Interface and Hardware Component Configuration Guide for Cisco NCS 540 Series Routers, IOS XR Release 6.3.x

First Published: 2018-03-30

Last Modified: 2019-01-22

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

Preconfiguring Physical Interfaces

This module describes the preconfiguration of physical interfaces.

Preconfiguration is supported for these types of interfaces and controllers:

- 1-Gigabit Ethernet
- 10-Gigabit Ethernet
- 25-Gigabit Ethernet
- 40-Gigabit Ethernet
- 100-Gigabit Ethernet
- Management Ethernet

Preconfiguration allows you to configure line cards before they are inserted into the router. When the line cards are inserted, they are instantly configured. The preconfiguration information is created in a different system database tree (known as the *preconfiguration directory* on the route processor), rather than with the regularly configured interfaces.

There may be some preconfiguration data that cannot be verified unless the line cards are present, because the verifiers themselves run only on the line cards. Such preconfiguration data is verified when the line cards is inserted and the verifiers are initiated. A configuration is rejected if errors are found when the configuration is copied from the preconfiguration area to the active area.

Note One Gigabit Ethernet interface is not supported. Only physical interfaces can be preconfigured.

Note Eight quadrature amplitude modulation (8QAM) requires V2 (or higher) CFP2 version and 5.23 (or higher) firmware.

Physical Interface Preconfiguration Overview

Preconfiguration is the process of configuring interfaces before they are present in the system. Preconfigured interfaces are not verified or applied until the actual interface with the matching location (rack/slot/module) is inserted into the router. When the anticipated line card is inserted and the interfaces are created, the precreated configuration information is verified and, if successful, immediately applied to the running configuration of the router.

Note When you plug the anticipated line card in, make sure to verify any preconfiguration with the appropriate **show** commands.

Use the **show run** command to see interfaces that are in the preconfigured state.

Note We recommend filling out preconfiguration information in your site planning guide, so that you can compare that anticipated configuration with the actual preconfigured interfaces when that line card is installed and the interfaces are up.

Tip Tip Use the **commit best-effort** command to save the preconfiguration to the running configuration file. The **commit best-effort** command merges the target configuration with the running configuration and commits only valid configuration (best effort). Some configuration might fail due to semantic errors, but the valid configuration still comes up.

Prerequisites for Preconfiguring Physical Interfaces

Before preconfiguring physical interfaces, ensure that this condition is met:

- Preconfiguration drivers and files are installed. Although it may be possible to preconfigure physical interfaces without a preconfiguration driver installed, the preconfiguration files are required to set the interface definition file on the router that supplies the strings for valid interface names.

Benefits of Interface Preconfiguration

Preconfigurations reduce downtime when you add new cards to the system. With preconfiguration, the new cards can be instantly configured and actively running during cards bootup.

Another advantage of performing a preconfiguration is that during a cards replacement, when the cards is removed, you can still see the previous configuration and make modifications.

How to Preconfigure Physical Interfaces

This task describes only the most basic preconfiguration of an interface.

SUMMARY STEPS

1. **configure**
2. **interface preconfigure** *type interface-path-id*
3. Use one of the following commands:
   - **ipv4 address** *ip-address subnet-mask*
   - **ipv4 address** *ip-address / prefix*
4. Configure additional interface parameters, as described in this manual in the configuration chapter that applies to the type of interface that you are configuring.
5. **end** or **commit** best-effort
6. **show running-config**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router#configure
```

Enters global configuration mode.

Step 2 **interface preconfigure** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface preconfigure HundredGigE 0/0/1/0
```

Enters interface preconfiguration mode for an interface, where *type* specifies the supported interface type that you want to configure and *interface-path-id* specifies the location where the interface will be located in *rack*/*slot*/*module*/*port* notation.

Step 3 Use one of the following commands:

- **ipv4 address** *ip-address subnet-mask*
- **ipv4 address** *ip-address / prefix*

Example:

```
RP/0/RP0/CPU0:router(config-if-pre)# ipv4 address 192.168.1.2/31
```

Assigns an IP address and mask to the interface.

Step 4 Configure additional interface parameters, as described in this manual in the configuration chapter that applies to the type of interface that you are configuring.

Step 5 **end** or **commit** best-effort

Example:

```
RP/0/RP0/CPU0:router(config-if-pre)# end
```

or

```
RP/0/RP0/CPU0:router(config-if-pre)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting (yes/no/cancel)?`
- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit best-effort** command to save the configuration changes to the running configuration file and remain within the configuration session. The **commit best-effort** command merges the target configuration with the running configuration and commits only valid changes (best effort). Some configuration changes might fail due to semantic errors.

Step 6 **show running-config**

Example:

```
RP/0/RP0/CPU0:router# show running-config
```

(Optional) Displays the configuration information currently running on the router.

Example

This example shows how to preconfigure a basic Ethernet interface:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface preconfigure HundredGigE 0/0/1/0
RP/0/RP0/CPU0:router(config-if)# ipv4 address 192.168.1.2/31
RP/0/RP0/CPU0:router(config-if-pre)# commit
```

Use of the Interface Preconfigure Command

Interfaces that are not yet present in the system can be preconfigured with the **interface preconfigure** command in global configuration mode.

