through the wire connected to the port on which the MEP is configured. It drops all CFM frames at its level or lower that come from the relay side. For CFM frames from the wire side, it processes all CFM frames at its level and drops CFM frames at lower levels except traffic going to the other lower-level down MEP. The MEP transparently forwards all CFM frames at a higher level, regardless of whether they are received from the relay or through the wire.

- Maintenance intermediate points (MIPs) are internal to a domain, not at the boundary, and respond to CFM only when triggered by traceroute and loopback messages. They forward CFM frames received from MEPs and other MIPs, drop all CFM frames at a lower level (if MIP filtering is enabled), and forward all CFM frames at a higher level and at a lower level and regardless of whether they are received from the relay or wire side. When MIP filtering is enabled, the MIP drops CFM frames at a lower level. MIPs also catalog and forward continuity check messages (CCMs), but do not respond to them.

  MIP filtering is disabled by default, and you can configure it to be enabled or disabled. When MIP filtering is disabled, all CFM frames are forwarded.

  You can manually configure a MIP or configure the device to automatically create a MIP. You can configure a MEP without a MIP. In case of a configuration conflict, manually created MIPs take precedence over automatically created MIPs.

  If port on which the MEP is configured is blocked by Spanning-Tree Protocol (STP), the MIP can receive and might respond to CFM messages from both the wire and relay side, but cannot forward any CFM messages.

Maintenance Point

A maintenance point is a demarcation point on an interface (port) that participates in CFM within a maintenance domain. Maintenance points on device ports act as filters that confine CFM frames within the bounds of a domain by dropping frames that do not belong to the correct level. Maintenance points must be explicitly configured on Cisco devices. Two classes of maintenance points exist, MEPs and MIPs.

Maintenance Endpoints

Maintenance endpoints (MEPs) have the following characteristics:

- Per maintenance domain (level) and service (S-VLAN or EVC)
- At the edge of a domain, define the boundary
- Within the bounds of a maintenance domain, confine CFM messages
- When configured to do so, proactively transmit Connectivity Fault Management (CFM) continuity check messages (CCMs)
- At the request of an administrator, transmit traceroute and loopback messages

Inward Facing MEPs

Inward facing means the MEP communicates through the Bridge Relay function and uses the Bridge-Brain MAC address. An inward facing MEP performs the following functions:

- Sends and receives CFM frames at its level through the relay function, not via the wire connected to the port on which the MEP is configured.
- Drops all CFM frames at its level (or lower level) that come from the direction of the wire.
- Processes all CFM frames at its level coming from the direction of the relay function.
- Drops all CFM frames at a lower level coming from the direction of the relay function.
- Transparently forwards all CFM frames at its level or a higher level, independent of whether they come in from the relay function side or the wire side.

Note A MEP of level L (where L is less than 7) requires a MIP of level M > L on the same port; hence, CFM frames at a level higher than the level of the MEP will be catalogued by this MIP.

- If the port on which the inward MEP is configured is blocked by Spanning-Tree Protocol, the MEP can no longer transmit or receive CFM messages.

Outward Facing MEPs for Port Channels

Outward facing means that the MEP communicates through the wire. Outward facing MEPs can be configured on port channels (using cross connect functionality). A MIP configuration at a level higher than the level of the outward facing MEP is not required.

Outward facing MEPs on port channels use the Bridge-Brain MAC address of the first member link. When port channel members change, the identities of outward facing MEPs do not have to change.

An outward facing MEP performs the following functions:

- Sends and receives CFM frames at its level via the wire connected to the port where the MEP is configured.
- Drops all CFM frames at its level (or at a lower level) that come from the direction of the relay function.
- Processes all CFM frames at its level coming from the direction of the wire.
- Drops all CFM frames at a lower level coming from the direction of the wire.
- Transparently forwards all CFM frames at levels higher than the level of the outward facing MEP, independent of whether they come in from the relay function side or the wire side.
- If the port on which the outward MEP is configured is blocked by the Spanning-Tree Protocol, the MEP can still transmit and receive CFM messages via the wire.

Maintenance Intermediate Points

MIPs have the following characteristics:

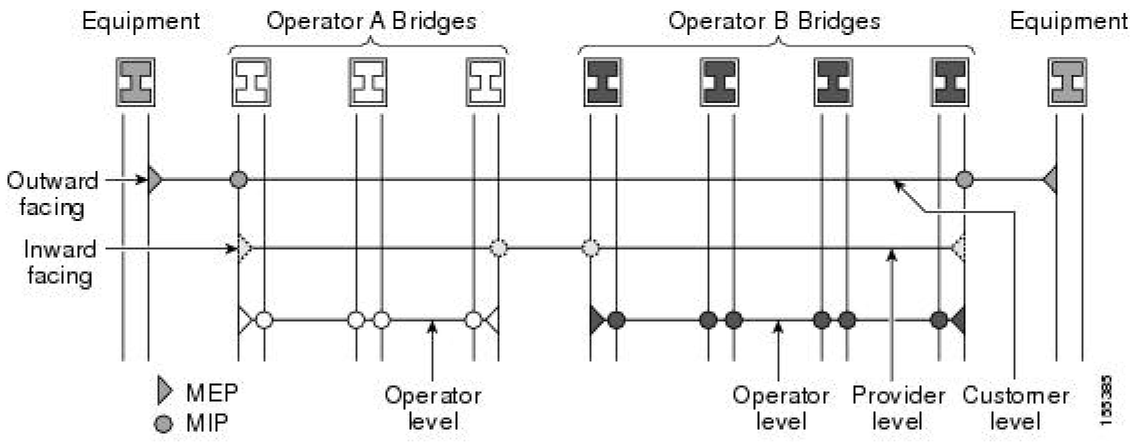
- Per maintenance domain (level) and for all S-VLANs enabled or allowed on a port.
- Internal to a domain, not at the boundary.

- CFM frames received from MEPs and other MIPs are cataloged and forwarded, using both the wire and the relay function.
- All CFM frames at a lower level are stopped and dropped, independent of whether they originate from the wire or relay function.
- All CFM frames at a higher level are forwarded, independent of whether they arrive from the wire or relay function.
- MIPs respond only when triggered by CFM traceroute and loopback messages.
- Bridge-Brain MAC addresses are used.

If the port on which a MIP is configured is blocked by Spanning-Tree Protocol, the MIP cannot receive CFM messages or relay them toward the relay function side. The MIP can, however, receive and respond to CFM messages from the wire.

A MIP has only one level associated with it and the command-line interface (CLI) does not allow you to configure a MIP for a domain that does not exist.

The figure below illustrates MEPs and MIPs at the operator, service provider, and customer levels.



CFM Messages

CFM uses standard Ethernet frames. CFM frames are distinguishable by EtherType and for multicast messages by MAC address. CFM frames are sourced, terminated, processed, and relayed by bridges. Routers can support only limited CFM functions.

Bridges that cannot interpret CFM messages forward them as normal data frames. All CFM messages are confined to a maintenance domain and to an S-VLAN (PE-VLAN or Provider-VLAN). Three types of messages are supported:

- Continuity Check
- Loopback
- Traceroute

Continuity Check Messages

CFM CCMs are multicast heartbeat messages exchanged periodically among MEPs. They allow MEPs to discover other MEPs within a domain and allow MIPs to discover MEPs. CCMs are confined to a domain and S-VLAN.

CFM CCMs have the following characteristics:

- Transmitted at a configurable periodic interval by MEPs. The interval can be from 10 seconds to 65535 seconds, the default is 30.
- Contain a configurable hold-time value to indicate to the receiver the validity of the message. The default is 2.5 times the transmit interval.
- Catalogued by MIPs at the same maintenance level.
- Terminated by remote MEPs at the same maintenance level.
- Unidirectional and do not solicit a response.
- Carry the status of the port on which the MEP is configured.

Loopback Messages

CFM loopback messages are unicast frames that a MEP transmits, at the request of an administrator, to verify connectivity to a particular maintenance point. A reply to a loopback message indicates whether a destination is reachable but does not allow hop-by-hop discovery of the path. A loopback message is similar in concept to an Internet Control Message Protocol (ICMP) Echo (ping) message.

A CFM loopback message can be generated on demand using the CLI. The source of a loopback message must be a MEP; the destination may be a MEP or a MIP. CFM loopback messages are unicast; replies to loopback messages also are unicast. CFM loopback messages specify the destination MAC address, VLAN, and maintenance domain.

Traceroute Messages

CFM traceroute messages are multicast frames that a MEP transmits, at the request of an administrator, to track the path (hop-by-hop) to a destination MEP. They allow the transmitting node to discover vital connectivity data about the path, and allow the discovery of all MIPs along the path that belong to the same maintenance domain. For each visible MIP, traceroute messages indicate ingress action, relay action, and egress action. Traceroute messages are similar in concept to User Datagram Protocol (UDP) traceroute messages.

Traceroute messages include the destination MAC address, VLAN, and maintenance domain and they have Time To Live (TTL) to limit propagation within the network. They can be generated on demand using the CLI. Traceroute messages are multicast; reply messages are unicast.

Cross-Check Function

The cross-check function is a timer-driven post-provisioning service verification between dynamically discovered MEPs (via CCMs) and expected MEPs (via configuration) for a service. The cross-check function verifies that all endpoints of a multipoint or point-to-point service are operational. The function supports notifications when the service is operational; otherwise it provides alarms and notifications for unexpected endpoints or missing endpoints.

The cross-check function is performed one time. You must initiate the cross-check function from the CLI every time you want a service verification.

Ethernet CFM and Ethernet OAM Interaction

To understand how CFM and OAM interact, you should understand the following concepts:

Ethernet Virtual Circuit

An EVC as defined by the Metro Ethernet Forum is a port-level point-to-point or multipoint-to-multipoint Layer 2 circuit. EVC status can be used by a CE device either to find an alternative path in to the service provider network or in some cases, to fall back to a backup path over Ethernet or over another alternative service such as ATM.

OAM Manager

The OAM manager is an infrastructure element that streamlines interaction between OAM protocols. The OAM manager requires two interworking OAM protocols, in this case Ethernet CFM and Ethernet OAM. Interaction is unidirectional from the OAM manager to the CFM protocol and the only information exchanged is the user network interface (UNI) port status. Additional port status values available include

- REMOTE_EE—Remote excessive errors
- LOCAL_EE—Local excessive errors
- TEST—Either remote or local loopback

After CFM receives the port status, it communicates that status across the CFM domain.

CFM over Bridge Domains

Connectivity Fault Management (CFM) over bridge domains allows untagged CFM packets to be associated with a maintenance end point (MEP). An incoming untagged customer CFM packet has an EtherType of CFM and is mapped to an Ethernet virtual circuit (EVC) or bridge domain based on the encapsulation configured on the Ethernet flow point (EFP). The EFP is configured specifically to recognize these untagged packets.

An EFP is a logical demarcation point of an EVC on an interface and can be associated with a bridge domain. The VLAN ID is used to match and map traffic to the EFP. VLAN IDs have local significance per port similar to an ATM virtual circuit. CFM is supported on a bridge domain associated with an EFP. The association between the bridge domain and the EFP allows CFM to use the encapsulation on the EFP. All EFPs in the same bridge domain form a broadcast domain. The bridge domain ID determines the broadcast domain.

The distinction between a VLAN port and the EFP is the encapsulation. VLAN ports use a default dot1q encapsulation. For EFPs, untagged, single tagged, and double tagged encapsulation exists with dot1q and IEEE dot1ad EtherTypes. Different EFPs belonging to the same bridge domain can use different encapsulations.

Both up MEP, down MEP and MIP are supported. If an up MEP is configured under an EFP within a bridge domain, CFM messages would be routed into the bridge, and the rest members of the same bridge domain would be able to receive messages from this MEP. If a down MEP is configured, the messages will not goes into the bridge domain.

How to Set Up Ethernet CFM in a Service Provider Network

Designing CFM Domains

Note To have an operator, service provider, or customer domain is optional. A network may have a single domain or multiple domains. The steps listed here show the sequence when all three types of domains will be assigned.

Before you begin

- Knowledge and understanding of the network topology.
- Understanding of organizational entities involved in managing the network; for example, operators, service providers, network operations centers (NOCs), and customer service centers.
- Understanding of the type and scale of services to be offered.
- Agreement by all organizational entities on the responsibilities, roles, and restrictions for each organizational entity.
- Determination of the number of maintenance domains in the network.
- Determination of the nesting and disjoint maintenance domains.
- Assignment of maintenance levels and names to domains based on agreement between the service provider and operator or operators.
- Determination of whether the domain should be inward or outward.

SUMMARY STEPS

1. Determine operator level MIPs.
2. Determine operator level MEPs.
3. Determine service provider MIPs.
4. Determine service provider MEPs.
5. Determine customer MIPs.
6. Determine customer MEPs.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Determine operator level MIPs.	Follow these steps: • Starting at lowest operator level domain, assign a MIP at every interface internal to the operator network to be visible to CFM. • Proceed to next higher operator level and assign MIPs.

	Command or Action	Purpose
		• Verify that every port that has a MIP at a lower level does not have maintenance points at a higher level. • Repeat steps a through d until all operator MIPs are determined.
Step 2	Determine operator level MEPs.	Follow these steps: • Starting at the lowest operator level domain, assign a MEP at every UNI that is part of a service instance. • Assign a MEP at the network to network interface (NNI) between operators, if there is more than one operator. • Proceed to next higher operator level and assign MEPs. • A port with a MIP at a lower level cannot have maintenance points at a higher level. A port with a MEP at a lower level should have either a MIP or MEP at a higher level.
Step 3	Determine service provider MIPs.	Follow these steps: • Starting at the lowest service provider level domain, assign service provider MIPs at the NNI between operators (if more than one). • Proceed to next higher service provider level and assign MIPs. • A port with a MIP at a lower level cannot have maintenance points at a higher level. A port with a MEP at a lower level should not have either a MIP or a MEP at a higher level.
Step 4	Determine service provider MEPs.	Follow these steps: • Starting at the lowest service provider level domain, assign a MEP at every UNI that is part of a service instance. • Proceed to next higher service provider level and assign MEPs. • A port with a MIP at a lower level cannot have maintenance points at a higher level. A port with a MEP at a lower level should have either a MIP or a MEP at a higher level.
Step 5	Determine customer MIPs.	Customer MIPs are allowed only on the UNIs at the uPEs if the service provider allows the customer to run CFM.

	Command or Action	Purpose
		Otherwise, the service provider can configure Cisco devices to block CFM frames. • Configure a MIP on every uPE, at the UNI port, in the customer maintenance domain. • Ensure the MIPs are at a maintenance level that is at least one higher than the highest level service provider domain.
Step 6	Determine customer MEPs.	Customer MEPs are on customer equipment. Assign an outward facing MEP within an outward domain at the appropriate customer level at the handoff between the service provider and the customer.

Configuring Ethernet CFM

Configuring Ethernet CFM consists of the following tasks:

Configuring CFM

This task explains minimal basic configuration for CFM.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ethernet cfm domain** *domain-name* **level** *level-id*
4. **service** *short-ma-name* **evc** *evc-name* **vlan** *vlanid* **direction down**
5. **continuity-check**
6. **continuity-check** [**interval** *cc-interval*]
7. **exit**
8. **mep archive-hold-time** *minutes*
9. **exit**
10. **ethernet cfm global**
11. **etheret cfm ieee**
12. **ethernet cfm traceroute cache**
13. **ethernet cfm traceroute cache size** *entries*
14. **ethernet cfm traceroute cache hold-time** *minutes*
15. **snmp-server enable traps ethernet cfm cc** [**mep-up**] [**mep-down**] [**config**] [**loop**] [**cross-connect**]
16. **snmp-server enable traps ethernet cfm crosscheck** [**mep-unknown** | **mep-missing** | **service-up**]
17. **end**
18. **interface** *type number*
19. **service instance** *id* **ethernet** [*evc-name*]
20. **encapsulation** *encapsulation-type*
21. **bridge-domain** *bridge-id*

22. **cfm mep domain** *domain-name* **mpid** *id*
23. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ethernet cfm domain** *domain-name* **level** *level-id* **Example:** `Device(config)# ethernet cfm domain Customer level 7`	Defines a CFM maintenance domain at a particular maintenance level and enters Ethernet CFM configuration mode.
Step 4	**service** *short-ma-name* **evc** *evc-name* **vlan** *vlanid* **direction down** **Example:** `Device(config-ecfm)# service s41 evc 41 vlan 41 direction down`	Configures a maintenance association within a maintenance domain and enters Ethernet connectivity fault management (CFM) service configuration mode. **Note** The **direction down** is used only for Down or Outward-facing MEPs. For Up MEPs or Inward-facing MEPs, do not specify **direction down**. **Note** To configure MA CFM service for EoMPLS, use **service** *shoet-ma-name* **evc** *evc-name*.
Step 5	**continuity-check** **Example:** `Device(config-ecfm-srv)# continuity-check`	Enables the transmission of continuity check messages (CCMs).
Step 6	**continuity-check** [**interval** *cc-interval*] **Example:** `Device(config-ecfm-srv)# continuity-check interval 10s`	Configures the time period between CCMs transmission. The default interval is 10 seconds.
Step 7	**exit** **Example:** `Device(config-ecfm-srv)# exit`	Returns to Ethernet connectivity fault management configuration mode.
Step 8	**mep archive-hold-time** *minutes* **Example:** `Device(config-ecfm)# mep archive-hold-time 60`	Sets the amount of time that data from a missing MEP is kept in the continuity check database or that entries are held in the error database before they are purged.

	Command or Action	Purpose
Step 9	**exit** **Example:** `Device(config-ecfm)# exit`	Returns to global configuration mode.
Step 10	**ethernet cfm global** **Example:** `Device(config)# ethernet cfm global`	Enables CFM processing globally on the device.
Step 11	**etheret cfm ieee** **Example:** `Router(config)# ethernef cfm ieee`	Enables CFM IEEE version of CFM. This command is automatically issued when the ethernet cfm global command is issued.
Step 12	**ethernet cfm traceroute cache** **Example:** `Device(config)# ethernet cfm traceroute cache`	Enables caching of CFM data learned through traceroute messages.
Step 13	**ethernet cfm traceroute cache size** *entries* **Example:** `Device(config)# ethernet cfm traceroute cache size 200`	Sets the maximum size for the CFM traceroute cache table.
Step 14	**ethernet cfm traceroute cache hold-time** *minutes* **Example:** `Device(config)# ethernet cfm traceroute cache hold-time 60`	Sets the amount of time that CFM traceroute cache entries are retained.
Step 15	**snmp-server enable traps ethernet cfm cc** [**mep-up**] [**mep-down**] [**config**] [**loop**] [**cross-connect**] **Example:** `Device(config)# snmp-server enable traps ethernet cfm cc mep-up mep-down config loop cross-connect`	Enables SNMP trap generation for Ethernet CFM continuity check events.
Step 16	**snmp-server enable traps ethernet cfm crosscheck** [**mep-unknown** | **mep-missing** | **service-up**] **Example:** `Device(config)# snmp-server enable traps ethernet cfm crosscheck mep-unknown mep-missing service-up`	Enables SNMP trap generation for Ethernet CFM continuity check events in relation to the cross-check operation between statically configured MEPS and those learned via CCMs.

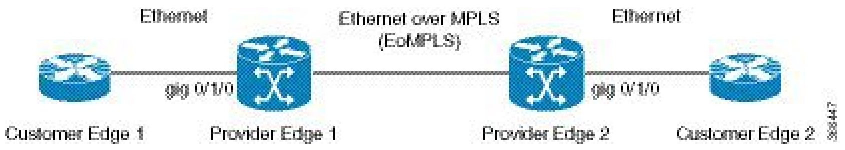
Step 17	**end**
	Example:
	`Device(config)# end`
		Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 18	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet0/0/1`	Specifies an interface and enters interface configuration mode.
Step 19	**service instance** *id* **ethernet** [*evc-name*] **Example:** `Device(config-if)# service instance 333 ethernet evc1`	Configures an Ethernet service instance on an interface and enters Ethernet service configuration mode.
Step 20	**encapsulation** *encapsulation-type* **Example:** `Device(config-if-srv)# encapsulation dot1q 5`	Sets the encapsulation method used by the interface.
Step 21	**bridge-domain** *bridge-id* **Example:** `Device(config-if-srv)# bridge-domain 100`	Binds a service instance to a bridge domain instance.
Step 22	**cfm mep domain** *domain-name* **mpid** *id* **Example:** `Device(config-if-srv)# cfm mep domain L4 mpid 4001`	Configures the MEP domain and the ID.

Step 23	**end**
	Example:
	`Device(config-if-srv)# end`
		Returns to privileged EXEC mode.

Example: Configuring CFM

The below example explains CFM configuration over Layer2 VPN (EoMPLS) network.



Example For Configuring CFM over EoMPLS

Note Ensure that EoMPLS configuration are UP and running before configuring CFM.

PE1 Configuration

```
ethernet cfm ieee
ethernet cfm global                   ! enable CFM on the router
ethernet cfm domain PE1-2 level 6     ! define domain PE1-2
 17

service EVC-PE-200 evc evc-200
```

```
  continuity-check
  continuity-check interval 1s
!

ethernet cfm logging
ethernet evc evc-200
!
interface GigabitEthernet0/1 /0
no ip address
negotiation auto
   service instance 200 ethernet evc-200
     encapsulation dot1q 200-300
     cfm mep domain PE1-2 mpid 1200                ! created MEP

exit
interface pseudowire 200
 encapsualtion mpls
 neighbor 10.10.4.4
!
l2vpn xconnect context PW200
 member GigabitEthernet0/1/0 service-instance 200
 member 10.10.4.4 200 encapsulation mpls
```

PE2 Configuration

```
ethernet cfm ieee
ethernet cfm global         ! enable CFM on the router
ethernet cfm domain PE1-2 level 6
service EVC-PE-200 evc evc-200
 continuity-check
 continuity-check interval 1s

!
ethernet cfm logging
ethernet evc evc-200
!
interface GigabitEthernet0/1/0
  no ip address
  negotiation auto
  service instance 200 ethernet evc -200
    encapsulation dot1q 200-300
    cfm mep domain PE1-2 mpid 1201   ! mpid must be different from remote end


!
interface pseudowire200
  encapsulation mpls
  neighbor 10.10.3.3 200


!
l2vpn xconnect context PW200
member gigabitethernet0/1/0
service-instance 200 pseudowire200
```

Example for Verifying CFM

show ethernet cfm maintenance-points local

```
Router# show ethernet cfm maintenance-points local
Local MEPs:
--------------------------------------------------------------------------------
MPID Domain Name                                 Lvl   MacAddress     Type CC
```

```
Ofld Domain Id                                   Dir   Port          Id
MA Name                                                SrvcInst      Source
EVC name
---------------------------------------------------------------------------------


1201 PE1-2                                        6     7010.5c51.a4bf XCON Y
No  PE1-2                                         Up    Gi0/1/0       N/A
EVC-PE-200                                              200           Static
evc-200
Total Local MEPs: 1
```

show ethernet cfm maintenance-points remote

```
ASR903-PE2# show ethernet cfm maintenance-points remote
--------------------------------------------------------------------------------
MPID  Domain Name                                 MacAddress          IfSt PtSt
Lvl   Domain ID                                   Ingress
RDI   MA Name                                     Type Id             SrvcInst
      EVC Name                                                        Age
      Local MEP Info
--------------------------------------------------------------------------------
1200  PE1-2                                      7010.5c51.8fbf       Up    Up
6    PE1-2                                       Gi0/1/0:(10.10.3.3, 200)
-    EVC-PE-200                                  XCON N/A            200
     evc-200                                                         0s
     MPID: 1201 Domain: PE1-2 MA: EVC-PE-200
Total Remote MEPs: 1
```

CFM Use Cases

Example For Configuring CFM over Bridge Domain

```
ethernet cfm ieee
ethernet cfm global
ethernet cfm domain cust1 level 7
  service s1 evc 1 vlan 1
   continuity-check
   continuity-check interval 3.3ms

service instance 1 ethernet 1
  encapsulation dot1q 1
  bridge-domain 1
  cfm mep domain cust1 mpid 1
```

Example For Configuring CFM over Trunk EFP

Note For trunk EFP, MEP is configured under the interface level configuration.

```
ethernet cfm domain oper2 level 7
service strunk evc 1000 vlan 800 direction down
  continuity-check
  continuity-check interval 3.3ms

ethernet cfm mep domain oper2 mpid 8191 service strunk --- this creates MEP
service instance trunk 1000 ethernet
  encapsulation dot1q 500-1000
  rewrite ingress tag pop 1 symmetric
  bridge-domain from-encapsulation
```

Example For Configuring CFM over VPLS

Note The EVC name used should be similar to the EVC configured in CFM configuration.

CFM over VPLS: Using the **legacy l2 vfi** command

```
ethernet cfm ieee
ethernet cfm global
ethernet cfm domain dom01 level 5

service serv01 evc evc26 vlan 26
  continuity-check
  continuity-check interval 3.3ms

service instance 26 ethernet evc26
  encapsulation dot1q 26
  rewrite ingress tag pop 1 symmetric
  bridge-domain 26
  cfm mep domain dom01 mpid 1

l2 vfi test manual evc26 ===== The evc name should be same as configuredin CFM config
vpn id 26
bridge-domain 26
neighbor 2.2.2.2 encapsulation mpls
```

CFM over VPLS: Using l2vpn vfi context command

```
ethernet cfm ieee
ethernet cfm global
ethernet cfm domain dom01 level 5
service serv01 evc evc26 vlan 26
  continuity-check
  continuity-check interval 3.3ms
l2vpn vfi context vpls26
  vpn id 26
  evc evc26
  member 2.2.2.2 encapsulation mpls
  member 1.1.1.1 encapsulation mpls
Int gi0/0/1
Service instance 26 ethernet evc26
Encapsulation dot1q 26
cfm mep domain dom01 mpid 1

bridge-domain 26
  member GigabitEthernet0/0/1 service-instance 26
  member vfi vpls26
```

Note The EVC name used should be similar to the EVC configured in CFM configuration.

Example For Configuring CFM over Default Encapsulation

```
ethernet cfm domain oper2 level 7
service cust1 evc 1000 vlan 1500 direction down
  continuity-check
  continuity-check interval 3.3ms

service instance 1000 ethernet 1000
```

```
  encapsulation default
  bridge-domain 1500
  cfm mep domain cust1 mpid 8191
  cfm encapsulation dot1q 1500
```

Verification Commands for CFM

Use the following commands to verify CFM:

- **show ethernet cfm maintenance-points local**
- **show ethernet cfm maintenance-points remote**
- **show ethernet cfm statistics**
- **show ethernet cfm ccm-learning-database**
- **show ethernet cfm errors**

SNMP Traps

The support provided by the Cisco IOS XE software implementation of Ethernet CFM traps is Cisco proprietary information. MEPs generate two types of Simple Network Management Protocol (SNMP) traps, continuity check (CC) traps and cross-check traps.

CC Traps

- MEP up--Sent when a new MEP is discovered, the status of a remote port changes, or connectivity from a previously discovered MEP is restored after interruption.
- MEP down--Sent when a timeout or last gasp event occurs.
- Cross-connect--Sent when a service ID does not match the VLAN.
- Loop--Sent when a MEP receives its own CCMs.
- Configuration error--Sent when a MEP receives a continuity check with an overlapping MPID.

Cross-Check Traps

- Service up--Sent when all expected remote MEPs are up in time.
- MEP missing--Sent when an expected MEP is down.
- Unknown MEP--Sent when a CCM is received from an unexpected MEP.

Steps to Generate SNMP Traps for CFM

To generate SNMP traps, following commands need to be configured on the router.

```
ethernet cfm logging
logging snmp-trap 0 7
logging history debugging
```

Note If syslog trap is enabled, by default trap is generated for messages of severity level emergency, alert, critical, error and warning (0-4). For other severity levels need to enable **logging snmp-trap 0 7** and **logging history debugging**

```
Router(config)#ethernet cfm logging
Router(config)#logging snmp-trap 0 7
Router(config)#logging history debugging
Router(config)#
```

Logs for MEP going DOWN

```
Console-logs:

Router(config)#
*Oct 26 21:32:06.663 IST: %E_CFM-3-REMOTE_MEP_DOWN: Remote MEP mpid 10 evc 2 vlan 2 MA name
 s2 in domain cust2 changed state to down with event code TimeOut.
*Oct 26 21:32:06.664 IST: %E_CFM-6-ENTER_AIS: local mep with mpid 20 level 2 BD/VLAN 2 dir
 D Interface Te0/3/1 enters AIS defect condition
*Oct 26 21:32:09.147 IST: %E_CFM-3-FAULT_ALARM: A fault has occurred in the network for the
 local MEP having mpid 20 evc 2 vlan 2 for service MA name s2 with the event code
DefRemoteCCM.
```

SNMP Server Side Logs

Received SNMPv2c Trap

```
Community: public
From: 7.32.22.154
sysUpTimeInstance = 04:00:54.27
snmpTrapOID.0 = clogMessageGenerated
clogHistFacility.76 = E_CFM
clogHistSeverity.76 = error(4)
clogHistMsgName.76 = REMOTE_MEP_DOWN
clogHistMsgText.76 = Remote MEP mpid 10 evc 2 vlan 2 MA name s2 in domain cust2 changed
state to down with event code TimeOut.
clogHistTimestamp.76 = 04:00:54.27
```

Received SNMPv2c Trap

```
Community: public
From: 7.32.22.154
sysUpTimeInstance = 04:00:54.27
snmpTrapOID.0 = clogMessageGenerated
clogHistFacility.77 = E_CFM
clogHistSeverity.77 = info(7)
clogHistMsgName.77 = ENTER_AIS
clogHistMsgText.77 = local mep with mpid 20 level 2 BD/VLAN 2 dir D Interface Te0/3/1 enters
 AIS defect condition
clogHistTimestamp.77 = 04:00:54.27
```

Received SNMPv2c Trap

```
Community: public
From: 7.32.22.154
sysUpTimeInstance = 04:00:56.75
snmpTrapOID.0 = dot1agCfmFaultAlarm
dot1agCfmMepHighestPrDefect.10.2.20 = defRemoteCCM(3)
```

Received SNMPv2c Trap

```
Community: public
From: 7.32.22.154
sysUpTimeInstance = 04:00:56.75
snmpTrapOID.0 = clogMessageGenerated
clogHistFacility.78 = E_CFM
clogHistSeverity.78 = error(4)
clogHistMsgName.78 = FAULT_ALARM
clogHistMsgText.78 = A fault has occurred in the network for the local MEP having mpid 20
evc 2 vlan 2 for service MA name s2 with the event code DefRemoteCCM.
clogHistTimestamp.78 = 04:00:56.75
```

Logs for MEP Coming Up

Console-logs

```
================================================
Router(config)#
*Oct 26 21:35:03.780 IST: %E_CFM-6-REMOTE_MEP_UP: Continuity Check message is received from
 a remote MEP with mpid 10 evc 2 vlan 2 MA name s2 domain cust2 interface status Up event
code Returning.
*Oct 26 21:35:03.781 IST: %E_CFM-6-EXIT_AIS: local mep with mpid 20 level 2 BD/VLAN 2 dir
D Interface Te0/3/1 exited AIS defect condition
```

SNMP Server Side Logs

Received SNMPv2c Trap

```
================================================
Community: public
From: 7.32.22.154
sysUpTimeInstance = 04:03:51.39
snmpTrapOID.0 = clogMessageGenerated
clogHistFacility.79 = E_CFM
clogHistSeverity.79 = info(7)
clogHistMsgName.79 = REMOTE_MEP_UP
clogHistMsgText.79 = Continuity Check message is received from a remote MEP with mpid 10
evc 2 vlan 2 MA name s2 domain cust2 interface status Up event code Returning.
clogHistTimestamp.79 = 04:03:51.38
```

Received SNMPv2c Trap

```
Community: public
From: 7.32.22.154
sysUpTimeInstance = 04:03:51.39
snmpTrapOID.0 = clogMessageGenerated
```

```
clogHistFacility.80 = E_CFM
clogHistSeverity.80 = info(7)
clogHistMsgName.80 = EXIT_AIS
clogHistMsgText.80 = local mep with mpid 20 level 2 BD/VLAN 2 dir D Interface Te0/3/1 exited
 AIS defect condition
clogHistTimestamp.80 = 04:03:51.38
```

Configuring and Enabling Cross-Checking for MEP

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **mep crosscheck mpid** *id* **vlan** *vlan-id* [**mac** *mac-address*]
4. **ethernet cfm mep crosscheck start-delay** *delay*
5. **ethernet cfm mep crosscheck** {**enable** | **disable**} **level** {*level-id* | *level-id-level-id* [,*level-id-level-id*]} **vlan** {*vlan-id* | **any** | *vlan-id-vlan-id* [,*vlan-id-vlan-id*]}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**mep crosscheck mpid** *id* **vlan** *vlan-id* [**mac** *mac-address*] **Example:** `Device(config-ether-cfm)# mep crosscheck mpid 402 vlan 100`	Statically defines a remote MEP on a specified VLAN within the domain.
Step 4	**ethernet cfm mep crosscheck start-delay** *delay* **Example:** `Device(config)# ethernet cfm mep crosscheck start-delay 60`	Configures the maximum amount of time that the device waits for remote MEPs to come up before the cross-check operation is started
Step 5	**ethernet cfm mep crosscheck** {**enable** | **disable**} **level** {*level-id* | *level-id-level-id* [,*level-id-level-id*]} **vlan** {*vlan-id* | **any** | *vlan-id-vlan-id* [,*vlan-id-vlan-id*]} **Example:** `Device# ethernet cfm mep crosscheck enable level 4 vlan 100`	Enables cross-checking between remote MEPs in the domain and MEPs learned through CCMs.

Configuring Cross-checking on MEP

```
Router(config)# ethernet cfm domain ServiceProvider level 4
ethernet cfm domain ServiceProvider level 4
mep crosscheck mpid 402 vlan 100
!
ethernet cfm mep crosscheck start-delay 60
```

Troubleshooting Tips

To verify and isolate a fault, start at the highest level maintenance domain and do the following:

- Check the device error status.
- When an error exists, perform a loopback test to confirm the error.
- Run a traceroute to the destination to isolate the fault.
- If the fault is identified, correct the fault.
- If the fault is not identified, go to the next lower maintenance domain and repeat these four steps at that maintenance domain level.
- Repeat the first four steps, as needed, to identify and correct the fault.

Troubleshooting CFM Features

Provides troubleshooting solutions for the CFM features.

Table 3: Troubleshooting Scenarios for CFM Features

Problem	**Solution**
CFM configuration errors	CFM configuration error occurs when when a MEP receives a continuity check with an overlapping MPID. To verify the source of the error, use the command **show ethernet cfm errors configuration** or **show ethernet cfm errors**.
CFM ping and traceroute result is "not found"	Complete these steps: **1.** Use **show run | i ethernet cfm** to view all CFM global configurations. **2.** Use **show ethernet cfm statistics** to view local MEPs and their CCM statistics **3.** Use **show ethernet cfm peer meps** command to View CFM CCM received from Peer MEPs. **4.** Use **trace ethernet cfm** command to start a CFM trace.

Problem	Solution
CFM connectivity is down and issues at the maintenance domain levels	Use the **ping ethernet {mac-address | mpid** *id* **| multicast} domain** *domain-name* { **vlan** v*lan-id* **| port | evc** *evc-name***} or the traceroute ethernet {mac-address | mpid** *id* **} domain** *domain-name* { **vlan** *vlan-id* **| port | evc evc-name}** commands to verify ethernet CFM connectivity. Share the output with TAC for further investigation.

Loop trap error	Use the **show ethernet cfm error** command to check for Loop Trap errors as shown here:
	`CE(config-if)#do sh ethernet cfm err`
	`-----`
	`Level Vlan MPID Remote MAC     Reason`
	`        Service ID`
	`-----`
	`5     711  550  1001.1001.1001 Loop Trap Error`
	`      OUT`
	`PE#sh ethernet cfm err`
	`-----`
	`Level Vlan MPID Remote MAC     Reason`
	`        Service ID`
	`-----`
	`5     711  550  1001.1001.1001 Loop Trap Error`
	`      OUT`

ethernet cfm logging	In a scale scenario, you configure either the console logging rate-limiting using **logging rate-limit** or using **logging buffered** instead of using **logging console**. The suggested rate-limit is around 30 messages per second.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases

Standards and RFCs

Standard/RFC	Title
No specific Standards and RFCs are supported by the features in this document.	—

MIBs

MIB	MIBs Link
—	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

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

Glossary

CCM—continuity check message. A multicast CFM frame that a MEP transmits periodically to ensure continuity across the maintenance entities to which the transmitting MEP belongs, at the MA level on which the CCM is sent. No reply is sent in response to receiving a CCM.

EVC—Ethernet virtual connection. An association of two or more user-network interfaces.

fault alarm—An out-of-band signal, typically an SNMP notification, that notifies a system administrator of a connectivity failure.

inward-facing MEP—A MEP that resides in a bridge and transmits to and receives CFM messages from the direction of the bridge relay entity.

maintenance domain—The network or part of the network belonging to a single administration for which faults in connectivity are to be managed. The boundary of a maintenance domain is defined by a set of DSAPs, each of which may become a point of connectivity to a service instance.

maintenance domain name—The unique identifier of a domain that CFM is to protect against accidental concatenation of service instances.

MEP—maintenance endpoint. An actively managed CFM entity associated with a specific DSAP of a service instance, which can generate and receive CFM frames and track any responses. It is an endpoint of a single MA, and terminates a separate maintenance entity for each of the other MEPs in the same MA.

MEP CCDB—A database, maintained by every MEP, that maintains received information about other MEPs in the maintenance domain.

MIP—maintenance intermediate point. A CFM entity, associated with a specific pair of ISS SAPs or EISS Service Access Points, which reacts and responds to CFM frames. It is associated with a single maintenance association and is an intermediate point within one or more maintenance entities.

MIP CCDB—A database of information about the MEPs in the maintenance domain. The MIP CCDB can be maintained by a MIP.

MP—maintenance point. Either a MEP or a MIP.

MPID—maintenance endpoint identifier. A small integer, unique over a given MA, that identifies a specific MEP.

OAM—operations, administration, and maintenance. A term used by several standards bodies to describe protocols and procedures for operating, administrating, and maintaining networks. Examples are ATM OAM and IEEE Std. 802.3ah OAM.

operator—Entity that provides a service provider a single network of provider bridges or a single Layer 2 or Layer 3 backbone network. An operator may be identical to or a part of the same organization as the service provider. For purposes of IEEE P802.1ag, Draft Standard for Local and Metropolitan Area Networks, the operator and service provider are presumed to be separate organizations.

Terms such as “customer,” “service provider,” and “operator” reflect common business relationships among organizations and individuals that use equipment implemented in accordance with IEEE P802.1ag.

UNI—user-network interface. A common term for the connection point between an operator's bridge and customer equipment. A UNI often includes a C-VLAN-aware bridge component. The term UNI is used broadly in the IEEE P802.1ag standard when the purpose for various features of CFM are explained. UNI has no normative meaning.

CHAPTER 7

Transparent CFM

CFM support on a customer VLAN (C-VLAN) allows a customer to provision maintenance intermediate points (MIPs) and Up maintenance endpoints (MEPs) on a C-VLAN component for EFP (Q-in-Q interfaces with dot1q or dot1ad C-UNI). MIPs and Up MEPs provide a customer with visibility to network traffic on the C-VLAN. CFM support on a C-VLAN also provides a common point for service verification and standardizes the user-network interface (UNI).

Transparent CFM is a mechanism to provide transparency on CFM frames between customer ends. Transparency helps the service provider network to pass the entire maintenance levels (0-7) of CFM frames from one customer end to another customer end by UP MEP that is configured on UNI-N port at any level.

Information About Transparent CFM

This section provides the information about transparent CFM.

EFP (Q-in-Q interfaces with dot1q or dot1ad C-UNI)

A EFP (Q-in-Q interfaces with dot1q or dot1ad C-UNI) represents a demarcation point between a C-VLAN space and an S-VLAN space. For purposes of CFM protocol processing, a dot1q-tunnel port is modeled as having two components: a C-VLAN component and an S- VLAN component. The C-VLAN component processes double-tagged packets from the relay-function and single-tagged packets from the wire. The S-VLAN component processes single-tagged packets from the relay-function and generates single tagged packets on relay.

CFM traffic belonging to each of the C-VLAN and S-VLAN components can be distinguished based on Ethernet layer encapsulation. This distinction allows each of the components to use the entire maintenance level range (0 to 7) without violating the maintenance domain hierarchy.

The CFM traffic generated by the C-VLAN component is transparent to the S-VLAN component if the maintenance levels of the C- VLAN component are lower than those of the S-VLAN component. The Ethernet encapsulation should be used in combination with the CFM maintenance level to determine which maintenance domain a particular traffic flow belongs to.

Benefits of Transparent CFM

The current implementation of IEEE 802.1ag CFM for EVC infrastructure provides for the provisioning of maintenance points only on S-VLANs; customers cannot monitor or troubleshoot their networks if they are provisioned on provider edge (PE) devices as aggregation nodes supporting QnQ or 802.1ad services.

The Transparent CFM support enhances the current IEEE CFM implementation by allowing customers to monitor the network at any level (0-7); CFM frames with single tag from the customer end and double tagged frames from network end are forwarded transparently.

S-VLAN Component with Transparent CFM Support

With the Transparent CFM support implemented, the S-VLAN component supports the following functions and attributes:

- Up MEPs at any level (0 to 7).
- All MEP uses the S-VLAN for processing.
- CFM frames transmitted and received by Up MEPs have a single VLAN tag (the Ethertype may be dot1q or dot1ad), and the VID is equal to the port's access VLAN (S-VLAN).

The reason for this configuration is that the dot1q-tunnel interface marks the endpoint of the S-VLAN domain; hence, its associated S-VLAN component should mark the endpoint of the CFM domain running over the S-VLAN space.

C-VLAN Component with Transparent CFM Support on C-VLANs

With the Transparent CFM support on C-VLANs feature implemented, the C-VLAN component supports the following functions and attributes:

- MIPs at any maintenance level (0 to 7).
- Transparent point functions.
- Up MEPs at any maintenance level (0 to 7).
- Up MEPs use a stack of two tags: an outer tag with a VID equal to the port's access VLAN (S-VLAN) and an inner tag with a selected C-VLAN that is allowed through the dot1q-tunnel port.

Prerequisites for Transparent CFM

- The CFM 802.1ag module must be present in the software image.

Restrictions for Transparent CFM

- Untagged encapsulation is not supported for transparent CFM.
- Transparent CFM is not supported on Smart SFP.
- Lower or similar level LTM/LTR cannot be forwarded on Transparent CFM service.
- Down MEP or Port MEP cannot be configured across EVC where Transparent CFM is configured.

- MIP on C-VLAN component is not supported.
- Transparent CFM is not supported on xConnect and VPLS.
- Transparent CFM should be configured only on Up MEPs.
- Transparent CFM is not supported on hardware offload (CCM interval < 1 second).
- The following table shows the supported rewrite combinations.

Table 4: Rewrite Combinations

Ingress Frame	Ingress Port	Egress Port
Single Tag	No Rewrite	POP1
	Translate 1-to-1	
Double Tag	No Rewrite	
	Translate 1-to-1	

Configuring Transparent CFM

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ethernet cfm global**
4. **ethernet cfm ieee**
5. **ethernet cfm domain** *domain-name* **level** *level-id*
6. **service** *MA-name* **evc** *evc-name* **vlan** *vlan-id*
7. **continuity-check**
8. **continuity-check** [*interval cc-interval*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal**	Enters global configuration mode.
Step 3	**ethernet cfm global**	Enables CFM processing globally on the device.
Step 4	**ethernet cfm ieee**	Enables the CFM IEEE version of CFM. • This command is automatically issued when the **ethernet cfm global** command is issued.

	Command or Action	Purpose
Step 5	**ethernet cfm domain** *domain-name* **level** *level-id*	Defines a CFM maintenance domain at a particular maintenance level and enters Ethernet CFM configuration mode.
Step 6	**service** *MA-name* **evc** *evc-name* **vlan** *vlan-id*	Configures an Ethernet service instance on an interface and enters EVC configuration mode.
Step 7	**continuity-check**	Enable sending and receiving of continuity check messages.
Step 8	**continuity-check** [*interval cc-interval*]	Specifies the number of continuity-check messages that are lost before CFM declares that a MEP is down (unreachable). Range is 2 to 255. Used in conjunction with **interval**.

Configuring Transparent CFM on EFP

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface gigabitethernet** *slot* / *subslot* / *port*
4. **no ip address**
5. **negotiation auto**
6. **service instance** *id* **ethernet** *evc-name*
7. **encapsulation dot1q** *vlan-id*
8. **transparent-cfm**
9. **bridge-domain** *domain-number*
10. **ethernet cfm mep domain** *domain-name* **mpid** *mpid*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal**	Enters global configuration mode.
Step 3	**interface gigabitethernet** *slot* / *subslot* / *port*	Specifies the Gigabit Ethernet or Ten Gigabit Ethernet interface to configure and enters interface configuration mode, where: • *slot* / *subslot* / *port*: The location of the interface.
Step 4	**no ip address**	Removes an IP address or disable IP processing.
Step 5	**negotiation auto**	Enables autonegotiation on a Gigabit Ethernet interface. Advertisement of flow control occurs.

	Command or Action	Purpose
Step 6	**service instance** *id* **ethernet** *evc-name*	Configures an Ethernet service instance on an interface and enters EVC configuration mode.
Step 7	**encapsulation dot1q** *vlan-id*	Configures the encapsulation. Defines the matching criteria that maps the ingress dot1q or untagged frames on an interface for the appropriate service instance. • Use the **second-dot1q** keyword and the *vlan-id* argument to specify the VLAN tags to be terminated on the subinterface
Step 8	**transparent-cfm**	Configures transparent CFM.
Step 9	**bridge-domain** *domain-number*	Binds a service instance to a bridge domain instance.
Step 10	**ethernet cfm mep domain** *domain-name* **mpid** *mpid*	Configures a CFM MEP domain.

Configuration Examples for Transparent CFM

This section shows the configuration examples for Transparent CFM.

Example: Configuration of Transparent CFM on EFP

The following example shows the sample configuration of Transparent CFM on EFP.

```
interface GigabitEthernet0/0/1
no ip address
negotiation auto
service instance 1 ethernet EVC
encapsulation dot1q 10
transparent-cfm
bridge-domain 4000
cfm mep domain MD5 mpid 100
```

Example: Configuration of Transparent CFM

The following example shows the sample output of configuration of Transparent CFM.

```
ethernet cfm ieee
ethernet cfm global
ethernet cfm alarm notification all

ethernet cfm domain MD2 level 5
service PMA2 evc evc1 vlan 4000
continuity-check
continuity-check interval 1s
!
ethernet cfm logging
ethernet evc evc1

interface TenGigabitEthernet0/0/3
no ip address
negotiation auto
service instance 1 ethernet evc1
encapsulation dot1q 10
transparent-cfm
bridge-domain 4000
cfm mep domain MD2 mpid 100
```

```
!
!
interface TenGigabitEthernet0/0/5
no ip address
negotiation auto
service instance 1 ethernet evc1
encapsulation dot1q 20
rewrite ingress tag pop 1 symmetric
bridge-domain 4000
!
!
```

CHAPTER 8

VLAN Translation with QoS

VLAN translation provides flexibility in managing VLANs and Metro Ethernet-related services.

Layer2 VPN services are required to be deployed in the following Ethernet service type constructs:

- Ethernet Line (E-Line) in E-line remote services: Provides a point-to-point Ethernet Virtual Circuit (EVC).
- Ethernet LAN (ELAN) in E-line remote services: Provides a multipoint-to-multipoint EVC.
- ELAN local
- ELAN remote

All the remote services are transported over Ethernet Over Multi Protocol Label Switching (EoMPLS) (point-to-point) or Virtual Private LAN Service VPLS (multipoint-to-multipoint) cloud. Each service can accommodate the traffic coming from the customer either with 1 tag or 2 tags. The CoS from the customer must be passed transparently through the service to the other CPEs (UNIs).

Benefits of VLAN Translation

Earlier, the router supported Rewrite Push and Pop operations to push and remove 1 or more 802.1Q tags from the service frames only. The CoS transparency could not be achieved along with VLAN tag manipulation.

This problem is solved with the VLAN Translation feature. The current implementation of the feature allows one or more 802.1Q tags to be replaced with other 802.1Q tags and thus the desired tag manipulation can be achieved. In a scenario with two EFPs egressing the same interface, each EFP can have a different VLAN rewrite operation, which is more flexible.

VLAN translation feature includes the following functionalities:

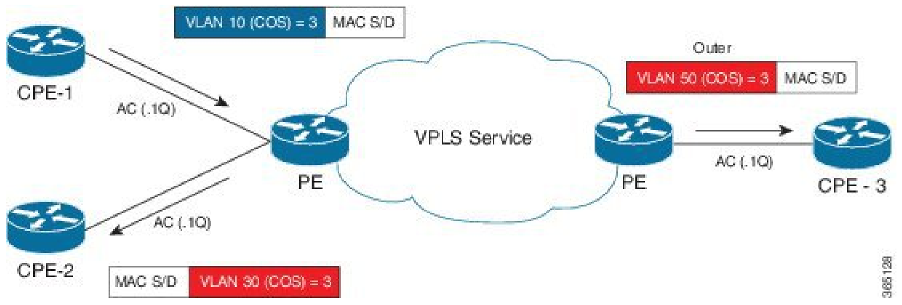
- 1:1 VLAN translation - The VLAN of the incoming traffic (CE VLAN) is replaced by another VLAN (PE VLAN). The specification of the VLAN translation happens during the creation of the service request. The CoS field of the new tag is set to the same value as the CoS field of the existing VLAN tag.
- 2:1 VLAN translation - The double tagged (Q-in-Q) traffic at the U-PE UNI port can be mapped to different flows to achieve service multiplexing. The CoS field of the new tag is set to the inner CE-VLAN (second tag) CoS value.
- 1:2 VLAN translation - The outermost tag can be replaced with two tags. The CoS field of the new tags is set to the same value as the CoS field of the incoming 802.1Q VLAN tag.
- 2:2 VLAN translation - The outermost two tags can be replaced with other two tags. The CoS field of the new tags is set to the same value as the CoS field of the incoming Q-in-Q (outer and inner tag CoS) service frame.

Scenarios showing VLAN Translation

The following scenarios show the VLAN translation.

Scenario 1 - 1:1 VLAN Translation

Figure 2: 1:1 VLAN Translation

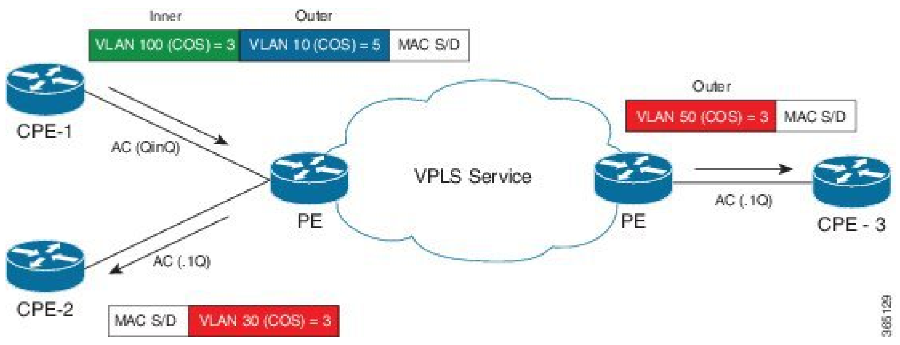


In the scenario above, t he broadcast or multicast from CPE1 has to be sent to CPE2 and CPE3. The incoming tag in the frame has a CoS value of 3. The service needs to be created that enables the CoS value to pass transparently to the other sites with the desired VLAN translation.

This behavior can be achieved using the 1:1 VLAN translation command on the service instance attached to CPE1. The Egress Service instance on Remote UPE device should be configured with the right encapsulation or Rewrite operation to achieve the correct tagging behavior (VLAN 50 on outgoing tag) for CPE3. As there is no inner tag here, the outer CoS is propagated in the newly added tag to both CPE2 and CPE3 ACs.

Scenario 2 - 2:1 VLAN Translation

Figure 3: 2:1 VLAN Translation



The above scenario depicts an instance of a local E-Line service, with one AC (AC1) with double VLAN ID (inner 100 andouter 10) and the other AC (AC2) with VLAN ID (30). The frame with CoS=3 from the inner VLAN 100 in AC1 has to be delivered in AC2 with VLAN 30 and CoS=3. Similarly, for remote instance, we have AC (AC3) with VLAN 50 and same inner CoS 3 should be transparently carried over MPLS cloud to AC3 from AC1. The way we can achieve this behavior on router is with 2:1 VLAN translation command on service instance connected to AC1.

In this particular scenario, since there is a inner Tag present, inner CoS will be propagated in the newly added Tag to both CPE2 and CPE3 ACs.

Limitations for VLAN Translation with QoS

- Only 1:1 and 2:1 translate rewrites are supported. 1:2 and 2:2 translations are not supported.
- Translate operation can only be applied to a unique tag matching service instance.
- VLAN Translation is not supported on TEFP, encapsulation untagged, and BDI interfaces.
- Any VLAN Translation with rewrite pop2 is not supported.
- Translation is only supported for 802.1Q (0x8100) encapsulation.
- Translation is not supported for 802.1AD (0x88A8) and Customer Ethertype (0x9100 and 0x9200).
- Egress QoS policy is not supported on Trans 2:1 and 1:1 VLAN Translation, if ingress Translation or push EFPs do not have policy.
- For 1:1 to 1:1 scenario, marking is not supported.
- Ingress POP 0 or 1:1 CoS marking is not supported.

Configuring 1:1 VLAN Translation

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** <*interface-number*>
4. **service instance** *id* **ethernet** {*evc-id*}

5. **encapsulation dot1q** {*vlan-id*}
6. **rewrite ingress tag translate 1-to-1 dot1q** *vlan-id* **symmetric**
7. **bridge-domain** *domain-number*
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal**	Enters global configuration mode.
Step 3	**interface** <*interface-number*>	Enters the interface configuration mode for the interface connected to the service-provider network. You can enter a physical interface or an EtherChannel port channel.
Step 4	**service instance** *id* **ethernet** {*evc-id*}	Configures an Ethernet service instance on the interface and enters Ethernet service configuration mode. • The Ethernet service instance identifier is a per-interface service identifier and does not map to a VLAN.
Step 5	**encapsulation dot1q** {*vlan-id*}	Configures the encapsulation. Defines the matching criteria that maps the ingress dot1q or untagged frames on an interface for the appropriate service instance.
Step 6	**rewrite ingress tag translate 1-to-1 dot1q** *vlan-id* **symmetric**	Specifies the encapsulation adjustment that is to be performed on the frame ingress to the service instance.
Step 7	**bridge-domain** *domain-number*	Binds a service instance to a bridge domain instance.
Step 8	**end**	Returns to privileged EXEC mode.

Configuring 2:1 VLAN Translation

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** <*interface-number*>
4. **service instance** *id* **ethernet** {*evc-id*}
5. **encapsulation dot1q** {*vlan-id*} **second-dot1q** {*vlan-id*}
6. **rewrite ingress tag translate 2-to-1 dot1q** *vlan-id* **symmetric**
7. **bridge-domain** *domain-number*
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** <*interface-number*>	Enters the interface configuration mode for the interface connected to the service-provider network. You can enter a physical interface or an EtherChannel port channel.
Step 4	**service instance** *id* **ethernet** {*evc-id*}	Configures an Ethernet service instance on the interface and enters Ethernet service configuration mode. • The Ethernet service instance identifier is a per-interface service identifier and does not map to a VLAN.
Step 5	**encapsulation dot1q** {*vlan-id*} **second-dot1q** {*vlan-id*}	Configures the encapsulation. Defines the matching criteria that maps the ingress dot1q or untagged frames on an interface for the appropriate service instance. • Use the **second-dot1q** keyword and the *vlan-id* argument to specify the VLAN tags to be terminated on the subinterface.
Step 6	**rewrite ingress tag translate 2-to-1 dot1q** *vlan-id* **symmetric**	Specifies the encapsulation adjustment that is to be performed on the frame ingress to the service instance.
Step 7	**bridge-domain** *domain-number*	Binds a service instance to a bridge domain instance.
Step 8	**end**	Returns to privileged EXEC mode.

Configuring policy for ingress QoS

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **class-map match-all** *cos value*
4. **match cos** *value*
5. **policy-map** *policy-name*
6. **class** *class-name*
7. **set cos** *cos value*

8. **police cir** *value*
9. **interface** *interface-number*
10. **no ip address**
11. **load-interval** *seconds*
12. **service instance** *id* **ethernet** *evc-id*
13. **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id*
14. **rewrite ingress tag translate 2-to-1 dot1q** *vlan-id* **symmetric**
15. **service-policy input** *policy-map-name*
16. **bridge-domain** *domain-number*
17. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal**	Enters global configuration mode.
Step 3	**class-map match-all** *cos value*	Determine how packets are evaluated when the packets meet all of the match criteria.
Step 4	**match cos** *value*	Matches a packet on the basis of a layer 2 CoS marking,
Step 5	**policy-map** *policy-name*	Creates or specifies the name of the traffic policy and enters policy-map configuration mode.
Step 6	**class** *class-name*	Specifies the name of a traffic class and enters policy-map class configuration mode.
Step 7	**set cos** *cos value*	Sets the Class of Service (CoS) value of an outgoing packet.
Step 8	**police cir** *value*	Need Information.
Step 9	**interface** *interface-number*	Enters the interface configuration mode for the interface connected to the service-provider network. You can enter a physical interface or an EtherChannel port channel.
Step 10	**no ip address**	Removes an IP address or disable IP processing.
Step 11	**load-interval** *seconds*	Changes the sampling interval for statistics collections on interfaces
Step 12	**service instance** *id* **ethernet** *evc-id*	Configures an Ethernet service instance on the interface and enters Ethernet service configuration mode. • The Ethernet service instance identifier is a per-interface service identifier and does not map to a VLAN.

	Command or Action	Purpose
Step 13	**encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id*	Configures the encapsulation. Defines the matching criteria that maps the ingress dot1q or untagged frames on an interface for the appropriate service instance. • Use the **second-dot1q** keyword and the *vlan-id* argument to specify the VLAN tags to be terminated on the subinterface.
Step 14	**rewrite ingress tag translate 2-to-1 dot1q** *vlan-id* **symmetric**	Specifies the encapsulation adjustment that is to be performed on the frame ingress to the service instance.
Step 15	**service-policy input** *policy-map-name*	Attaches a policy map to an interface.
Step 16	**bridge-domain** *domain-number*	Binds a service instance to a bridge domain instance.
Step 17	**end**	Returns to privileged EXEC mode.

Configuration Example for 1:1 VLAN Translation

The following example shows the sample configuration for 1:1 VLAN Translation.

```
service instance 50 ethernet
encapsulation dot1q 50
rewrite ingress tag translate 1-to-1 dot1q 500 symmetric
bridge-domain 50
```

Configuration Example for 2:1 VLAN Translation

The following example shows the sample configuration for 2:1 VLAN Translation.

```
service instance 50 ethernet
encapsulation dot1q 10 second-dot1q 20
rewrite ingress tag translate 2-to-1 dot1q 500 symmetric
bridge-domain 50
```

Configuration Example for policing ingress QoS

The following example shows the sample configuration of policing ingress QoS.

```
class-map match-all cos6
match cos 6
class-map match-all cos3
match cos 3
policy-map mark_cos3to6
class cos3
set cos 6
police cir 900000000
interface TenGigabitEthernet0/0/12
no ip address
load-interval 30
service instance 1 ethernet
encapsulation dot1q 10 second-dot1q 20
```

```
rewrite ingress tag translate 2-to-1 dot1q 30 symmetric
service-policy input mark_cos3to6
bridge-domain 1
```

Configuration Verifications for VLAN Translation with QoS

The following sections show the configuration verifications for VLAN Translation with QoS.

Verifying the VLAN configuration

The **show running-config interface [number]** command displays and verifies the VLAN configuration.

```
#show running-config interface gigabitEthernet 0/0/5
interface GigabitEthernet0/0/5
 no ip address
 media-type auto-select
 negotiation auto
 service instance 1 ethernet
  encapsulation dot1q 1
  rewrite ingress tag translate 1-to-1 dot1q 2 symmetric
  bridge-domain 1
 end
```

Verifying policy-map on ingress QoS

The **show policy-map interface** command verifies the policy-map on ingress QoS.

```
show policy-map interface gig0/0/3 service instance 1
 GigabitEthernet0/0/3: EFP 1

  Service-policy input: in_policy_cos

    Class-map: cos3 (match-all)
      7077065 packets, 452932160 bytes
      30 second offered rate 19984000 bps, drop rate 0000 bps
      Match: cos  3
      QoS Set
        cos 4
          Marker statistics: Disabled

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      30 second offered rate 0000 bps, drop rate 0000 bps
      Match: any
```

Verifying policy-map on egress QoS

The **show policy-map interface** command verifies the policy-map on egress QoS.

```
show policy-map interface gig0/0/4 service instance 1
 GigabitEthernet0/0/4: EFP 1

  Service-policy output: classify_policy

    Class-map: class_cos4 (match-all)
```

```
        6891220 packets, 468602960 bytes
        30 second offered rate 21359000 bps
        Match: cos  4

      Class-map: class-default (match-any)
        0 packets, 0 bytes
        30 second offered rate 0000 bps, drop rate 0000 bps
        Match: any
```

Verifying the QoS Labels

The **show platform hardware pp active feature qos label structs** command displays the QoS labels in use.

```
#show platform hardware pp active feature qos label structs
PRINTING BIT LIST OF LABELS IN USE
0-3,8-15,125-127
Qos Label = 1, Ref_count = 1, Set_ref_count =1, edir = 0
Label Key is as follows -
outer_dscp = 0, inner_dscp = 0,outer_cos = 0, inner_cos = 0
 outer_cfi = 0, inner_cfi = 0,outer_exp = 0, inner_exp = 5
mpls_tunnel_bit = 1, qos_group = 0,discard_class = 0, rwtype = 0, set_action = 1
Match criteria bit list for this label:
8,11
PRINTING BIT LIST OF LABELS IN USE
0-3,8-15,125-127
Qos Label = 2, Ref_count = 1, Set_ref_count =1, edir = 0
Label Key is as follows -
outer_dscp = 0, inner_dscp = 0,outer_cos = 0, inner_cos = 0
 outer_cfi = 0, inner_cfi = 0,outer_exp = 0, inner_exp = 0
mpls_tunnel_bit = 1, qos_group = 0,discard_class = 0, rwtype = 0, set_action = 1
Match criteria bit list for this label:
8,11
```

Verifying Egress TCAM Details

The **show platform hardware pp active feature qos tcam eqos 0 all** command displays and verifies the egress TCAM details.

```
#show platform hardware pp active feature qos tcam eqos 0 all
FIELD 0: total 125, used 60, min 125, first_entry 0, hole:0, size:0
=========================================================================
FIELD 1: total 0, used 0, min 0, first_entry 125, hole:0, size:0
=========================================================================
FIELD 2: total 799, used 0, min 0, first_entry 125, hole:0, size:0
=========================================================================
FIELD 3: total 50, used 0, min 50, first_entry 924, hole:0, size:0
=========================================================================
index 0: 1 contiguous entries in same hw_handle, aclType EGRESSCLASSIFY,lookupTable NA
index 1: 1 contiguous entries in same hw_handle, aclType EGRESSCLASSIFY,lookupTable NA
index 2: 1 contiguous entries in same hw_handle, aclType EGRESSCLASSIFY,lookupTable NA
index 3: 1 contiguous entries in same hw_handle, aclType EGRESSCLASSIFY,lookupTable NA
index 4: 1 contiguous entries in same hw_handle, aclType EGRESSCLASSIFY,lookupTable NA
```

Verifying TCAM Index Details

The **show platform hardware pp active feature qos tree service-instance <num> port-number <num> input tcam-info** command displays the TCAM index details pertaining to the specified interface.

```
Tcam-handle=2253
        First-Index=184
```

```
Last-Index=249
Total-Count=66
```

CHAPTER 9

G.8032 and CFM Support for Microwave Adaptive Bandwidth

The G.8032 and CFM Support for Microwave Adaptive Bandwidth feature enables the G.8032 Ethernet Protection Ring (ERP) mechanism to be used as a trigger in response to bandwidth degradation occurrences (such as a signal degradation [SD] indicator) on microwave links. Ethernet Connectivity Fault Management (CFM) interacts with the microwave transceiver to continuously check the quality and the bandwidth of the microwave link. When microwave link degradation (based on the configured service level agreement [SLA] in use) is detected, CFM notifies the Embedded Event Manager (EEM), which in turn notifies a mechanism such as, G.8032 ERP. G.8032 ERP ensures that the degraded microwave link is bypassed and no longer used. The degraded microwave link can still be used by one or more of the G.8032 ERP instances. Only the affected G.8032 ERP instances are switched to alternate link.

Prerequisites for G.8032 and CFM Microwave Adaptive Bandwidth Support

- The microwave transceiver in the network topology must support adaptive bandwidth modulation, and the microwave transceiver must support the Ethernet Connectivity Fault Management (CFM) extension for microwave devices as defined by Cisco.
- All devices connected directly to the microwave transceiver must support signal degradation (SD) functions. Devices not connected directly to the microwave transceiver can be standard-compliant nodes or enhanced SD-capable nodes.
- In any homogeneous ring topology, all links must be microwave links and all devices must support microwave SD-based ring protection.
- A ring topology with multiple microwave links can experience a signal degradation condition on one or more of the microwave links. Only one signal degradation condition per ring instance is supported. This support is provided on a first-come, first-serve basis, per ring instance.

Limitations for G8032

Follow the rules below while adding or deleting VLANsin inclusion list:

- While adding a VLAN, first add the VLAN in the interface and then in add the VLAN in G8032 inclusion list.
- While removing a VLAN, VLAN has to be first removed from the G8032 inclusion list and then from the interface trunk.

Addition or deletion of VLANs in exclusion list is not supported.

About G.8032 and CFM Support for Microwave Adaptive Bandwidth

Microwave Adaptive Bandwidth Feature Functionality

The G.8032 and CFM Support for Microwave Adaptive Bandwidth feature extends the functionality of the G.8032 Ethernet Protection Ring (ERP) mechanism and Ethernet Connectivity Fault Management (CFM).

This feature enables the G.8032 ERP mechanism to be used as a trigger in response to bandwidth degradation occurrences (such as a signal degradation [SD] indicator) on microwave links. Ethernet CFM interacts with the microwave transceiver to continuously check the quality and the bandwidth of the microwave link. When microwave link degradation (based on the configured service level agreement [SLA] in use) is detected, CFM notifies the Embedded Event Manager (EEM), which in turn notifies a mechanism such as, G.8032 ERP. G.8032 ERP ensures that the degraded microwave link is bypassed and no longer used. Depending upon the severity of the signal degradation and the configured threshold, G.8032 protection switching occurs on a per-instance basis.

For more information about Ethernet CFM, see the “Configuring IEEE Standard-Compliant Ethernet CFM in a Service Provider Network” module or the “Configuring Ethernet Connectivity Fault Management in a Service Provider Network” module.

For more information about G.8032 ERP, see the “ITU-T G.8032 Ethernet Ring Protection Switching” module.

Fixed Versus Adaptive Bandwidth Modulation and the Microwave Adaptive Bandwidth Feature

Traditional microwave radios use fixed modulation schemes whereby any degradation in the wave propagation conditions (for example, due to adverse weather conditions such as heavy fog or rain) led to complete loss of the signal and a disruption of traffic. In a fixed modulation scheme, the microwave radio link had a binary state of either “'available” (on) or “unavailable” (off).

More technologically advanced microwave radios use an adaptive modulation scheme. In an adaptive modulation scheme, when the microwave link degrades due to adverse weather conditions, the radio changes its modulation scheme to a more robust scheme. The radio continues to broadcast but with less capacity. As a result, the radio can be in several capacity or bandwidth states, and not just on or off.

In the case of microwave links with adaptive modulation, the control Operation, Administration, and Maintenance (OAM) protocols are unable to make best use of the available bandwidth due of the following OAM characteristics:

- If the protocol used for failure detection is tagged as high-priority traffic, the OAM frames bypass the degraded (congested) microwave links and no protection switching is triggered.

- If the protocol used for failure detection is tagged as low-priority traffic, then momentary congestion over the native Ethernet (that is, the nonmicrowave) links could lead to loss of continuity and spurious protection switching.

Even though the network topology must be provisioned with enough redundant bandwidth to handle a complete failure, in certain situations where the service committed information rate (CIR) is very low, forwarding as much excess traffic (above the CIR) as possible is important. Therefore, for those situations, treating bandwidth degradation as a complete failure is not desirable.

Adaptive Bandwidth Multi-hop Extensions

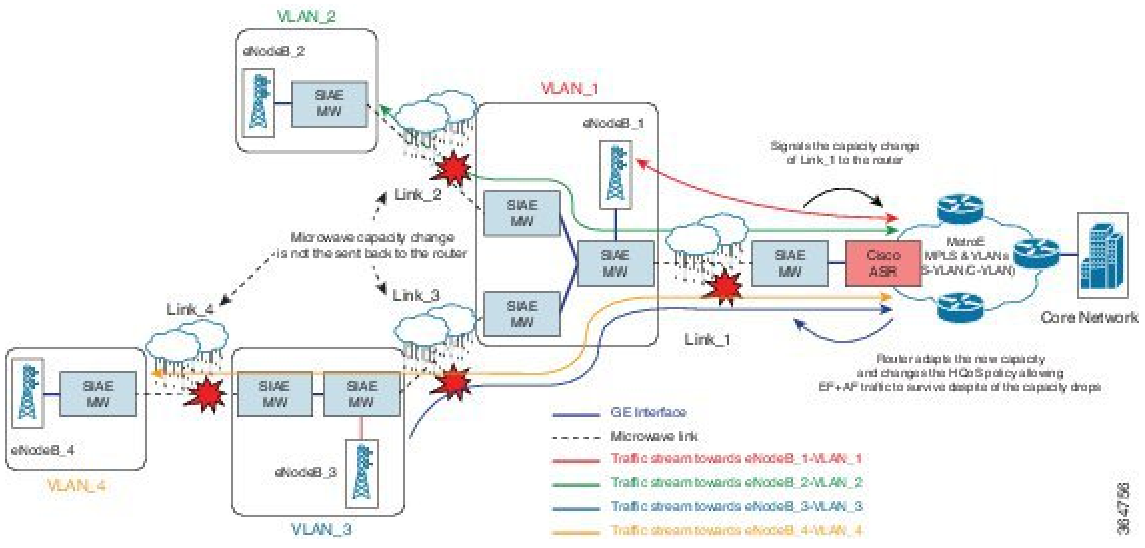
In a network topology consider a single interface on the head-end router is connected to a topology consisting of multiple microwave links either in series or in a hub-and-spoke arrangement. In such scenarios, the links degrade independently, and send their own VSMs containing current and nominal bandwidth for their links. Identifying the VSMs with the source MAC address does not help identify the degraded link.

To help identify the degraded links in a multi-hop topology, Link IDs can be configured on the VSM. The Link ID is configured in the EEM configuration using the **event ethernet microwave sd interface** command. The command support registrations on one or more individual links identified by either link ID or source MAC address.

With multi-hop topology, all individual links identified by the interface. If the interface is degraded, the links too get degraded.

The multi-hop topology is supported when multiple microwave links are grouped together into a port-channel.

Figure 4: Adaptive Bandwidth with Multi-hop Extensions


VLAN_2
eNodeB_2
SIAE MW
VLAN_1
eNodeB_1
SIAE MW
Signals the capacity change of Link_1 to the router
Link_2
Microwave capacity change is not the sent back to the router
Link_3
Link_4
SIAE MW
Cisco ASR
MetroE MPLS & VLANs S-VLAN,C-VLAN)
Core Network
Link_1
Router adapts the new capacity and changes the HQoS policy allowing EF+AF traffic to survive despite of the capacity drops
eNodeB_4
VLAN_4
eNodeB_3
VLAN_3
GE Interface
Microwave link
Traffic stream towards eNodeB_1-VLAN_1
Traffic stream towards eNodeB_2-VLAN_2
Traffic stream towards eNodeB_3-VLAN_3
Traffic stream towards eNodeB_4-VLAN_4
364756

Prerequisites for Assigning Link IDs

- If VSMs for multiple links are sent from the same source MAC address, then link IDs must be used.
- Link IDs must be unique within the network segment connected to a single physical link on the head-end router.

Restrictions for Assigning Link IDs

- If link IDs are used, the EEM scripts are registered on either a set of source MAC addresses or a set of link IDs but not a mixture of both.
- When registering an EEM script and specifying multiple links it is recommended that the threshold is equal to or lower than the minimum nominal bandwidth across all those links.

How to Configure G.8032 and CFM Support for Microwave Adaptive Bandwidth

Creating the Ethernet Microwave Event and Using G.8032 to Specify Appropriate Actions

For more information on how to configure the ethernet ring profile, see LAN Switching Configuration Guide IOS XE Release 3S (Cisco ASR 900 Series).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event ethernet microwave sd** {**interface** *type number* **threshold** *threshold-bandwidth*}
5. **action** *action-id* **switch ring g8032** *ring-name* **instance** *instance-id*
6. **event ethernet microwave clear-sd** {**interface** *type number*}
7. **action** *action-id* **switch ring g8032 clear** *ring-name* **instance** {*instance-id* | **all**}
8. Repeat steps 4 through 7 for each Ethernet microwave event you want to create. Then proceed to step 9.
9. **exit**

DETAILED STEPS

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

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Device(config)# event manager applet mw_ring_sd1`	Registers an applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 4	**event ethernet microwave sd** {**interface** *type number* **threshold** *threshold-bandwidth*} **Example:** `Device(config-applet)# event ethernet microwave sd interface gigabitethernet0/0/0 threshold 400`	Creates the Ethernet microwave signal degradation (SD) event. • After the event is created, use the **action switch ring g8032 instance** command at step 5 to specify the appropriate action to take on this event.
Step 5	**action** *action-id* **switch ring g8032** *ring-name* **instance** *instance-id* **Example:** `Device(config-applet)# action 1 switch ring g8032 ringA instance 1`	Specifies the protocol switch action for an instance on a link of a G.8032 Ethernet Protection Ring (ERP).
Step 6	**event ethernet microwave clear-sd** {**interface** *type number*} **Example:** `Device(config-applet)# event ethernet microwave clear-sd interface gigabitethernet0/0/0`	Creates the Ethernet microwave event to be associated with bandwidth SD occurrences. • After the event is created, use the **action switch ring g8032 clear instance** command at step 7 to clear the SD occurrence and bring the ring back to the normal (idle) state.

Step 7	**action** *action-id* **switch ring g8032 clear** *ring-name* **instance** {*instance-id* | **all**}
	Example:
	`Device(config-applet)# action 1 switch ring g8032 clear ringA instance 1`
		Specifies the action of clearing an SD occurrence on a link of a G.8032 Ethernet Protection Ring (ERP) topology.

Step 8	Repeat steps 4 through 7 for each Ethernet microwave event you want to create. Then proceed to step 9. **Example:** —	
Step 9	**exit** **Example:** `Device(config-applet)# exit`	Exits applet configuration mode.

Modifying Ethernet Microwave Event Settings

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type name*
4. **ethernet event microwave hold-off** *seconds*
5. **ethernet event microwave loss-threshold** *number-of-messages*
6. **ethernet event microwave wtr** *seconds*
7. **exit**
8. **show ethernet event microwave status** [**interface** *type number*]
9. **show ethernet event microwave statistics** [**interface** *type number*]
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type name* **Example:** `Device(config)# interface gigabitethernet0/0/0`	Specifies an interface and enters interface configuration mode.
Step 4	**ethernet event microwave hold-off** *seconds* **Example:** `Device(config-if)# ethernet event microwave hold-off 30`	Specifies the microwave bandwidth degradation hold-off time, in seconds. • This time is used to prevent changes in the state of the network node as a result of signal degradation (SD) occurrences.
Step 5	**ethernet event microwave loss-threshold** *number-of-messages* **Example:** `Device(config-if)# ethernet event microwave loss-threshold 3`	Specifies the number of bandwidth Vendor-Specific Messages (VSM) sent from the microwave transceiver to the Cisco device. Once the link experiences signal degradation, the microwave transceiver sends periodic bandwidth VSM messages to the Cisco device until the bandwidth is fully restored. The interval of these messages is controlled by the microwave transceiver.

	Command or Action	Purpose
		This configuration specifies the continuous bandwidth VSM messages the Cisco device misses before declaring a signal recovery event.
Step 6	**ethernet event microwave wtr** *seconds* **Example:** `Device(config-if)# ethernet event microwave wtr 45`	Specifies the wait-to-restore (WTR) time, in seconds. • This time is used to prevent changes in the state of the network node as a result of recovery events after an SD occurrence.
Step 7	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.
Step 8	**show ethernet event microwave status** [**interface** *type number*] **Example:** `Device# show ethernet event microwave status GigabitEthernet 0/0/2`	(Optional) Displays the microwave event status.
Step 9	**show ethernet event microwave statistics** [**interface** *type number*] **Example:** `Device# show ethernet event microwave statistics GigabitEthernet 0/0/2`	(Optional) Displays the microwave event statistics.

Step 10	**end**
	Example:
	`Device# end`
		Returns to user EXEC mode.

Configuration Examples for G.8032 and CFM Support for Microwave Adaptive Bandwidth

Example: Configuring the Ethernet Microwave Event

In this example, two Ethernet microwave events have been created, mw_ring_sdl1 and mw_ring_sd_2:

```
Device> enable
Device> configure terminal
Device(config)# event manager applet mw_ring_sd1
Device(config-applet)# event ethernet microwave sd interface gigabitethernet0/0/0 threshold
 400
```

```
Device(config-applet)# action 1 switch ring g8032 ringA instance 1
Device(config-applet)# exit
Device(config)# event manager applet mw_ring_sd2
Device(config-applet)# event ethernet microwave sd interface gigabitethernet0/0/0 threshold
 400
Device(config-applet)# action 1 switch ring g8032 ringA instance 2
Device(config-applet)# exit
```

In this example, a microwave event has been configured that clears all the signal degradation (SD) events, as defined by the **action switch ring g8032 clear instance all** command:

```
Device> enable
Device> configure terminal
Device(config)# event manager applet mw_ring_clear_sd
Device(config-applet)# event ethernet microwave clear-sd interface gigabitethernet0/0/0
Device(config-applet)# action 1 switch ring g8032 clear ringA instance all
Device(config-applet)# exit
```

Example: Verifying the Ethernet Microwave Event Configuration

The following is sample output from the **show ethernet event microwave status** command where GigabitEthernet interface 0/0/2 has been specified. Use the command to confirm that the configuration is performing as intended.

```
Device# show ethernet event microwave status GigabitEthernet 0/0/2

Microwave Bandwidth Status for GigabitEthernet0/0/2
 State : Degraded
 Elapsed time in this state : 1:25:33
 Nominal Bandwidth : 512Mbps
 Current Bandwidth : 256Mbps
 Lowest Bandwidth Since Entering Degraded : 64Mbps
 Last VSM Received : Oct 27 14:06:19.983
 Sender Transmit Period : 1 second
 Sender Address : 01AB.CC00.1881
 Hold Timer : Not Running
 Restore Timer : Not Running
 Periodic Timer : 2333 msec <--Calculated using the configured loss-threshold value.
 Hold Time : 0 seconds      <--This is hold-off timer value.
 Restore Time : 10 seconds  <--This is the wait-to-restore value.
 Loss-Threshold: 3
```

The following is sample output from the **show ethernet event microwave statistics** command where GigabitEthernet interface 0/0/2 has been specified:

```
Device#show ethernet event microwave statistics GigabitEthernet 0/0/2

Microwave Bandwidth Statistics for GigabitEthernet0/0/2
 Total VSM Receive Count : 145
 Total VSM Drop Count : 0
 Number of transitions into Degraded state : 2
```

Example: Signal Degraded Event Syslog Messages

This example shows the sample output of signal degraded event syslog messages

```
Apr 30 16:33:45.497 IST: %ETHERNET_EVENT-4-MW_BW_CHANGE:
Available microwave bandwidth for link with source MAC 64F6.9D67.A006, link ID 0 on
```

```
GigabitEthernet0/0/7 has changed due to VSM,
current is 500Mbps, nominal is 600Mbps.
Apr 30 16:33:45.502 IST: %HA_EM-6-LOG: DEGRADED: eem started
```

Example: Configuring the TRUNK EFP with ACM Microwave

The following example shows the configuration of MEP on a trunk EFP.

```
interface GigabitEthernet0/3/3
no ip address
negotiation auto
ethernet cfm mep domain md1 mpid 1 service ma1
service instance trunk 1 ethernet
encapsulation dot1q 1-109
rewrite ingress tag pop 1 symmetric
l2protocol peer lacp
bridge-domain from-encapsulation
!
End
```

The MEP is configured outside EFP. The corresponding domain/service configuration would look like:

```
ethernet cfm ieee
ethernet cfm global
ethernet cfm domain md1 level 1
service ma1 evc evc1 vlan 100 direction down
continuity-check
```

CHAPTER 10

Configuring Ethernet Local Management Interface at a Provider Edge

The advent of Ethernet as a metropolitan-area network (MAN) and WAN technology imposes a new set of Operation, Administration, and Management (OAM) requirements on Ethernet's traditional operations, which had centered on enterprise networks only. The expansion of Ethernet technology into the domain of service providers, where networks are substantially larger and more complex than enterprise networks and the user-base is wider, makes operational management of link uptime crucial. More importantly, the timeliness in isolating and responding to a failure becomes mandatory for normal day-to-day operations, and OAM translates directly to the competitiveness of the service provider.

The “Configuring Ethernet Local Management Interface at a Provide Edge” module provides general information about configuring an Ethernet Local Management Interface (LMI), an OAM protocol, on a provider edge (PE) device.

Prerequisites for Configuring Ethernet Local Management Interface at a Provider Edge

- Ethernet Operation, Administration, and Management (OAM) must be operational in the network.
- For Ethernet OAM to operate, the provider edge (PE) side of a connection must be running Ethernet Connectivity Fault Management (CFM) and Ethernet Local Management Interface (LMI).
- All VLANs used on a PE device to connect to a customer edge (CE) device must also be created on that CE device.
- To use nonstop forwarding (NSF) and In Service Software Upgrade (ISSU), stateful switchover (SSO) must be configured and working properly.

Restrictions for Configuring Ethernet Local Management Interface at a Provider Edge

- Ethernet Local Management Interface (LMI) is not supported on routed ports, EtherChannel port channels, ports that belong to an EtherChannel, private VLAN ports, IEEE 802.1Q tunnel ports, Ethernet over Multiprotocol Label Switching (MPLS) ports, or Ethernet Flow Points (EFPs) on trunk ports.
- Ethernet LMI cannot be configured on VLAN interfaces.
- QinQ encapsulation is not supported on the Cisco ASR 1000 Series Aggregation Services Router for CFM for routed subinterfaces.

Information About Configuring Ethernet Local Management Interface at a Provider Edge

Ethernet Virtual Circuits Overview

An Ethernet virtual circuit (EVC) as defined by the Metro Ethernet Forum is a port level point-to-point or multipoint-to-multipoint Layer 2 circuit. EVC status can be used by a customer edge (CE) device to find an alternative path in to the service provider network or in some cases to fall back to a backup path over Ethernet or another alternative service such as ATM.

Ethernet LMI Overview

Ethernet Local Management Interface (LMI) is an Ethernet Operation, Administration, and Management (OAM) protocol between a customer edge (CE) device and a provider edge (PE) device. Ethernet LMI provides CE devices with the status of Ethernet virtual circuits (EVCs) for large Ethernet metropolitan-area networks (MANs) and WANs and provides information that enables CE devices to autoconfigure. Specifically, Ethernet LMI runs on the PE-CE User-Network Interface (UNI) link and notifies a CE device of the operating state of an EVC and the time when an EVC is added or deleted. Ethernet LMI also communicates the attributes of an EVC.

Ethernet LMI interoperates with Ethernet Connectivity Fault Management (CFM), an OAM protocol that runs within the provider network to collect OAM status. Ethernet CFM runs at the provider maintenance level (user provider edge [UPE] to UPE at the UNI). Ethernet LMI relies on the OAM Ethernet Infrastructure (EI) to interwork with CFM to learn the end-to-end status of EVCs across CFM domains.

Ethernet LMI is disabled globally by default. When Ethernet LMI is enabled globally, all interfaces are automatically enabled. Ethernet LMI can also be enabled or disabled at the interface to override the global configuration. The last Ethernet LMI command issued is the command that has precedence. No EVCs, Ethernet service instances, or UNIs are defined, and the UNI bundling service is bundling with multiplexing.

Ethernet CFM Overview

Ethernet Connectivity Fault Management (CFM) is an end-to-end per-service-instance (per VLAN) Ethernet layer Operation, Administration, and Management (OAM) protocol that includes proactive connectivity monitoring, fault verification, and fault isolation. End-to-end CFM can be from provider edge (PE) device to PE device or from customer edge (CE) device to CE device. For more information about Ethernet CFM, see "Configuring Ethernet Connectivity Fault Management in a Service Provider Network" in the *Carrier Ethernet Configuration Guide*.

OAM Manager Overview

The OAM manager is an infrastructure element that streamlines interaction between Operation, Administration, and Management (OAM) protocols. The OAM manager requires two interworking OAM protocols, Ethernet Connectivity Fault Management (CFM) and Ethernet Local Management Interface (LMI). No interactions are required between Ethernet LMI and the OAM manager on the customer edge (CE) side. On the User Provider-Edge (UPE) side, the OAM manager defines an abstraction layer that relays data collected from Ethernet CFM to the Ethernet LMI device.

Ethernet LMI and the OAM manager interaction is unidirectional, from the OAM manager to Ethernet LMI on the UPE side of the device. An information exchange results from an Ethernet LMI request or is triggered by the OAM manager when it receives notification from the OAM protocol that the number of UNIs has changed. A change in the number of UNIs may cause a change in Ethernet virtual circuit (EVC) status.

The OAM manager calculates EVC status given the number of active user network interfaces (UNIs) and the total number of associated UNIs. You must configure CFM to notify the OAM manager of all changes to the number of active UNIs or to the remote UNI ID for a given service provider VLAN (S-VLAN) domain.

The information exchanged is as follows:

- EVC name and availability status (active, inactive, partially active, or not defined)
- Remote UNI name and status (up, disconnected, administratively down, excessive frame check sequence [FCS] failures, or not reachable)
- Remote UNI counts (the total number of expected UNIs and the number of active UNIs)

Benefits of Ethernet LMI at a Provider Edge

- Communication of end-to-end status of the Ethernet virtual circuit (EVC) to the customer edge (CE) device
- Communication of EVC and user network interface (UNI) attributes to a CE device
- Competitive advantage for service providers

How to Configure Ethernet Local Management Interface at a Provider Edge

Configuring Ethernet LMI Interaction with CFM

For Ethernet Local Management Interface (LMI) to function with Connectivity Fault Management (CFM), you must configure Ethernet virtual circuits (EVCs), Ethernet service instances including untagged Ethernet flow points (EFPs), and Ethernet LMI customer VLAN mapping. Most of the configuration occurs on the provider edge (PE) device on the interfaces connected to the customer edge (CE) device. On the CE device, you need only enable Ethernet LMI on the connecting interface. Also, you must configure operations, administration, and management (OAM) parameters; for example, EVC definitions on PE devices on both sides of a metro network.

CFM and OAM interworking requires an inward facing Maintenance Entity Group End Point (MEP).

Configuring the OAM Manager

Note If you configure, change, or remove a user network interface (UNI) service type, Ethernet virtual circuit (EVC), Ethernet service instance, or customer edge (CE)-VLAN configuration, all configurations are checked to ensure that the configurations match (UNI service type with EVC or Ethernet service instance and CE-VLAN configuration). The configuration is rejected if the configurations do not match.

Perform this task to configure the OAM manager on a provider edge (PE) device.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ethernet cfm domain** *domain-name* **level** *level-id*
4. **service** *csi-id* **evc** *evc-name* **vlan** *vlan-id*
5. **continuity-check**
6. **continuity-check interval** *time*
7. **exit**
8. **exit**
9. **ethernet evc** *evc-id*
10. **oam protocol** {**cfm domain** *domain-name* | **ldp**}
11. **uni count** *value* [**multipoint**]
12. **exit**
13. Repeat Steps 3 through 12 to define other CFM domains that you want OAM manager to monitor.
14. **interface** *type number*
15. **service instance** *id* **ethernet** [*evc-id*]
16. **ethernet lmi ce-vlan map** {*vlan-id* [**untagged**] | **any** | **default** | **untagged**}
17. **ethernet lmi interface**

18. **encapsulation dot1q** *vlan-id*
19. **bridge-domain** *domain-number*
20. **cfm mep domain** *domain-name* **mpid** *mpid-id*
21. **exit**
22. **service instance** *service-instance-id* **ethernet**
23. **encapsulation untagged**
24. **l2protocol peer**
25. **bridge-domain** *bridge-domain-number*
26. **exit**
27. **ethernet uni** [**bundle** [**all-to-one**] | **id** *uni-id* | **multiplex**]
28. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ethernet cfm domain** *domain-name* **level** *level-id* **Example:** `Device(config)# ethernet cfm domain cstmr1 level 3`	Defines a Connectivity Fault Management (CFM) domain, sets the domain leve,l and enters Ethernet CFM configuration mode.
Step 4	**service** *csi-id* **evc** *evc-name* **vlan** *vlan-id* **Example:** `Device(config-ecfm)# service csi2 evc evc_1 vlan 10`	Defines a universally unique customer service instance (CSI) and VLAN ID within the maintenance domain, and enters Ethernet CFM service configuration mode.
Step 5	**continuity-check** **Example:** `Device(config-ecfm-srv)# continuity-check`	Enables the transmission of continuity check messages (CCMs).
Step 6	**continuity-check interval** *time* **Example:** `Device(config-ecfm-srv)# continuity-check interval 1s/10s/1m/10m`	Enables the transmission of continuity check messages (CCMs) at specific intervals.

	Command or Action	Purpose
Step 7	**exit** **Example:** `Device(config-ecfm-srv)# exit`	Returns to Ethernet CFM configuration mode.
Step 8	**exit** **Example:** `Device(config-ecfm)# exit`	Returns to global configuration mode.
Step 9	**ethernet evc** *evc-id* **Example:** `Device(config)# ethernet evc 50`	Defines an EVC and enters EVC configuration mode.

Step 10	**oam protocol** {**cfm domain** *domain-name* | **ldp**}
	Example:
	`Device(config-evc)# oam protocol cfm domain cstmr1`
		Configures the Ethernet virtual circuit (EVC) operations, administration, and management (OAM) protocol as CFM for the CFM domain maintenance level as configured in Steps 3 and 4.
		Note If the CFM domain does not exist, this command is rejected, and an error message is displayed.

Step 11	**uni count** *value* [**multipoint**] **Example:** `Device(config-evc)# uni count 3`	(Optional) Sets the User Network Interface (UNI) count for the EVC. • If this command is not issued, the service defaults to a point-to-point service. If a value of 2 is entered, point-to-multipoint service becomes an option. If a value of 3 or greater is entered, the service is point-to-multipoint. **Note** If you enter a number greater than the number of endpoints, the UNI status is partially active even if all endpoints are up. If you enter a UNI count less than the number of endpoints, status might be active, even if all endpoints are not up.
Step 12	**exit** **Example:** `Device(config-evc)# exit`	Returns to global configuration mode.
Step 13	Repeat Steps 3 through 12 to define other CFM domains that you want OAM manager to monitor. **Example:** `—`	

	Command or Action	Purpose
Step 14	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 1/3/1`	Specifies a physical interface connected to the CE device and enters interface configuration mode.
Step 15	**service instance** *id* **ethernet** [*evc-id*] **Example:** `Device(config-if)# service instance 400 ethernet 50`	Configures an Ethernet service instance on the interface and enters Ethernet service configuration mode. • The Ethernet service instance identifier is a per-interface service identifier and does not map to a VLAN.

Step 16	**ethernet lmi ce-vlan map** {*vlan-id* [**untagged**] | **any** | **default** | **untagged**}
	Example:
	`Device(config-if-srv)# ethernet lmi ce-vlan map 30`
		Configures an Ethernet LMI customer VLAN-to-EVC map for a particular UNI.
		Note To specify both VLAN IDs and untagged VLANs in the map, specify the VLAN IDs first and then specify the **untagged** keyword as follows: **ethernet lmi ce-vlan map 100,200,300,untagged**. Also, if the **untagged** keyword is not specified in the map configuration, the main interface line protocol on the Customer Edge (CE) device will be down.

Step 17	**ethernet lmi interface** **Example:** `Device(config-if-srv)# ethernet lmi interface`	Enables Ethernet local management interface (LMI) on a UNI.
Step 18	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 2`	Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance.
Step 19	**bridge-domain** *domain-number* **Example:** `Device(config-if-srv)# bridge-domain 1`	Binds a service instance to a bridge domain instance.
Step 20	**cfm mep domain** *domain-name* **mpid** *mpid-id* **Example:** `Device(config-if-srv)# cfm mep domain provider mpid 10`	Configures a maintenance endpoint (MEP) for a domain.
Step 21	**exit** **Example:**	Returns to interface configuration mode.

	Command or Action	Purpose
	`Device(config-if-srv)# exit`	
Step 22	**service instance** *service-instance-id* **ethernet** **Example:** `Device(config-if)# service instance 22 ethernet`	Configures an Ethernet service instance on an interface and enters Ethernet service configuration mode.
Step 23	**encapsulation untagged** **Example:** `Device(config-if-srv)# encapsulation untagged`	Defines the matching criteria to map untagged ingress Ethernet frames on an interface to the appropriate service instance.
Step 24	**l2protocol peer** **Example:** `Device(config-if-srv)# l2protocol peer`	Configures transparent Layer 2 protocol peering on the interface.
Step 25	**bridge-domain** *bridge-domain-number* **Example:** `Device(config-if-srv)# bridge-domain 1`	Binds a service instance to a bridge domain instance.
Step 26	**exit** **Example:** `Device(config-if)# exit`	Returns to interface configuration mode.
Step 27	**ethernet uni** [**bundle** [**all-to-one**] | **id** *uni-id* | **multiplex**] **Example:** `Device(config-if)# ethernet uni bundle`	Sets UNI bundling attributes.

Step 28	**end**
	Example:
	`Device(config-if)# end`
		Returns to privileged EXEC mode.

Enabling Ethernet LMI

The order in which the global and interface configuration commands are issued determines the configuration. The last command that is issued has precedence.

Perform this task to enable Ethernet Local Management Interface (LMI) on a device or on an interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **interface** *type number*
4. **ethernet lmi interface**
5. **ethernet lmi** {**n393** *value* | **t392** *value*}
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface ethernet 1/3`	Defines an interface to configure as an Ethernet LMI interface and enters interface configuration mode.
Step 4	**ethernet lmi interface** **Example:** `Device(config-if)# ethernet lmi interface`	Configures Ethernet LMI on the interface. • When Ethernet LMI is enabled globally, it is enabled on all interfaces unless you disable it on specific interfaces. If Ethernet LMI is disabled globally, you can use this command to enable it on specified interfaces.

Step 5	**ethernet lmi** {**n393** *value* | **t392** *value*}
	Example:
	`Device(config-if)# ethernet lmi n393 10`
		Configures Ethernet LMI parameters for the UNI.

Step 6	**end**
	Example:
	`Device(config-if)# end`
		Returns to privileged EXEC mode.

Displaying Ethernet LMI and OAM Manager Information

Perform this task to display Ethernet Local Management Interface (LMI) or Operation, Administration, and Management (OAM) manager information. After step 1, all the steps are optional and can be performed in any order.

SUMMARY STEPS

1. **enable**
2. **show ethernet lmi** {{**evc** [**detail** *evc-id* [**interface** *type number*] | **map interface** *type number*]} | {**parameters** | **statistics**} **interface** *type number* | **uni map** [**interface** *type number*]}
3. **show ethernet service evc** [**detail** | **id** *evc-id* [**detail**] | **interface** *type number* [**detail**]]
4. **show ethernet service instance** [**detail** | **id** *id* | **interface** *type number* | **policy-map** | **stats**]
5. **show ethernet service interface** [*type number*] [**detail**]

DETAILED STEPS

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

Step 2	**show ethernet lmi** {{**evc** [**detail** *evc-id* [**interface** *type number*] | **map interface** *type number*]} | {**parameters** | **statistics**} **interface** *type number* | **uni map** [**interface** *type number*]}
	Example:
	`Device# show ethernet lmi evc`
		Displays information that was sent to the customer edge (CE).

Step 3	**show ethernet service evc** [**detail** | **id** *evc-id* [**detail**] | **interface** *type number* [**detail**]] **Example:** `Device# show ethernet service evc`	Displays information about all Ethernet virtual circuits (EVCs) or about a specified EVC.
Step 4	**show ethernet service instance** [**detail** | **id** *id* | **interface** *type number* | **policy-map** | **stats**] **Example:** `Device# show ethernet service instance detail`	Displays information about customer service instances.
Step 5	**show ethernet service interface** [*type number*] [**detail**] **Example:** `Device# show ethernet service interface ethernet 1/3 detail`	Displays interface-only information about Ethernet customer service instances for all interfaces or for a specified interface.

Examples

The following example shows sample output from the **show ethernet lmi** command using the **evc** keyword:

```
Device# show ethernet lmi evc

St  EVC Id                                                       Port
--- ------------------------------------------------------------ --------------
A  EVC_MP2MP_101                                                Gi0/1
A  EVC_P2P_110                                                  Gi0/1
```

The following example is sample output from the **show ethernet service evc** command:

```
Device# show ethernet service evc

Identifier                     Type  Act-UNI-cnt Status
50                             MP-MP     0       NotDefined
```

The following is sample output from the **show ethernet service interface** command using the **detail** keyword:

```
Device# show ethernet service interface gigabitethernet 1/3/1 detail

Interface: Gigabitethernet 1/3/1
ID: uni2
CE-VLANS: 30
EVC Map Type: Bundling
Associated EVCs:
    EVC-ID                         CE-VLAN
    50                             30
Associated Service Instances:
    Service-Instance-ID CE-VLAN
    400                 30
```

The following is sample output from the **show ethernet service instance** command using the **detail** keyword:

```
Device# show ethernet service instance detail

Service Instance ID: 400
Associated Interface: GigabitEthernet1/3/1
Associated EVC: 50
CE-Vlans: 30
State: AdminDown
EFP Statistics:
   Pkts In   Bytes In   Pkts Out  Bytes Out
         0          0          0          0
```

Configuration Examples for Ethernet Local Management Interface at a Provider Edge

Example: Ethernet OAM Manager on a PE Device Configuration

This example shows a sample configuration of Operation, Administration, and Management (OAM) manager, Connectivity Fault Management (CFM), and Ethernet Local Management Interface (LMI) on a provider edge (PE) device. In this example, a bridge domain is specified.

```
Device> enable
Device# configure terminal
Device(config)# ethernet cfm global
Device(config)# ethernet cfm domain provider level 4
Device(config-ecfm)# service customer_1 evc test1 vlan 10
Device(config-ecfm-srv)# continuity-check
Device(config-ecfm-srv)# continuity-check interval 1s/10s/1m/10m
Device(config-ecfm-srv)# exit
Device(config-ecfm)# exit
Device(config)# ethernet evc test1
Device(config-evc)# uni count 3
Device(config-evc)# oam protocol cfm domain provider
Device(config-evc)# exit
Device(config)#  interface gigabitEthernet 0/5/1
Device(config-if)# ethernet lmi interface
Device(config-if)# ethernet uni id CISCO
Device(config-if)# service instance 1 ethernet
Device(config-if-srv)# encapsulation untagged
Device(config-if-srv)# l2protocol peer
Device(config-if-srv)# bridge-domain 1
Device(config-if-srv)# exit
Device(config-if)# service instance 2 ethernet1
Device(config-if-srv)# ethernet lmi ce-vlan map 101
Device(config-if-srv)# encapsulation dot1q 2
Device(config-if-srv)# bridge-domain 2
Device(config-if-srv)# cfm mep domain provider mpid 10
Device(config-if-srv-ecfm-mep)# end
```

This example shows a configuration of OAM manager, CFM, and Ethernet LMI over an Xconnect configuration:

```
Device> enable
Device# configure terminal
Device(config)# ethernet cfm global
Device(config)# ethernet cfm domain provider level 4
Device(config-ecfm)# service customer_1 evc test1
Device(config-ecfm-srv)# continuity-check
Device(config-ecfm-srv)# continuity-check interval 1s,10s,1m,10m
Device(config-ecfm-srv)# exit
Device(config-ecfm)# exit
Device(config)# ethernet evc test1
Device(config-evc)# oam protocol cfm domain provider
Device(config-evc)# exit
Device(config)#  interface gigabitEthernet 0/5/1
Device(config-if)# ethernet lmi interface
Device(config-if)# ethernet uni id CISCO
Device(config-if)# service instance 1 ethernet
Device(config-if-srv)# encapsulation untagged
Device(config-if-srv)# l2protocol peer
Device(config-if-srv)# bridge-domain 1
Device(config-if-srv)# exit
Device(config-if)# service instance 2 ethernet
Device(config-if-srv)# ethernet lmi ce-vlan map 101
Device(config-if-srv)# encapsulation dot1q 2
Device(config-if-srv)# xconnect 10.1.1.1 100 encapsulation mpls
Device(cfg-if-ether-vc-xconn)# exit
Device(config-if-srv)# cfm mep domain provider mpid 10
Device(config-if-srv-ecfm-mep)# end
```

Example: Ethernet LMI on a CE Device Configuration

This example shows how to configure Ethernet Local Management Interface (LMI) globally on a customer edge (CE) device:

```
Device# configure terminal
Device(config)# ethernet lmi global
Device(config)# ethernet lmi ce
Device(config)# exit
```

CHAPTER 11

ITU-T Y.1731 Performance Monitoring in a Service Provider Network

ITU-T Y.1731 performance monitoring provides standard-based Ethernet performance monitoring that encompasses the measurement of Ethernet frame delay, frame-delay variation, and throughput as outlined in the ITU-T Y.1731 specification and interpreted by the Metro Ethernet Forum (MEF). Service providers offer service level agreements (SLAs) that describe the level of performance customers can expect for services. This document describes the Ethernet performance management aspect of SLAs.

Prerequisites for ITU-T Y.1731 Performance Monitoring in a Service Provider Network

- IEEE-compliant connectivity fault management (CFM) must be configured and enabled for Y.1731 performance monitoring to function.

Note Y1731 is supported over Port Channel interfaces.

Restrictions for ITU-T Y.1731 Performance Monitoring in a Service Provider Network

- A frame-delay measurement message (DMM) with CFMoXconnect on a Cisco ASR 920 router works only if the **control-word** command is enabled. If the remote end of the pseudowire does not enable the **control-word** command by default, enable the command to get the DMM time stamp working. Cisco

ASR 9000 Series Routers and ALU 7450 are examples of peer platforms where the **control-word** command needs to be enabled.

- When the core network has multiple paths, the Tx and Rx DMM/DMR packets can be sent and received on different ports. If the ports belong to a different interface module (IM), time stamping can be out of sync and in certain cases the Rx value can be lower than the Tx value. This value will be displayed as 0 in the raw database output. As a workaround, configure Precision Time Protocol (PTP) between the two connectivity fault management (CFM) endpoint routers.
- Y1731 is supported with the **rewrite** command configuration on Ethernet Flow Points (EFPs) throughout the Layer 2 circuit. However, the configuration may be in such a way that the Y1731 PDUs may be transmitted without a tag. This results in the other end of the Layer 2 circuit not being able to ascertain the CoS value which determines the SLA session to which the PDUs belong. Therefore, the **rewrite** command configuration is not supported
- Y.1731 Performance Monitoring (PM) is not supported on MEPs that are configured on TEFP.
- Y.1731 Performance Monitoring (PM) is not supported on MEPs that are configured on ports.

Note In ITU-T Y1731, 1DM measurement should mandate only PTP to have clock sync between sender & receiver.

Information About ITU-T Y.1731 Performance Monitoring in a Service Provider Network

Frame Delay and Frame-Delay Variation

The Frame Delay parameter can be used for on-demand OAM measurements of frame delay and frame-delay variation. When a maintenance end point (MEP) is enabled to generate frames with frame-delay measurement (ETH-DM) information, it periodically sends frames with ETH-DM information to its peer MEP in the same maintenance entity. Peer MEPs perform frame-delay and frame-delay variation measurements through this periodic exchange during the diagnostic interval.

An MEP requires the following specific configuration information to support ETH-DM:

- MEG level—MEG level at which the MEP exists
- Priority
- Drop eligibility—marked drop ineligible
- Transmission rate
- Total interval of ETH-DM
- MEF10 frame-delay variation algorithm

A MEP transmits frames with ETH-DM information using the TxTimeStampf information element. TxTimeStampf is the time stamp for when the ETH-DM frame was sent. A receiving MEP can compare the TxTimeStampf value with the RxTimef value, which is the time the ETH-DM frame was received, and calculate one-way delay using the formula *frame delay = RxTimef – TxTimeStampf*.

One-way frame-delay measurement (1DM) requires that clocks at both the transmitting MEP and the receiving MEPs are synchronized. Measuring frame-delay variation does not require clock synchronization and the variation can be measured using 1DM or a frame-delay measurement message (DMM) and a frame-delay measurement reply (DMR) frame combination.

If it is not practical to have clocks synchronized, only two-way frame-delay measurements can be made. In this case, the MEP transmits a frame containing ETH-DM request information and the TxTimeStampf element, and the receiving MEP responds with a frame containing ETH-DM reply information and the TxTimeStampf value copied from the ETH-DM request information.

Two-way frame delay is calculated as *(RxTimeb–TxTimeStampf)–(TxTimeStampb–RxTimeStampf)*, where RxTimeb is the time that the frame with ETH-DM reply information was received. Two-way frame delay and variation can be measured using only DMM and DMR frames.

To allow more precise two-way frame-delay measurement, the MEP replying to a frame with ETH-DM request information can also include two additional time stamps in the ETH-DM reply information:

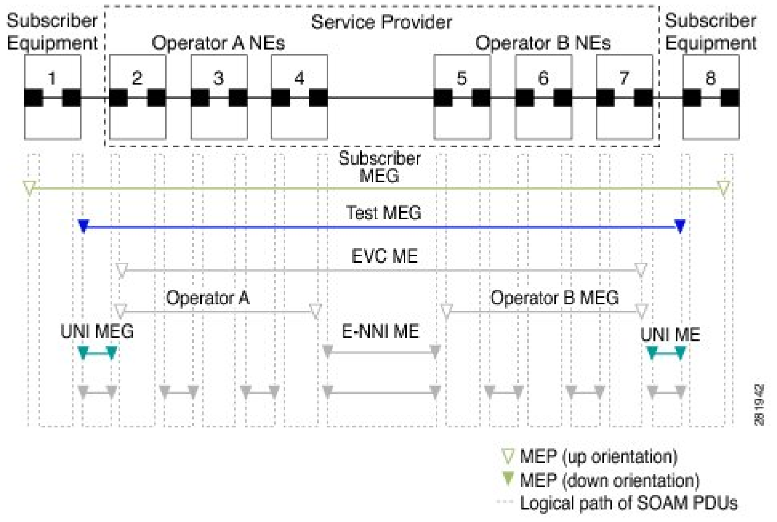
- RxTimeStampf—Time stamp of the time at which the frame with ETH-DM request information was received.
- TxTimeStampb—Time stamp of the time at which the transmitting frame with ETH-DM reply information was sent.
- The timestamping happens at the hardware level for DMM operations.

Note The frame-loss, frame-delay, and frame-delay variation measurement processes are terminated when faults related to continuity and availability occur or when known network topology changes occur.

An MIP is transparent to the frames with ETH-DM information; therefore, an MIP does not require information to support the ETH-DM function.

The figure below shows a functional overview of a typical network in which Y.1731 performance monitoring is used.

Figure 5: Y.1731 Performance Monitoring



Benefits of ITU-T Y.1731 Performance Monitoring

Combined with IEEE-compliant connectivity fault management (CFM), Y.1731 performance monitoring provides a comprehensive fault management and performance monitoring solution for service providers. This comprehensive solution in turn lessens service providers' operating expenses, improves their service-level agreements (SLAs), and simplifies their operations.

How to Configure ITU-T Y.1731 Performance Monitoring in a Service Provider Network

Configuring Performance Monitoring Parameters

The following new commands were introduced that can be used to configure and display performance monitoring parameters: **debug ethernet cfm pm**, **monitor loss counters**, and **show ethernet cfm pm**.

For more information about CFM and Y.1731 performance monitoring commands, see the *Cisco IOS Carrier Ethernet Command Reference*. For more information about debug commands, see the *Cisco IOS Debug Command Reference*.

Configuration Examples for Configuring ITU-T Y.1731 Performance Monitoring Functions

Example: Configuring Performance Monitoring

For Y.1731 performance monitoring configuration examples, see Configuring IP SLAs Metro-Ethernet 3.0 (ITU-T Y.1731) Operations. For information on Y.1731 On-Demand and Concurrent Operations see, IPSLA Y1731 On-Demand and Concurrent Operations.

CHAPTER 12

Using Link Layer Discovery Protocol in Multivendor Networks

Link Layer Discovery Protocol (LLDP), standardized by the IEEE as part of 802.1ab, enables standardized discovery of nodes, which in turn facilitates future applications of standard management tools such as Simple Network Management Protocol (SNMP) in multivendor networks. Using standard management tools makes physical topology information available and helps network administrators detect and correct network malfunctions and inconsistencies in configuration.

Media Endpoint Discovery (MED) is an LLDP enhancement that was formalized by the Telecommunications Industry Association (TIA) for voice over IP (VoIP) applications.

The Cisco implementation of LLDP is based on the IEEE 802.1ab standard. This document describes LLDP and LLDP-MED and how they are supported in Cisco software.

Prerequisites for Using Link Layer Discovery Protocol in Multivendor Networks

- Type-Length-Value (TLV) types 0 through 127
- To support LLDP-MED, the following organizationally specific TLVs must be implemented:
  - Extended Power-via-Media Dependent Interface (MDI)
  - Inventory
  - LLDP-MED Capabilities
  - MAC/PHY Configuration Status
  - Network Policy
  - Port VLAN ID

Restrictions for Using Link Layer Discovery Protocol in Multivendor Networks

- Use of LLDP is limited to 802.1 media types such as Ethernet, Token Ring, and Fiber Distributed Data Interface (FDDI) networks.
- The maximum number of neighbor entries per chassis is limited on MED-capable network connectivity devices.

Information About Using Link Layer Discovery Protocol in Multivendor Networks

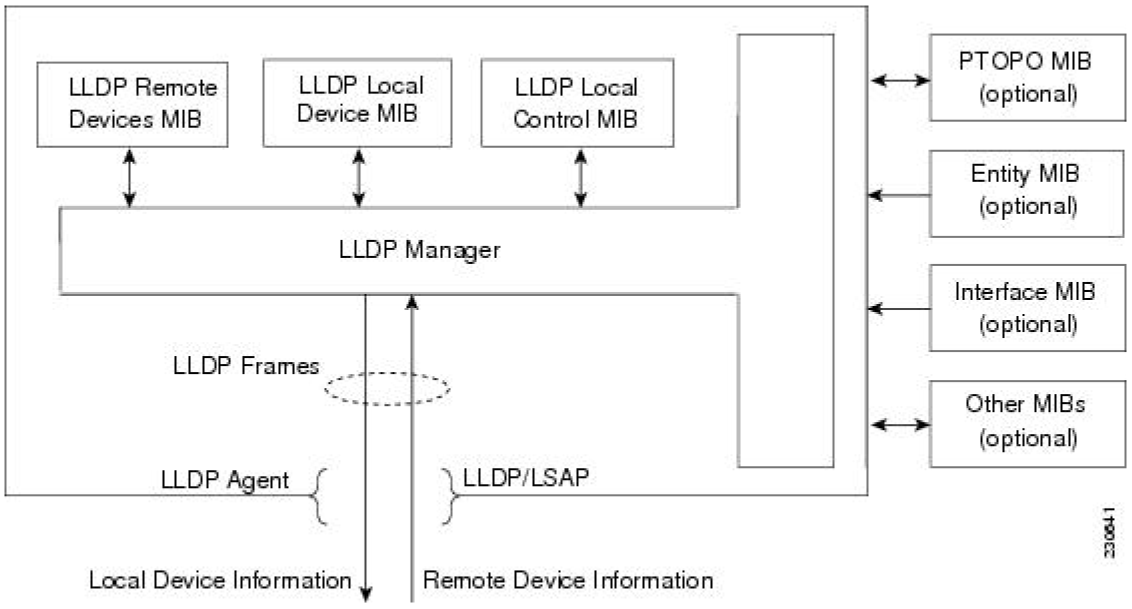
IEEE 802.1ab LLDP

IEEE 802.1ab Link Layer Discovery Protocol (LLDP) is an optional link layer protocol for network topology discovery in multivendor networks. Discovery information includes device identifiers, port identifiers, versions, and other details. As a protocol that aids network management, LLDP provides accurate network mapping, inventory data, and network troubleshooting information.

LLDP is unidirectional, operating only in an advertising mode. LLDP does not solicit information or monitor state changes between LLDP nodes. LLDP periodically sends advertisements to a constrained multicast address. Devices supporting LLDP can send information about themselves while they receive and record information about their neighbors. Additionally, devices can choose to turn off the send or receive functions independently. Advertisements are sent out and received on every active and enabled interface, allowing any device in a network to learn about all devices to which it is connected. Applications that use this information include network topology discovery, inventory management, emergency services, VLAN assignment, and inline power supply.

Note LLDP and Cisco Discovery Protocol can operate on the same interface.

The figure below shows a high-level view of LLDP operating in a network node.



When you configure LLDP or Cisco Discovery Protocol location information on a per-port basis, remote devices can send Cisco medianet location information to the switch. For more information, see the *Using Cisco Discovery Protocol module.*

LLDP-MED

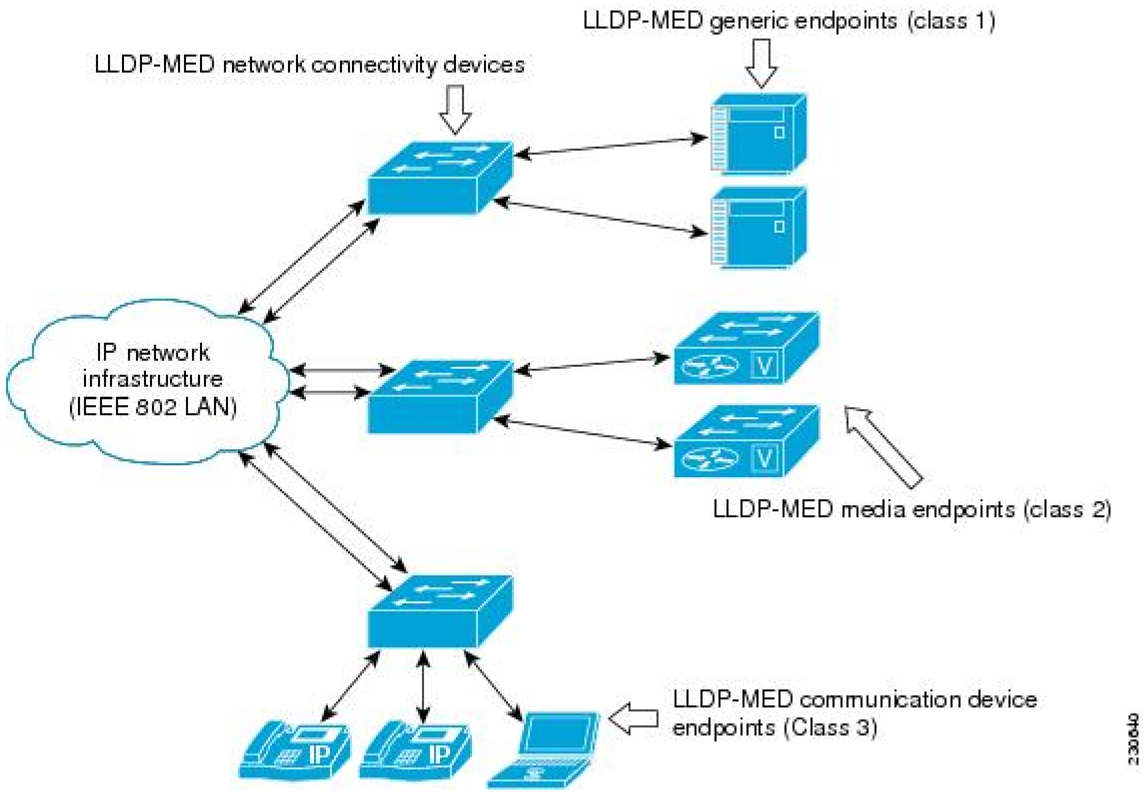
LLDP-MED operates between several classes of network equipment such as IP phones, conference bridges, and network connectivity devices such as routers and switches. By default, a network connectivity device sends out only LLDP packets until it receives LLDP-MED packets from an endpoint device. The network device then sends out LLDP-MED packets until the remote device to which it is connected ceases to be LLDP-MED capable.

Classes of Endpoints

LLDP-MED network connectivity devices provide IEEE 802 network access to LLDP-MED endpoints. LLDP-MED supports the following three classes of endpoints:

- Generic (class 1)—Basic participant endpoints; for example, IP communications controllers.
- Media (class 2)—Endpoints that support media streams; for example, media gateways and conference bridges.
- Communication Device (class 3)—Endpoints that support IP communications end users; for example, IP phones and Softphone.

The figure below shows an LLDP-MED-enabled LAN.


LLDP-MED generic endpoints (class 1)
LLDP-MED network connectivity devices
IP network infrastructure (IEEE 802 LAN)
LLDP-MED media endpoints (class 2)
LLDP-MED communication device endpoints (Class 3)
230640

Types of Discovery Supported

LLDP-MED provides support to discover the following types of information, which are crucial to efficient operation and management of endpoint devices and the network devices supporting them:

- **Capabilities** —Endpoints determine the types of capabilities that a connected device supports and which ones are enabled.
- **Inventory** —LLDP-MED support exchange of hardware, software, and firmware versions, among other inventory details.
- **LAN speed and duplex** —Devices discover mismatches in speed and duplex settings.
- **Location identification** —An endpoint, particularly a telephone, learns its location from a network device. This location information may be used for location-based applications on the telephone and is important when emergency calls are placed.
- **Network policy** —Network connectivity devices notify telephones about the VLANs they should use.
- **Power** —Network connectivity devices and endpoints exchange power information. LLDP-MED provides information about how much power a device needs and how a device is powered. LLDP-MED also determines the priority of the device for receiving power.

Benefits of LLDP-MED

- Follows an open standard
- Supports E-911 emergency service, which is aided by location management

- Provides fast start capability
- Supports interoperability between multivendor devices
- Supports inventory management (location, version, etc.)
- Provides MIB support
- Supports plug and play installation
- Provides several troubleshooting (duplex, speed, network policy) mechanisms

TLV Elements

Link Layer Discovery Protocol (LLDP) and LLDP-Media Endpoint Discovery (MED) use Type-Length-Values (TLVs) to exchange information between network and endpoint devices. TLV elements are embedded in communications protocol advertisements and used for encoding optional information. The size of the type and length fields is fixed at 2 bytes. The size of the value field is variable. The type is a numeric code that indicates the type of field that this part of the message represents, and the length is the size of the value field, in bytes. The value field contains the data for this part of the message.

LLDP-MED supports the following TLVs:

- LLDP-MED capabilities TLV—Allows LLDP-MED endpoints to determine the capabilities that the connected device supports and has enabled.
- Network policy TLV—Allows both network connectivity devices and endpoints to advertise VLAN configurations and associated Layer 2 and Layer 3 attributes for the specific application on that port. For example, the switch can notify a phone of the VLAN number that it should use. The phone can connect to any switch, obtain its VLAN number, and then start communicating with the call control.

By defining a network-policy profile TLV, you can create a profile for voice and voice signalling by specifying the values for VLAN, class of service (CoS), differentiated services code point (DSCP), and tagging mode. These profile attributes are then maintained centrally on the switch and propagated to the phone.

- Power management TLV—Enables advanced power management between LLDP-MED endpoint and network connectivity devices. Allows switches and phones to convey power information, such as how the device is powered, power priority, and how much power the device needs. Supports advertisement of fractional wattage power requirements, endpoint power priority, and endpoint and network connectivity-device power status but does not provide for power negotiation between the endpoint and the network connectivity devices. When LLDP is enabled and power is applied to a port, the power TLV determines the actual power requirement of the endpoint device so that the system power budget can be adjusted accordingly. The switch processes the requests and either grants or denies power based on the current power budget. If the request is granted, the switch updates the power budget. If the request is denied, the switch turns off power to the port, generates a syslog message, and updates the power budget. If LLDP-MED is disabled or if the endpoint does not support the LLDP-MED power TLV, the initial allocation value is used throughout the duration of the connection.

Note A system power budget is the default power allocated to a device based on its device class. However, the total power that can be sourced from a switch is finite, and there will be some power budgeting done by the power module based on the number of ports already being served, total power that can be served, and how much new ports are requesting.

- Inventory management TLV—Allows an endpoint to send detailed inventory information about itself to the switch, including information hardware revision, firmware version, software version, serial number, manufacturer name, model name, and asset ID TLV.
- Location TLV—Provides location information from the switch to the endpoint device. The location TLV can send this information:
    - Civic location information—Provides the civic address information and postal information. Examples of civic location information are street address, road name, and postal community name information.
    - ELIN location information—Provides the location information of a caller. The location is determined by the Emergency location identifier number (ELIN), which is a phone number that routes an emergency call to the local public safety answering point (PSAP) and which the PSAP can use to call back the emergency caller.

Benefits of LLDP

- Follows IEEE 802.1ab standard.
- Enables interoperability among multivendor devices.
- Facilitates troubleshooting of enterprise networks and uses standard network management tools.
- Provides extension for applications such as VoIP.

How to Configure Link Layer Discovery Protocol in Multivendor Networks

Enabling and Disabling LLDP Globally

LLDP is disabled globally by default. This section describes the tasks for enabling and disabling LLDP globally.

Enabling LLDP Globally

Perform this task to enable LLDP globally.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **lldp run**
4. **end**

DETAILED STEPS

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

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**lldp run** **Example:** `Device(config)# lldp run`	Enables LLDP globally. **Note** To disable LLDP globally, use the **no lldp run** command.

Step 4	**end**
	Example:
	`Device(config)# end`
		Returns to privileged EXEC mode.

Disabling LLDP Globally

Perform this task to disable LLDP globally.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no lldp run**
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no lldp run** **Example:** `Device(config)# no lldp run`	Disables LLDP globally.

	Command or Action	Purpose

Step 4	**end**
	Example:
	`Device(config)# end`
		Returns to privileged EXEC mode.

Disabling and Enabling LLDP on a Supported Interface

LLDP is enabled by default on all supported interfaces. This section describes the tasks for disabling and enabling LLDP on a supported interface.

Disabling LLDP on a Supported Interface

Perform this task to disable LLDP on a supported interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no lldp** {**med-tlv-select** *tlv* | **receive** | **transmit**}
5. **end**

DETAILED STEPS

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

Step 4	**no lldp** {**med-tlv-select** *tlv* | **receive** | **transmit**}
	Example:
	`Device(config-if)# no lldp receive`
		Disables an LLDP-MED TLV or LLDP packet reception on a supported interface.
		Note To enable LLDP on a Supported Interface, use the **lldp** {**med-tlv-select** *tlv* | **receive** | **transmit** command.

	Command or Action	Purpose

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

Enabling LLDP on a Supported Interface

LLDP information can be transmitted and received only on an interface where LLDP is configured and enabled. Perform this task to enable LLDP.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **lldp** {**med-tlv-select** *tlv* | **receive** | **transmit**}
5. **end**

DETAILED STEPS

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

Step 4	**lldp** {**med-tlv-select** *tlv* | **receive** | **transmit**}
	Example:
	`Device(config-if)# lldp transmit`
		Enables an LLDP-MED TLV or LLDP packet transmission on a supported interface.

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

Setting LLDP Packet Hold Time

Hold time is the duration that a receiving device should maintain LLDP neighbor information before aging it. Perform this task to define a hold time for an LLDP-enabled device.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **lldp holdtime** *seconds*
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**lldp holdtime** *seconds* **Example:** `Device(config)# lldp holdtime 100`	Specifies the hold time.

Step 4	**end**
	Example:
	`Device(config)# end`
		Returns to privileged EXEC mode.

Setting LLDP Packet Frequency

Perform this task to specify an interval at which the Cisco software sends LLDP updates to neighboring devices.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **lldp timer** *rate*
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**lldp timer** *rate* **Example:** `Device(config)# lldp timer 75`	Specifies the rate at which LLDP packets are sent every second.

Step 4	**end**
	Example:
	`Device(config)# end`
		Returns to privileged EXEC mode.

Monitoring and Maintaining LLDP in Multivendor Networks

Perform this task to monitor and maintain LLDP in multivendor networks. This task is optional, and Steps 2 and 3 can be performed in any sequence.

SUMMARY STEPS

1. **enable**
2. **show lldp** [**entry** {***** | *word*} | **errors** | **interface** [**ethernet** *number*]| **neighbors** [**ethernet** *number*| **detail**]| **traffic**]
3. **clear lldp** {**counters** | **table**}
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show lldp** [**entry** {*****	*word*}	**errors**	**interface** [**ethernet** *number*]| **neighbors** [**ethernet** *number*| **detail**]| **traffic**]	Displays summarized and detailed LLDP information.

	Command or Action	Purpose
	Example: `Device# show lldp entry *`	**Note** When the **show lldp neighbors** command is issued, if the device ID has more than 20 characters, the ID is truncated to 20 characters in command output because of display constraints.

Step 3	**clear lldp** {**counters** | **table**}
	Example:
	`Device# clear lldp counters`
		Resets LLDP traffic counters and tables to zero.

Step 4	**end**
	Example:
	`Device# end`
		Returns to user EXEC mode.

Enabling and Disabling LLDP TLVs

LLDP TLV support is enabled by default if LLDP is enabled globally and locally on a supported interface. Specific TLVs, however, can be enabled and suppressed.

Enabling LLDP TLVs

Perform this task to enable an LLDP TLV on a supported interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **lldp tlv-select** *tlv*
5. **end**

DETAILED STEPS

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

	Command or Action	Purpose
Step 3	**interface** *type number* **Example:** `Device(config)# interface Gigabitethernet 0/1`	Specifies the interface type and number on which to enable LLDP-MED and enters interface configuration mode.
Step 4	**lldp tlv-select** *tlv* **Example:** `Device(config-if)# lldp tlv-select power-management`	Enables a specific LLDP TLV on a supported interface. **Note** To disable LLDP TLVs, use the **no lldp tlv-select** *tlv*

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

Disabling LLDP TLVs

Perform this task to disable an LLDP TLV on a supported interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no lldp tlv-select** *tlv*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface ethernet 0/1`	Specifies the interface type and number on which to disable LLDP-MED and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**no lldp tlv-select** *tlv* **Example:** `Device(config-if)# no lldp tlv-select system-description`	Disables a specific LLDP TLV on a supported interface.

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

Enabling and Disabling LLDP-MED TLVs

LLDP-MED TLV support is enabled by default if LLDP is enabled globally and locally on a supported interface. Specific TLVs, however, can be enabled and suppressed.

Enabling LLDP-MED TLVs

Perform this task to enable a specific LLDP-MED TLV on a supported interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **lldp med-tlv-select** *tlv*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface Gigabitethernet 0/1`	Specifies the interface type and number on which to enable LLDP-MED and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**lldp med-tlv-select** *tlv* **Example:** `Device(config-if)# lldp med-tlv-select inventory-management`	Enables a specific LLDP-MED TLV on a supported interface. **Note** To disable LLDP-MED TLVs, use the **no lldp med-tlv-select** *tlv* command.

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

Disabling LLDP-MED TLVs

Perform this task to disable a specific LLDP-MED TLV from a supported interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no lldp med-tlv-select** *tlv*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface ethernet 0/1`	Specifies the interface type and number on which to disable LLDP-MED and enters interface configuration mode.
Step 4	**no lldp med-tlv-select** *tlv* **Example:** `Device(config-if)# no lldp med-tlv-select inventory-management`	Disables a specific LLDP-MED TLV from a supported interface.

	Command or Action	Purpose

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

Configuration Examples for Link Layer Discovery Protocol in Multivendor Networks

Example Configuring LLDP on Two Devices

The following example shows how to configure LLDP timer, hold time, and TLVs on two devices in a network. In each case we assume that the Ethernet interfaces being configured are in the UP state.

```
! Configure LLDP on Device 1 with hold time, timer, and TLV options.

Device1> enable
Device1# configure terminal
Device1(config)# lldp run
Device1(config)# lldp holdtime 150
Device1(config)# lldp timer 15
Device1(config)# lldp tlv-select port-vlan
Device1(config)# lldp tlv-select mac-phy-cfg
Device1(config)# interface ethernet 0/0
Device1(config-if)# end
00:08:32: %SYS-5-CONFIG_I: Configured from console by console
! Show the updated running configuration. LLDP is enabled with hold time, timer, and TLV
options configured.

Device1# show running-config

Building configuration...
Current configuration : 1397 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Device1
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
clock timezone PST -8
ip subnet-zero
!
!
lldp timer 15
lldp holdtime 150
!
```

```
! Configure LLDP on Device 2 with hold time, timer, and TLV options.

Device2> enable
Device2# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device2(config)# lldp run
Device2(config)# lldp holdtime 150
Device2(config)# lldp timer 15
Device2(config)# lldp tlv-select port-vlan
Device2(config)# lldp tlv-select mac-phy-cfg
Device2(config)# interface ethernet 0/0
Device2(config-if)# end
00:08:32: %SYS-5-CONFIG_I: Configured from console by console

! Show the updated running configuration on Device 2. LLDP is enabled with hold time, timer,
 and TLV options configured.

Device2# show running-config
Building configuration...
Current configuration : 1412 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname R2
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
clock timezone PST -8
ip subnet-zero
!
!
lldp timer 15
lldp holdtime 150
!


! After both devices are configured for LLDP, issue the show
 command from each device to view traffic and device information.

Device1# show lldp traffic
LLDP traffic statistics:
    Total frames out: 20
    Total entries aged: 0
    Total frames in: 15
    Total frames received in error: 0
    Total frames discarded: 0
    Total TLVs unrecognized: 0
Device1# show lldp neighbors
Capability codes:
    (R) Router, (B) Bridge, (T) Telephone, (C) DOCSIS Cable Device
    (W) WLAN Access Point, (P) Repeater, (S) Station, (O) Other
Device ID           Local Intf     Hold-time  Capability      Port ID
Device2             Et0/0          150        R               Et0/0
Total entries displayed: 1
Device2# show lldp traffic
LLDP traffic statistics:
    Total frames out: 15
    Total entries aged: 0
    Total frames in: 17
    Total frames received in error: 0
    Total frames discarded: 2
    Total TLVs unrecognized: 0
Device2# show lldp neighbors
Capability codes:
    (R) Router, (B) Bridge, (T) Telephone, (C) DOCSIS Cable Device
    (W) WLAN Access Point, (P) Repeater, (S) Station, (O) Other
Device ID           Local Intf     Hold-time  Capability      Port ID
Device1             Et0/0          150        R               Et0/0
Total entries displayed: 1
```

CHAPTER 13

Configuring Switched Port Analyzer

This document describes how to configure local Switched Port Analyzer (SPAN) and remote SPAN (RSPAN) on the router.

Prerequisites for Configuring Local SPAN and RSPAN

Local SPAN

- Use a network analyzer to monitor interfaces.

RSPAN

- Before configuring RSPAN sessions, you must first configure:

1. Source interface
2. Destination Bridge Domain over VPLS

Restrictions for Local Span and RSPAN

Local Span

- Local SPAN is only supported on physical ports.
- VLAN filtering is not supported.
- SPAN monitoring of port-channel interfaces or port-channel member-links is *not* supported.

- Combined Egress local SPAN bandwidth supported is 1 GB.
- Local SPAN is not supported on logical interfaces such as VLANs or EFPs.
- Up to 14 active local SPAN sessions (ingress and egress) are supported. The router supports up to 14 ingress sessions and up to 12 egress sessions.
- Only one local SPAN destination interface is supported. You *cannot* configure a local SPAN destination interface to receive ingress traffic.
- Outgoing Cisco Discovery Protocol (CDP) and Bridge Protocol Data Unit (BPDU) packets are not replicated.
- When enabled, local SPAN uses any previously entered configuration.
- When you specify source interfaces and do not specify a traffic direction (**Tx**, **Rx**, or **both**), **both** is used by default.
- Local SPAN destinations never participate in any spanning tree instance. Local SPAN includes BPDUs in the monitored traffic, so any BPDUs seen on the local SPAN destination are from the local SPAN source.
- Local SPAN sessions with overlapping sets of local SPAN source interfaces or VLANs are *not* supported.

RSPAN

- RSPAN VLAN/BD is *not* used for data traffic.
- The maximum number of supported RSPAN sessions are 14.
- Only one source port is supported per RSPAN.
- Only port channel RSPAN is supported.
- Per member link RSPAN is not supported.
- Source ranges (VLAN range or port range) is *not* supported.
- VLAN filtering is not supported.
- If two RSPAN configurations sessions are configured on two RSPAN BDs associated to the same Trunk EFP, the traffic from the first session flows to the second session after it is configured.
- RSPAN destination configuration for Layer2 pseudowire is *not* supported.
- If RSPAN BD is associated with a VPLS pseudowire, the traffic flows through the VPLS pseudowire.
- Do not have RSPAN bridge domain as part of RSPAN source interface.
- RSPAN spans the Rx traffic even when the classifying service instance of the receiving port is in admin down state.
- If RSPAN source and destinations are separated by pseudowire, then the RSPAN details must be updated on both RSPAN source switch and destination switch. The pseudowire should also be dedicated for RSPAN traffic.

Note Incomplete configuration of RSPAN / LSPAN will result in traffic drop issues.

Understanding Local SPAN and RSPAN

Information About Local SPAN Session and RSPAN Session

Local SPAN Session

A local Switched Port Analyzer (SPAN) session is an association of a destination interface with a set of source interfaces. You can configure local SPAN sessions to monitor all traffic in a specified direction. Local SPAN sessions allow you to monitor traffic on one or more interfaces and to send either ingress traffic, egress traffic, or both to one destination interface.

Local SPAN sessions do not interfere with the normal operation of the switch. You can enable or disable SPAN sessions with command-line interface (CLI) commands. When enabled, a local SPAN session might become active or inactive based on various events or actions, and this would be indicated by a syslog message. The **show monitor session span session number** command displays the operational status of a SPAN session.

A local SPAN session remains inactive after system power-up until the destination interface is operational.

The following configuration guidelines apply when configuring local SPAN on the router:

- When enabled, local SPAN uses any previously entered configuration.
- Use the **no monitor session** *session number* command with no other parameters to clear the local SPAN session number.

Local SPAN Traffic

Network traffic, including multicast, can be monitored using SPAN. Multicast packet monitoring is enabled by default. In some SPAN configurations, multiple copies of the same source packet are sent to the SPAN destination interface. For example, a bidirectional (both ingress and egress) SPAN session is configured for sources a1 and a2 to a destination interface d1. If a packet enters the switch through a1 and gets switched to a2, both incoming and outgoing packets are sent to destination interface d1; both packets would be the same (unless a Layer-3 rewrite had occurred, in which case the packets would be different).

RSPAN Session

An RSPAN source session is an association of source ports or VLAN across your network with an RSPAN Vlan. The RSPAN VLAN/BD on the router is the destination RSPAN session.

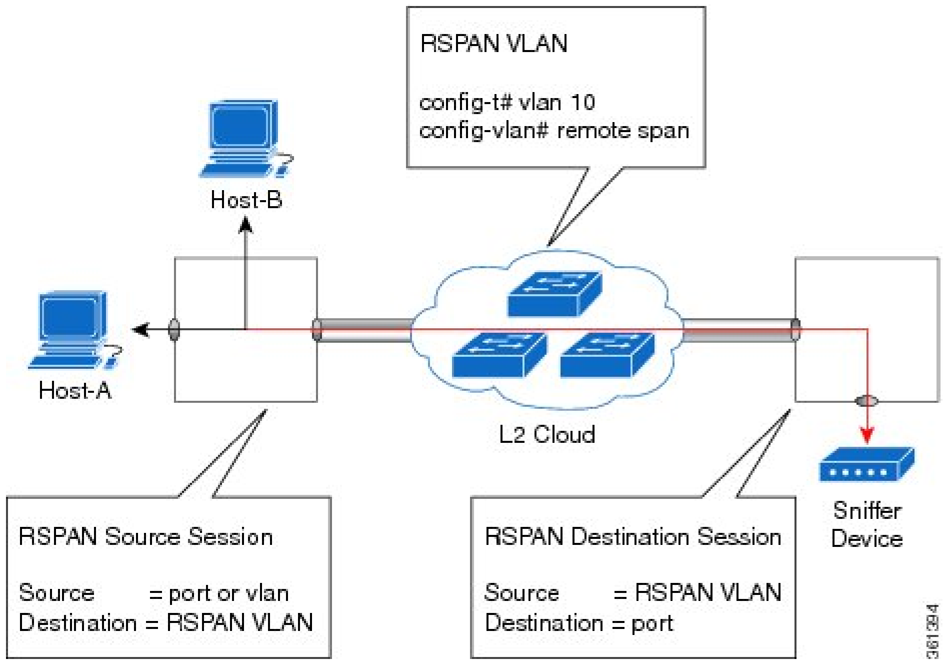
RSPAN Traffic for RSP2 Module

RSPAN supports source ports and source VLANs in the source switch and destination as RSPAN VLAN/BD.

The figure below shows the original traffic from the Host A to Host B via the source ports or VLANs on Host A. The source ports or VLANs of Host A is mirrored to Host B using RSPAN VLAN 10. The traffic for each RSPAN session is carried over a user-specified RSPAN VLAN that is dedicated for that RSPAN session in all participating devices. The traffic from the source ports or VLANs are mirrored into the RSPAN VLAN and forwarded over Trunk or the EVC bridge domain (BD) ports carrying the RSPAN VLAN to a destination session monitoring the RSPAN VLAN.

Each RSPAN source must have either ports or VLANs as RSPAN sources. On RSPAN destination, the RSPAN VLAN is monitored and mirrored to the destination physical port connected to the sniffer device.

Figure 6: RSPAN Traffic



RSPAN allows remote monitoring of traffic where the source and destination switches are connected by L2VPN networks

The RSPAN source is either ports or VLANs as in a traditional RSPAN. However, the SPAN source and destination devices are connected through a L2 pseudowire associated with the RSPAN VLAN over an MPLS/IP network. The L2 pseudowire is dedicated for only RSPAN traffic. The mirrored traffic from the source port or VLAN is carried over the pseudowire associated with the RSPAN VLAN towards the destination side. On the destination side, a port belonging to the RSPAN VLAN or EVC BD is connected to sniffer device.

Destination Interface

A destination interface, also called a monitor interface, is a switched interface to which SPAN or RSPAN sends packets for analysis. You can have only one destination interface for SPAN sessions.

An interface configured as a destination interface cannot be configured as a source interface. Specifying a trunk interface as a SPAN or RSPAN destination interface stops trunking on the interface.

Source Interface

A source interface is an interface monitored for network traffic analysis. An interface configured as a destination interface cannot be configured as a source interface.

Traffic Directions

Ingress SPAN (Rx) copies network traffic received by the source interfaces for analysis at the destination interface. Egress SPAN (Tx) copies network traffic transmitted from the source interfaces to the destination interface. Specifying the configuration option (both) copies network traffic received and transmitted by the source interfaces to the destination interface.

The following table lists the supported traffic types for RSPAN.

Table 5: RSPAN over VPLS Traffic for RSP3 module

Source	Ingress Mirror (Rx)	Egress Mirror (Tx)	Both
CFM	Not Supported	Supported	Not Supported
Layer 2	Supported	Supported	Supported
Layer 3	Incoming Ethernet and VLAN header are stripped off and RSPANed over VPLS	Supported	Not Supported
L2VPN	Not Supported	Supported	Not Supported
L3VPN	Not Supported	Supported	Not Supported
L3VPN over BDI	Not Supported	Supported	Not Supported
MPLS	Incoming Ethernet and VLAN header are stripped off and RSPANed over VPLS	Supported	Not Supported
Routed PW	Not Supported	Supported	Not Supported
VPLS	Not supported for bidirectional traffic	Supported	Not Supported

Table 6: RSPAN Traffic

Source	Ingress Mirror (Rx)	Egress Mirror (Tx)	Both
Layer2 or Layer3	Supported	Supported	Supported
VLAN	Supported	Not supported	Not supported
EFP	Not supported	Not supported	Not supported
Pseudowire	Not supported	Not supported	Not supported

The following table lists the supported **rewrite** traffic for RSPAN on the EFP, Trunk with the associated RSPAN Bridge Domains (BD).

Table 7: Rewrite Traffic for RSPAN BD

Rewrite Operations	Source	EFP/Trunk associated with RSPAN BD
no-rewrite	Pop1, Pop2, Push1	Only Pop1

The following tables lists the format of the spanned packets at the destination port for both Ingress and Egress RSPAN. The tables lists the formats of untagged, single, and double tagged source packets for EFPs under source port configured with **rewrite** operations (no-rewrite, pop1, pop2 and push1).

Table 8: Destination Port Ingress and Egress Spanned Traffic for EVC RSPAN BD

	Ingress Traffic	Egress Traffic
(Untagged Traffic) - Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + packet	RSPAN BD tag + packet
pop1 tag	NA	NA
pop2 tag	NA	NA
push1 tag	NA	NA
(Single Traffic)-Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outer-tag + packet	RSPAN BD tag + source-outer-tag + packet
pop1 tag		
pop2 tag		NA
push1 tag		RSPAN BD tag + source-outer-tag + packet
(Double traffic) - Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outer-tag + source-inner-tag + packet	RSPAN BD tag + Source-inner-tag + packet
pop1 tag		
pop2 tag		
push1 tag		

Table 9: Destination Port Ingress and Egress Spanned Traffic for TEFP RSPAN BD

	Ingress Traffic	Egress Traffic
(Untagged traffic)- Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**

	Ingress Traffic	Egress Traffic
no-rewrite	RSPAN BD tag + packet	RSPAN BD tag + packet
pop1 tag	NA	NA
pop2 tag	NA	NA
push1 tag	NA	NA
(Single traffic)-Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outertag + packet	RSPAN BD tag + source-outertag + packet
pop1 tag		
pop2 tag		NA
push1 tag		RSPAN BD tag + source-outertag + packet
(Double traffic) -Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outertag + source-innertag+ packet	RSPAN BD tag + source-outertag + source-innertag + packet
pop1 tag		
pop2 tag		
push1 tag		

Table 10: Destination Port Ingress and Egress Spanned Traffic for RSPAN BD with VPLS Pseudowire (RSP2 module)

	Ingress Traffic	Egress Traffic
(Untagged traffic) - Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + packet	RSPAN BD tag + packet
pop1 tag	NA	NA
pop2 tag	NA	NA
push1 tag	NA	NA
(Single traffic)- Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outer-tag + packet	RSPAN BD tag + source-outer-tag + packet
pop1 tag		

	Ingress Traffic	**Egress Traffic**
pop2 tag	NA	NA
push1 tag	RSPAN BD tag + source-outer-tag + packet	RSPAN BD tag + source-outer-tag + packet
(Double traffic)-Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outer-tag + source-inner-tag + packet	RSPAN BD tag + source-outer-tag + source-inner-tag + packet
pop1 tag		
pop2 tag		
push1 tag		

Configuring Local SPAN and RSPAN

Configuring Sources and Destinations for Local SPAN

To configure sources and destinations for a SPAN session:

SUMMARY STEPS

1. **configure terminal**
2. **monitor session {***session_number***} type local**
3. **source interface** *interface_type slot/subslot/port* **[, | - | rx | tx | both]**
4. **destination interface** *interface_type slot/subslot/port* **[, | -]**
5. **no shutdown**
6. **End**

DETAILED STEPS

Step 1 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 2 **monitor session {***session_number***} type local**

Example:

```
Router(config)# monitor session 1 type local
```

Specifies the local SPAN session number and enters the local monitoring configuration mode.

- *session_number*—Indicates the monitor session. The valid range is 1 through 14.

Step 3 **source interface** *interface_type slot/subslot/port* **[, | - | rx | tx | both]**

Example:

```
Router(config-mon-local)# source interface gigabitethernet 0/2/1 rx
```

Specifies the source interface and the traffic direction:

- *interface_type*—Specifies the Gigabit Ethernet or Ten Gigabit Ethernet interface.
  - *slot/subslot/port*—The location of the interface.
- ","—List of interfaces
- "–"—Range of interfaces
- rx—Ingress local SPAN
- tx—Egress local SPAN
- both

Step 4 **destination interface** *interface_type slot/subslot/port* **[, | -]**

Example:

```
Router(config-mon-local)# destination interface gigabitethernet 0/2/4
```

Specifies the destination interface that sends both ingress and egress local spanned traffic from source port to the prober or sniffer.

- *interface_type*—Specifies the Gigabit Ethernet or Ten Gigabit Ethernet interface.
  - *slot/subslot/port*—The location of the interface.
- ","—List of interfaces
- "–"—Range of interfaces

Step 5 **no shutdown**

Example:

```
Router(config-mon-local)# no shutdown
```

Enables the local SPAN session.

Step 6 **End**

Removing Sources or Destinations from a Local SPAN Session

To remove sources or destinations from a local SPAN session, use the following commands beginning in EXEC mode:

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **no monitor session** *session-number*

Example:

```
Router(config)# no monitor session 2
```

Clears existing SPAN configuration for a session.

Configuring RSPAN Source Session

To configure the source for a RSPAN session:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **monitor session** *RSPAN_source_session_number* **type rspan-source**
4. **Filter vlan***vlan id*
5. **source** {*single_interface* slot/subslot/port| *single_vlan* [**rx** | **tx** | **both**]
6. **destination remote vlan** *rspan_vlan_ID*
7. **no shutdown**
8. **end**

DETAILED STEPS

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **monitor session** *RSPAN_source_session_number* **type rspan-source**

Example:

```
Router(config)# monitor session 1
 type rspan-source
```

Configures an RSPAN source session number and enters RSPAN source session configuration mode for the session.

- *RSPAN_source_session_number—*

  Valid sessions are 1 to 14.
- **rspan-source**—Enters the RSPAN source-session configuration mode.

Step 4 **Filter vlan***vlan id*

Example:

```
filter vlan 100
```

Applies the VLAN access map to the VLAN ID; valid values are from 1 to 4094.

Step 5 **source** {*single_interface* slot/subslot/port| *single_vlan* [**rx** | **tx** | **both**]

Example:

```
Router(config-mon-rspan-src)# source interface gigabitethernet 0/2/1 tx
```

Specifies the RSPAN session number, the source interfaces and the traffic direction to be monitored.

- *single_interface*—Specifies the Gigabit Ethernet or Ten Gigabit Ethernet interface.
  - *slot/subslot/port*—The location of the interface.
- *single_vlan*

  —Specifies the single VLAN.
- **both**

  —(Optional) Monitors the received and the transmitted traffic.
- **rx**

  —(Optional) Monitors the received traffic only.
- **tx**—(Optional) Monitors the transmitted traffic only.

Step 6 **destination remote vlan** *rspan_vlan_ID*

Example:

```
Router(config-mon-rspan-src)# destination remote vlan2
```

Associates the RSPAN source session number session number with the RSPAN VLAN.

- *rspan_vlan_ID*—Specifies the Vlan ID.

  Note *rspan_vlan_ID* is the RSPAN BD that is configured under the EFP or port which carries the RSPANd traffic.

Step 7 **no shutdown**

Example:

```
Router(config-mon-rspan-src)# no shutdown
```

Enables RSPAN source.

Step 8 **end**

Example:

```
Router(config-mon-rspan-src)# end
```

Exists the configuration.

Configuring RSPAN Destination Session

To configure the destination for a RSPAN session for remote Vlan:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **monitor session** *RSPAN_destination_session_number* **type rspan-destination**
4. **source remote vlan** *rspan_vlan_ID*
5. **destination** {*single_interface slot/subslot/port}*
6. **no shutdown**
7. **end**

DETAILED STEPS

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **monitor session** *RSPAN_destination_session_number* **type rspan-destination**

Example:

```
Router(config)# monitor session 1 type rspan-destination
```

Configures a RPAN session.

- *RSPAN_destination_session_number*—Valid sessions are 1 to 80.
- **rspan-destination**—Enters the RSPAN destination-session configuration mode.

Step 4 **source remote vlan** *rspan_vlan_ID*

Example:

```
Router(config-mon-rspan-dst)# source remote vlan2
```

Associates the RSPAN destination session number RSPAN VLAN.

- *rspan_vlan_ID*—Specifies the Vlan ID

Step 5 **destination** {*single_interface slot/subslot/port}*

Example:

```
Router(config-mon-rspan-dst)# destination interface gigabitethernet 0/0/1
```

Associates the RSPAN destination session number with the destination port.

- *single_interface* —Specifies the Gigabit Ethernet or Ten Gigabit Ethernet interface.
  - *slot/subslot/port*—The location of the interface.

Step 6 **no shutdown**

Example:

```
Router(config-mon-rspan-dst)# no shutdown
```

Restarts the interface

Step 7 **end**

Example:

```
Router(config-mon-rspan-dst)# end
```

Exists the configuration

Removing Sources or Destinations from a RSPAN Session

To remove source or destination from a RSPAN session, delete and recreate the RSPAN session. The following are the steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no monitor session** *session number*

4. **end**

DETAILED STEPS

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **no monitor session** *session number*

Example:

```
Router(config)# no monitor session 1
```

Exits monitor session.

Step 4 **end**

Example:

```
Router(config-mon-rspan-src)# end
```

Exits configuration mode.

Sample Configurations

The following sections contain configuration example for SPAN and RSPAN on the router.

Configuration Example: Local SPAN

The following example shows how to configure local SPAN session 8 to monitor bidirectional traffic from source interface Gigabit Ethernet interface to destination:

```
Router(config)# monitor session 8 type local
Router(config)# source interface gigabitethernet 0/0/10
Router(config)# destination interface gigabitethernet 0/0/3
Router(config)# no shut
```

Configuration Example: Removing Sources or Destinations from a Local SPAN Session

This following example shows how to remove a local SPAN session:

```
Router(config)# no monitor session 8
```

Configuration Example: RSPAN Source

The following example shows how RSPAN session 2 to monitor bidirectional traffic from source interface Gigabit Ethernet 0/0/1:

```
Router(config)# monitor session 2 type RSPAN-source
Router(config-mon-RSPAN-src)# source interface gigabitEthernet0/0/1 [tx |rx|both]
Router(config-mon-RSPAN-src)# destination remote VLAN 100
Router(config-mon-RSPAN-src)# no shutdown
Router(config-mon-RSPAN-src)# end
```

The following example shows how RSPAN session 3 to monitor bidirectional traffic from source Vlan 200:

```
Router(config)# monitor session 3 type RSPAN-source
Router(config-mon-RSPAN-src)# filter vlan 100
Router(config-mon-RSPAN-src)# source interface Te0/0/23 rx
Router(config-mon-RSPAN-src)# destination remote VLAN 200
Router(config-mon-RSPAN-src)# no shutdown
Router(config-mon-RSPAN-src)# end
```

Configuration Example: RSPAN Destination

The following example shows how to configure interface Gigabit Ethernet 0/0/1 as the destination for RSPAN session 2:

```
Router(config)# monitor session 2 type RSPAN-destination
Router(config-mon-RSPAN-dst)# source remote VLAN 100
Router(config-mon-RSPAN-dst)# destination interface gigabitEthernet 0/0/1
Router(config-mon-RSPAN-dst)# end
```

Verifying Local SPAN and RSPAN

Use the **show monitor session** command to view the sessions configured.

- The following example shows the Local SPAN source session with Tx as source:

```
Router# show monitor session 8
Session 8
---------
Type : Local Session
Status : Admin Enabled
Source Ports :
TX Only : Gi0/0/10
Destination Ports : Gi0/0/3
```

```
MTU : 1464
Dest RSPAN VLAN : 100
```

- The following example shows the RSPAN source session with Gigabit Ethernet interface 0/0/1 as source:

```
Router# show monitor session 2
Session 2
---------
Type                    : Remote Source Session
Status                  : Admin Enabled
Source Ports            :
    Both                : Gi0/0/1
MTU                     : 1464
```

- The following example shows the RSPAN source session with Vlan 20 as source:

```
Router# show monitor session 3
Session 3
---------
Type                    : Remote Source Session
Status                  : Admin Enabled
Source VLANs            :
    RX Only             : 20
MTU                     : 1464
```

- The following example shows the RSPAN destination session with Gigabit Ethernet interface 0/0/1 as destination:

```
Router# show monitor session 2
Session 2
---------
Type                    : Remote Destination Session
Status                  : Admin Enabled
Destination Ports       : Gi0/0/1
MTU                     : 1464
Source RSPAN VLAN : 100
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mcl/allreleasemcl/all-book.html

Standards and RFCs

Standard/RFC	Title
No specific Standards and RFCs are supported by the features in this document.	—

MIBs

MIB	MIBs Link
—	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

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

CHAPTER 14

Configuring IEEE 802.1ad

Provider networks handle traffic from a large number of customers. It is important that one customer's traffic is isolated from the other customer's traffic.

IEEE 802.1ad enables the service providers to use the architecture and protocols of IEEE 802.1Q to offer separate LANs, bridged local area networks, or virtual bridged local area networks to a number of customers, with minimal cooperation or no cooperation between each customer and the service provider.

IEEE 802.1ad implements standard protocols for double tagging of data. The data traffic coming from the customer side are double tagged in the provider network where the inner tag is the customer-tag (C-tag) and the outer tag is the provider-tag (S-tag). The control packets are tunneled by changing the destination MAC address in the provider network.

A service provider's Layer 2 network transports the subscriber's Layer 2 protocols transparently. Provider Bridge allows the service provider switches to transparently carry customer Layer 2 control frames, such as spanning tree Bridge Protocol Data Units (BPDUs) or Cisco proprietary protocol frames such as Cisco Discovery Protocol (CDP) without mixing the service provider's own traffic and with other customer traffic in the service provider's network. A provider bridge is just like a standard 802.1Q bridge, but it imposes a set of requirements, defined by IEEE 802.1ad standards, on a port in a provider bridge which interfaces to customer. This port is a UNI Port. 802.1ad Provider Bridge thus achieves the same functionality as being addressed with L2PT and QinQ.

When Connectivity Fault Management (CFM) is configured on 802.1ad interfaces, all CFM, Link Ethernet Operations, Administration, and Maintenance (OAM), Enhanced Local Management Interface (ELMI) or Y.1731 performance monitoring packets have their own peer or data rules depending on the type of 802.1ad port configured.

Prerequisites for 802.1ad

- Ethertype should be configured.

Restrictions for 802.1ad

- 802.1ad is supported only on EFP and Trunk EFPs (TEFP).
- SVI based cross-connect, EFP local connect is *not* supported.
- Termination of Layer3 interfaces is *not* supported.
- QoS support is same as supported on the 802.1q EVCs.
- QoS support for classification, marking, weighted tail-drop is not supported on 802.1ad EVCs.
- Routing over BDI with 802.1ad EVC is *not* supported.
- Outer tag Ethertype 0X88a8 is only supported.
- Global **dot1ad** command is not supported.
- Ethernet 802.1ad is *not* supported on port-channels.
- VPLS is not supported for 802.1ad.
- VLAN Translate is not supported for 802.1ad.
- **l2protocol peer** and **l2protocol drop** commands are *not* supported.
- CFI is *not* retained for rewrite push operation.
- Inconsistent VLAN tagging behavior may been seen with rewrite push is configured for S-UNI port and rewrite pop is configured for NNI port
- Translation of UNI-C to NNI port is *not* supported for EVC configuration.
- CFM and Y.1731 are not supported on Smart SFP (SSFP) enabled interfaces for both xconnect and BD.
- NNI port does not drop packets with dot1q outer tag.
- **Encapsulation dot1ad <dot1q>** on NNI ports with rewrite configured as `rewrite ingress tag pop 2 symm` at egress results in pushing 2 dot1q tags. It is advised not to use **pop 2**.

Rewrite Configuration Model for 802.1ad Ports

Note This section is not applicable for RSP3 Module

The table describes the rewrite configuration supported on service instances for the C-UNI, S-UNI and NNI ports

Port	Rewrite	Ingress Direction	Egress Direction
Bridge Domain			
C-UNI	Push	Supported	Not supported
S-UNI			

Port	Rewrite	Ingress Direction	Egress Direction
NNI	Pop	Supported	Not supported
	Push	Not supported	Supported
Trunk EFP			
C-UNI	Pop	Supported	Supported
S-UNI	NA	NA	NA
NNI	Pop	Supported	Not supported
	Push	Not supported	Supported
Cross Connect			
C-UNI	Push	Supported	Not supported
S-UNI			
NNI	NA	NA	NA

Information About 802.1ad

802.1ad Ports

In 802.1ad, a port is configured as either a customer user-network interface (C-UNI), a service-provider UNI (S-UNI), or a network-to-network interface (NNI). Only Layer 2 interfaces can be 802.1ad ports.

- **C-UNI**—an be either an access port or an 802.1Q trunk port. The port uses the customer bridge addresses. To configure a C-UNI port, enter the ethernet dot1ad uni c-port interface configuration command.
- **S-UNI**—an access port that provides the same service to all customer VLANs entering the interface, marking all C-VLANs entering the port with the same S-VLAN. In this mode, the customer's port is configured as a trunk port, and traffic entering the S-UNI is tagged. Use the ethernet dot1ad uni s-port interface configuration command on an access port with an access VLAN.
- **NNI**—entering the ethernet dot1ad nni interface command on a trunk port creates 802.1ad EtherType (0x88a8) and uses S-bridge addresses for CPU-generated Layer 2 protocol PDUs.

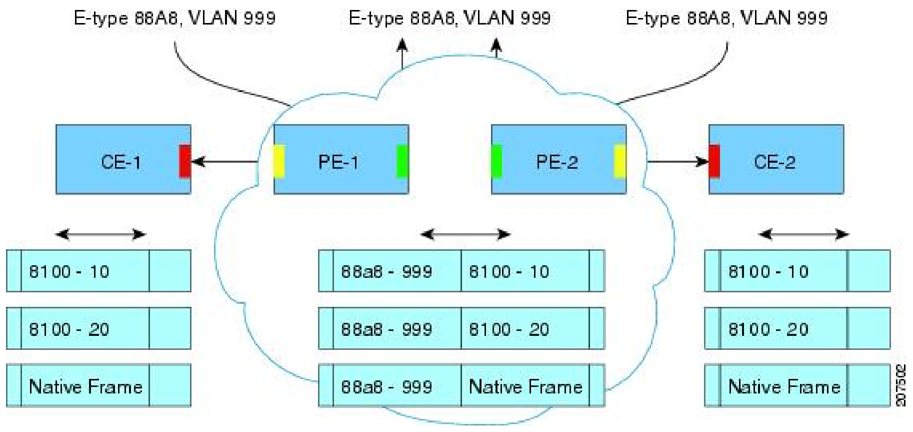
Service Provider Bridges

Provider bridges pass the network traffic of multiple customers. The traffic flow of each customer must be isolated from one another. For Layer 2 protocols within customer domains to function properly, geographically separated customer sites must appear to be connected via a LAN and the provider network must be transparent.

The IEEE has reserved 33 Layer 2 MAC addresses for customer devices that operate Layer 2 protocols. If a provider bridge uses these standard MAC addresses for its Layer 2 protocols, the Layer 2 traffic of the customer devices and the service provider is mixed together. Provider bridges solve this traffic-mixing issue by providing

Layer 2 protocol data unit (PDU) tunneling when a provider bridge (S-bridge) component and a provider edge bridge (C-bridge) component are used. The figure below shows the topology.

Figure 7: Layer 2 PDU Tunneling



S-Bridge Component

The S-bridge component is capable of inserting or removing a service provider VLAN (S-VLAN) for all traffic on a particular port. IEEE 802.1ad adds a new tag called a Service tag (S-tag) to all ingress frames traveling from the customer to the service provider.

The VLAN in the S-tag is used for forwarding the traffic in the service provider network. Different customers use different S-VLANs, which results in isolation of traffic of each customer. In the S-tag, provider bridges do not understand the standard Ethertype. Hence, they use an Ethertype value that is different from the standard 802.1Q Ethertype value. This difference makes customer traffic that is tagged with the standard Ethertype appear as untagged in the provider network. The customer traffic is tunneled in the port VLAN of the provider port. 802.1ad service provider user network interfaces (S-UNIs) and network-network interfaces (NNIs) implement the S-bridge component.

For example, a VLAN tag has a VLAN ID of 1, the C-tag Ethertype has a value of 8100 0001, the S-tag Ethertype has a value of 88A8 0001, and the class of service (CoS) has a value of zero.

C-tag S-tag

-- ---

0x8100 | Priority bits | CFI | C-VLAN-ID 0x88A8 | Priority bits | 0 | S-VLAN-ID

-- ---

C-Bridge Component

All customer VLANs (C-VLANs) that enter a user network interface (UNI) port in an S-bridge component receive the same service (marked with the same S-VLAN). C-VLAN components are not supported, but a customer may want to tag a particular C-VLAN packet separately to differentiate between services. Provider bridges allow C-VLAN packet tagging with a provider edge bridge, called the C-bridge component of the provider bridge. C-bridge components are C-VLAN aware and can insert or remove a C-VLAN 802.1Q tag. The C-bridge UNI port is capable of identifying the customer 802.1Q tag and inserting or removing an S-tag on the packet on a per-service instance or C-VLAN basis. A C-VLAN tagged service instance allows service instance selection and identification by C-VLAN. The 801.1ad customer user network interfaces (C-UNIs) implement the C-component.

NNI Port

Dot1ad NNI port are core facing ports. On this port dot1ad (0x88A8) ethertype is used. The customer facing S-bridge port is identified by using the ethernet dot1ad nni command. The frames forwarded on this port are double tagged with the S-Tag ethertype set at. 0x88a8.

MAC Addresses for Layer 2 Protocols

Layer 2 protocol data units (PDUs) of customers that are received by a provider bridge are not forwarded. Hence, Layer 2 protocols running at customer sites do not know the complete network topology. By using different set of addresses for the Layer 2 protocols running on provider bridges, IEEE 802.1ad causes Layer 2 PDUs of the customers device that enter the provider bridge to appear as unknown multicast traffic and forwards it on customer ports (on the same service provider VLAN (S-VLAN)). Layer 2 protocols of customer device can then run transparently.

The table below shows Layer 2 MAC addresses that are reserved for the C-VLAN component.

Table 11: Reserved Layer 2 MAC Addresses for the C-VLAN Component

Assignment	Value
Bridge Group Address	01-80-C2-00-00-00
IEEE 802.3 Full Duplex PAUSE Operation	01-80-C2-00-00-01
IEEE 802.3 Slow_Protocols_Multicast_Address	01-80-C2-00-00-02
IEEE 802.1X PAE Address	01-80-C2-00-00-03
Provider Bridge Group Address	01-80-C2-00-00-08
Provider Bridge GVRP Address	01-80-C2-00-00-0D
IEEE 802.1AB Link Layer Discovery Protocol Multicast Address	01-80-C2-00-00-0E
Reserved for future standardization	01-80-C2-00-00-04 01-80-C2-00-00-05 01-80-C2-00-00-06 01-80-C2-00-00-07 01-80-C2-00-00-09 01-80-C2-00-00-0A 01-80-C2-00-00-0B 01-80-C2-00-00-0C 01-80-C2-00-00-0F

The table below shows Layer 2 MAC addresses that are reserved for the S-VLAN component. These addresses are a subset of the C-VLAN component addresses, and the C-bridge does not forward the bridge protocol data units (BPDUs) of a provider to a customer network.

Table 12: Reserved Layer 2 MAC Addresses for the S-VLAN Component

Assignment	Value
IEEE 802.3 Full Duplex PAUSE Operation	01-80-C2-00-00-01
IEEE 802.3 Slow_Protocols_Multicast_Address	01-80-C2-00-00-02
IEEE 802.1X PAE Address	01-80-C2-00-00-03
Provider Bridge Group Address	01-80-C2-00-00-08
Reserved for future standardization	01-80-C2-00-00-04 01-80-C2-00-00-05 01-80-C2-00-00-06 01-80-C2-00-00-07 01-80-C2-00-00-09 01-80-C2-00-00-0A

Bridge Protocol Data Units Destination MAC Addresses

The table summarizes the actions when a packet is received with destination MAC address for C-UNI, S-UNI and NNI interfaces.

Table 13: Destination MAC Addresses C-UNI, S-UNI and NNI Ports

MAC Address	Protocol	C-UNI Action	S-UNI Action	NNI Action
01-80-C2-00-00-00	Bridge Protocol Data Units (BPDUs)	Peer	Data	Data
01-80-C2-00-00-01	802.3X Pause Protocol	Drop	Drop	Drop
01-80-C2-00-00-02	Slow protocol address: 802.3ad LACP, 802.3ah OAM	Peer	Peer	Peer
01-80-C2-00-00-03	802.1x	Not supported	Not supported	Not supported
01-80-C2-00-00-04	Reserved for future media access method	Drop	Drop	Drop
01-80-C2-00-00-05	Reserved for future media access method	Drop	Drop	Drop
01-80-C2-00-00-06	Reserved for future bridge use	Drop	Drop	Drop

MAC Address	Protocol	C-UNI Action	S-UNI Action	NNI Action
01-80-C2-00-00-07	Reserved for future bridge use	Drop	Drop	Drop
01-80-C2-00-00-08	Provider STP (BPDU)	Drop	Drop	Peer
01-80-C2-00-00-09	Reserved for future bridge use	Drop	Drop	Drop
01-80-C2-00-00-0A	Reserved for future bridge use	Drop	Drop	Drop
01-80-C2-00-00-0B	Reserved for future S-bridge use	Drop	Drop	Drop
01-80-C2-00-00-0C	Reserved for future S-bridge purposes	Drop	Drop	Drop
01-80-C2-00-00-0D	Provider bridge Generic VLAN Registration Protocol (GVRP) address	Drop	Drop	Drop
01-80-C2-00-00-0E	802.1ab Link Layer Discovery Protocol (LLDP)	may Peer	Data	Data
01-80-C2-00-00-0F	Reserved for future C- bridge or Q-bridge use	Drop	Drop	Drop
01-80-C2-00-00-10	All bridge domains	Not supported	Not supported	Not supported
01-80-C2-00-00-20	GARP Multicast Registration Protocol (GMRP)	Data	Data	Data
01-80-C2-00-00-21	Generic VLAN Registration Protocol (GVRP)	Data	Data	Data
01-80-C2-00-00-22-2F	Other GARP addresses	Data	Data	Data

MAC Address	Protocol	C-UNI Action	S-UNI Action	NNI Action
01-00-0C-CC-CC-CC	Port Aggregation Protocol (PagP), UniDirectional Link Detection (UDLD),	Peer	Peer	Peer
	Cisco Discovery Protocol (CDP), VLAN Trunk Protocol (VTP)	Peer	Data	Data
	Dynamic Trunking Protocol (DTP)	NA	NA	NA
01-00-0C-CC-CC-CD	Per-VLAN Spanning Tree (PVST)	Peer	Data	Data

How to Configure 802.1ad

Configuring the IEEE 802.1ad on Service Instances

Note

- The **rewrite pop** command is *not* supported on C-UNI ports.
- Only **encapsulation default** command is supported on S-UNI ports

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface-id*
4. **ethernet dot1ad** {**nni** | **uni** {**c-port** | **s-port**}}
5. **service instance** *number* **ethernet** [*name*]
6. **encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}
7. **bridge-domain** *bridge-id* [**split-horizon group** *group-id*]
8. **rewrite ingress tag** { **pop** {**1** | **2**} **symmetric** | **push dot1ad** *vlan-id* [**dot1q** *vlan-id*] **symmetric**}
9. **end**

DETAILED STEPS

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

	Command or Action	Purpose
	`Router> enable`	
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enter global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Router(config)# interface gigabitethernet0/0/1`	Enter interface configuration mode. Valid interfaces are physical ports.

Step 4	**ethernet dot1ad** {**nni** | **uni** {**c-port** | **s-port**}}
	Example:
	`Router(config-if)# ethernet dot1ad nni`
	`or`
	`Router(config-if)# ethernet dot1ad uni c-port`
	`or`
	`Router(config-if)# ethernet dot1ad uni s-port`
		Configures dot1ad NNI, C-port or S-port on the interface.

Step 5	**service instance** *number* **ethernet** [*name*] **Example:** `Router(config-if)# service instance 1 Ethernet`	Configure an EFP (service instance) and enter service instance configuration) mode. • The number is the EFP identifier, an integer from 1 to 4000. • (Optional) **ethernet** name is the name of a previously configured EVC. You do not need to use an EVC name in a service instance.

Step 6	**encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}
	Example:
	`Router(config-if-srv)# encapsulation dot1q 10`
		Configure encapsulation type for the service instance.
		• **default**—Configure to match all unmatched packets.
		• **dot1q**—Configure 802.1Q encapsulation. See Encapsulation for details about options for this keyword.
		• **priority-tagged**—Specify priority-tagged frames, VLAN-ID 0 and CoS value of 0 to 7.
		• **untagged**—Map to untagged VLANs. Only one EFP per port can have untagged encapsulation.

Step 7	**bridge-domain** *bridge-id* [**split-horizon group** *group-id*] **Example:** `Router(config-if-srv)# bridge-domain 3000`	Configure the bridge domain ID. The range is from 1 to 4000. You can use the **split-horizon** keyword to configure the port as a member of a split horizon group. The *group-id* range is from 0 to 2.
Step 8	**rewrite ingress tag** { **pop** {**1** | **2**} **symmetric** | **push dot1ad** *vlan-id* [**dot1q** *vlan-id*] **symmetric**}	(Optional) Specify that encapsulation modification to occur on packets at ingress.

	Command or Action	Purpose

	Example:
	`Router(config-if-srv)# **rewrite ingress tag pop 1 symmetric**`
	`Router(config-if-srv)# **rewrite ingress tag push dot1ad 30 symmetric**`
		• **pop 1**—Pop (remove) the outermost tag.
		• **pop 2**—Pop (remove) the two outermost tags.
		• **symmetric—**Configure the packet to undergo the reverse of the ingress action at egress. If a tag is popped at ingress, it is pushed (added) at egress. This keyword is required for **rewrite** to function properly.
		• **push**—Adds a tag to an ingress packet.
		• **dot1ad** *vlan-id*—Specifies the 802.1 do1ad tag. Valid Vlan ID range is from 1 to 4094.
		• **dot1q** *vlan-id*—Specifies the 802.1 do1q tag. Valid Vlan ID range is from 1 to 4094.

Step 9	**end**
	Example:
	`Router(config-if-srv)# **end**`
		Return to privileged EXEC mode.

Configuration Examples

The example shows the C-UNI port.

```
interface GigabitEthernet0/3/7
ethernet dot1ad uni c-port
service instance 20 ethernet
encapsulation dot1q 20
rewrite ingress tag push dot1ad 30 symmetric
bridge-domain 20
```

The example shows the S-UNI port.

```
interface GigabitEthernet0/3/7
ethernet dot1ad uni s-port
service instance 20 ethernet
encapsulation default
rewrite ingress tag push dot1ad 30 symmetric
bridge-domain 20
```

The example shows the NNI port.

```
interface GigabitEthernet0/5/2
ethernet dot1ad nni
service instance 20 ethernet
encapsulation dot1ad 30
bridge-domain 20
```

Configuring the IEEE 802.1ad on Trunk EFP Service Instances

Note

- The **rewrite pop** command is *not* supported on C-UNI ports.
- Trunk EFP is *not* supported on S-UNI ports

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface-id*
4. **ethernet dot1ad** {**nni** | **uni** {**c-port**}}
5. **service instance** [**trunk**] *number* **ethernet**
6. **encapsulation dot1q**
7. **bridge-domin** *bridge-id* **from-encapsulation**
8. **rewrite ingress tag** { **pop** {**1** | **2**} **symmetric**
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enter global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Router(config)# interface gigabitethernet0/0/1`	Enter interface configuration mode. Valid interfaces are physical ports.

Step 4	**ethernet dot1ad** {**nni** | **uni** {**c-port**}}
	Example:
	`Router(config-if)# ethernet dot1ad nni`
	`or`
	`Router(config-if)# ethernet dot1ad uni c-port`
		Configures Trunk EFP on dot1ad NNI and C-UNI ports on the interface.
		Note Trunk EFP is *not* supported on the S-UNI port.

Step 5	**service instance** [**trunk**] *number* **ethernet** **Example:** `Router(config-if)# service instance trunk 1 ethernet`	Configure an EFP (service instance) and enter service instance configuration) mode • The number is the EFP identifier, an integer from 1 to 4000

	Command or Action	Purpose
		• The trunk keyword identifies the trunk ID to which the service instance is assigned. **Note** Trunk EFP (without port channel) supports encapsulation of up to 1000 Vlans.
Step 6	**encapsulation dot1q** **Example:** `Router(config-if-srv)# encapsulation dot1q 10`	Configure encapsulation type for the service instance. • **dot1q**—Configure 802.1Q encapsulation.
Step 7	**bridge-domin** *bridge-id* **from-encapsulation** **Example:** `Router(config-if-srv)# bridge-domain from-encapsulation`	Configures the router to derive bridge domains from the encapsulation VLAN list.
Step 8	**rewrite ingress tag** { **pop** {**1** | **2**} **symmetric** **Example:** `Router(config-if-srv)# rewrite ingress tag pop 1 symmetric`	(Optional) Specify that encapsulation modification to occur on packets at ingress. • **pop 1**—Pop (remove) the outermost tag. • **pop 2**—Pop (remove) the two outermost tags. • **symmetric—**Configure the packet to undergo the reverse of the ingress action at egress. If a tag is popped at ingress, it is pushed (added) at egress. This keyword is required for **rewrite** to function properly.
Step 9	**end** **Example:** `Router(config-if-srv)# end`	Return to privileged EXEC mode.

Configuration Examples

The example shows the Trunk EFP configuration on the C-UNI port.

```
interface GigabitEthernet0/3/7
ethernet dot1ad uni c-port
service instance trunk 20 ethernet
encapsulation dot1q 20-30
rewrite ingress tag pop1 symmetric
bridge-domain from-encapsulation
```

The example shows the Trunk EFP configuration on the NNI port.

```
interface GigabitEthernet0/5/2
ethernet dot1ad nni
service instance trunk 20 ethernet
encapsulation dot1ad 20-30
rewrite ingress tag pop1 symmetric
bridge-domain from-encapsulation
```

Configuring the IEEE 802.1ad on Cross-Connect on EFP

Note

- The **rewrite push** command is supported on C-UNI and S-UNI ports. Rewrite is *not* supported for NNI ports.
- Only **encapsulation default** command is supported on S-UNI ports

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface-id*
4. **ethernet dot1ad** {**nni** | **uni** {**c-port** | **s-port**}}
5. **service instance** *number* **ethernet** [*name*]
6. **encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}
7. **rewrite ingress tag push dot1ad** *vlan-id* [**dot1q** *vlan-id*] **symmetric**}
8. **xconnect** *peer-router-id vcid* **pw-class** *pw-class name*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enter global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Router(config)# interface gigabitethernet0/0/1`	Enter interface configuration mode. Valid interfaces are physical ports.

Step 4	**ethernet dot1ad** {**nni** | **uni** {**c-port** | **s-port**}}
	Example:
	`Router(config-if)# ethernet dot1ad nni`
	`or`
	`Router(config-if)# ethernet dot1ad uni c-port`
	`or`
	`Router(config-if)# ethernet dot1ad uni s-port`
		Configures dot1ad NNI, C-port or S-port on the interface.

Step 5	**service instance** *number* **ethernet** [*name*] **Example:**	Configure an EFP (service instance) and enter service instance configuration) mode.

	Command or Action	Purpose
	`Router(config-if)# service instance 1 Ethernet`	• The number is the EFP identifier, an integer from 1 to 4000. • (Optional) **ethernet** name is the name of a previously configured EVC. You do not need to use an EVC name in a service instance.

Step 6	**encapsulation** {**default** | **dot1q** | **priority-tagged** | **untagged**}
	Example:
	`Router(config-if-srv)# encapsulation dot1q 10`
		Configure encapsulation type for the service instance.
		• **default**—Configure to match all unmatched packets.
		• **dot1q**—Configure 802.1Q encapsulation. See Encapsulation for details about options for this keyword.
		• **priority-tagged**—Specify priority-tagged frames, VLAN-ID 0 and CoS value of 0 to 7.
		• **untagged**—Map to untagged VLANs. Only one EFP per port can have untagged encapsulation.

Step 7	**rewrite ingress tag push dot1ad** *vlan-id* [**dot1q** *vlan-id*] **symmetric**} **Example:** `Router(config-if-srv)# rewrite ingress tag push dot1ad 30 symmetric`	(Optional) Specify that encapsulation modification to occur on packets at ingress. • **symmetric—**Configure the packet to undergo the reverse of the ingress action at egress. If a tag is popped at ingress, it is pushed (added) at egress. This keyword is required for **rewrite** to function properly. • **push**—Adds a tag to an ingress packet. • **dot1ad** *vlan-id*—Specifies the 802.1 do1ad tag. Valid Vlan ID range is from 1 to 4094. • **dot1q** *vlan-id*—Specifies the 802.1 do1q tag. Valid Vlan ID range is from 1 to 4094.
Step 8	**xconnect** *peer-router-id vcid* **pw-class** *pw-class name* **Example:** `Router(config-if-srv)# xconnect 10.10.10.10 123 encapsulation mpls`	Bind the attachment circuit to a pseudowire virtual circuit (VC) and enter cross connect configuration mode.

Step 9	**end**
	Example:
	`Router(config-if-srv)# end`
		Return to privileged EXEC mode.

Configuration Examples

The example shows the C-UNI port.

```
interface GigabitEthernet0/3/7
ethernet dot1ad uni c-port
service instance 20 ethernet
encapsulation dot1q 20
rewrite ingress tag push dot1ad 30 symmetric
xconnect 2.2.2.2 20 encap mpls
```

The example shows the S-UNI port.

```
interface GigabitEthernet0/3/7
ethernet dot1ad uni s-port
service instance 20 ethernet
encapsulation default
rewrite ingress tag push dot1ad 30 symmetric
xconnect 2.2.2.2 20 encap mpls
```

The example shows the NNI port.

```
interface GigabitEthernet0/3/7
ethernet dot1ad nni
service instance 20 ethernet
encapsulation dot1ad 20
xconnect 2.2.2.2 20 encap mpls
```

Verifying IEEE 802.1ad

Use the **show ethernet do1ad**commands to verify IEEE 802.1ad configuration.

- **show ethernet do1ad**

  This command displays 802.1ad configuration globally on the router. The following is a sample output from the command:

```
Router# show ethernet dot1ad

Interface: GigabitEthernet0/2/1
DOT1AD NNI Port
L2protocol pass

Interface: GigabitEthernet0/2/7
DOT1AD C-Bridge Port
L2protocol pass
```

- **show ethernet do1ad** [**inteface** *inteface-name*]

  This command displays interface dot1ad configuration The following is a sample output from the command:

```
Router# show ethernet do1ad interface gigabitethernet 0/2/1

Interface: GigabitEthernet0/2/1
DOT1AD NNI Port
L2protocol pass
```