The **interface preconfigure** command places the router in interface configuration mode. Users should be able to add any possible interface commands. The verifiers registered for the preconfigured interfaces verify the configuration. The preconfiguration is complete when the user enters the **end** command, or any matching exit or global configuration mode command.

Note It is possible that some configurations cannot be verified until the line card is inserted.

Do not enter the **no shutdown** command for new preconfigured interfaces, because the no form of this command removes the existing configuration, and there is no existing configuration.

Users are expected to provide names during preconfiguration that will match the name of the interface that will be created. If the interface names do not match, the preconfiguration cannot be applied when the interface is created. The interface names must begin with the interface type that is supported by the router and for which drivers have been installed. However, the slot, port, subinterface number, and channel interface number information cannot be validated.

Note Specifying an interface name that already exists and is configured (or an abbreviated name like Hu0/3/0/0) is not permitted.

CHAPTER 2

Configuring Management Ethernet Interface

This module describes the configuration of Management Ethernet interfaces.

Before you can use Telnet to access the router through the LAN IP address, you must set up a Management Ethernet interface and enable Telnet servers.

> **Note** Although the Management Ethernet interfaces on the system are present by default, you must configure these interfaces to use them for accessing the router, using protocols and applications such as Simple Network Management Protocol (SNMP), HTTP, extensible markup language (XML), TFTP, Telnet, and command-line interface (CLI).

> **Note** In a High Availability setup, when an active RP interface is shut the ping to gateway fails, though standby RP or virtual RP is up and running. RSP4 does not support inject packets from a standby RP management interface.

Information About Configuring Management Ethernet Interfaces

To configure Management Ethernet interfaces, you must understand the following concept:

Prerequisites for Configuring Management Ethernet Interfaces

Before performing the Management Ethernet interface configuration procedures that are described in this chapter, be sure that the following tasks and conditions are met:

- You have performed the initial configuration of the Management Ethernet interface.
- You know how to apply the generalized interface name specification *rack/slot/module/port*.

> **Note** For transparent switchover, both active and standby Management Ethernet interfaces are expected to be physically connected to the same LAN or switch.

Restrictions on Cisco NCS 540 Series Routers

The restrictions are applicable on the following routers.

- N540-28Z4C-SYS-A
- N540-28Z4C-SYS-D
- N540X-16Z4G8Q2C-A
- N540X-16Z4G8Q2C-D
- N540-12Z20G-SYS-A
- N540-12Z20G-SYS-D
- N540X-12Z16G-SYS-A
- N540X-12Z16G-SYS-D

- Virtual IPv6 addressing is not supported on the management interface.

How to Perform Advanced Management Ethernet Interface Configuration

This section contains the following procedures:

IPv6 Stateless Address Auto Configuration on Management Interface

Perform this task to enable IPv6 stateless auto configuration on Management interface.

SUMMARY STEPS

1. **configure**
2. **interface MgmtEth** *interface-path-id*
3. **ipv6 address autoconfig**
4. **end** or **commit**
5. **show ipv6 interfaces** *interface-path-id*

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface MgmtEth** *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface MgmtEth 0/RP0/CPU0/0
```

Enters interface configuration mode and specifies the Ethernet interface name and notation *rack*/*slot*/*module*/*port*.

The example indicates port 0 on the RP card that is installed in slot 0.

Step 3 **ipv6 address autoconfig**

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv6 address autoconfig
```

Enable IPv6 stateless address auto configuration on the management port.

Step 4 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# end
```

or

```
RP/0/RP0/CPU0:router(config-if)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?
  [cancel]:
  ```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5 **show ipv6 interfaces** *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router# show ipv6 interfaces gigabitEthernet 0/0/0/0
```

(Optional) Displays statistics for interfaces on the router.

Example

This example displays :

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface MgmtEth 0/RP0/CPU0/0
RP/0/RP0/CPU0:router(config-if)# ipv6 address autoconfig
RP/0/RP0/CPU0:router(config-if)# commit
RP/0/RP0/CPU0:router# show ipv6 interfaces gigabitEthernet 0/0/0/0

Fri Nov  4 16:48:14.372 IST
GigabitEthernet0/2/0/0 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::d1:1eff:fe2b:baf
  Global unicast address(es):
    5::d1:1eff:fe2b:baf [AUTO CONFIGURED], subnet is 5::/64  <<<<<<  auto configured address

  Joined group address(es): ff02::1:ff2b:baf ff02::2 ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Configuring a Management Ethernet Interface

Perform this task to configure a Management Ethernet interface. This procedure provides the minimal configuration required for the Management Ethernet interface.

SUMMARY STEPS

1. **configure**
2. **interface MgmtEth** *interface-path-id*
3. **ipv4 address** *ip-address mask*
4. **no ipv6 address autoconfig**
5. **mtu** *bytes*
6. **no shutdown**
7. **end** or **commit**
8. **show interfaces MgmtEth** *interface-path-id*

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface MgmtEth** *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface MgmtEth 0/RP0/CPU0/0
```

Enters interface configuration mode and specifies the Ethernet interface name and notation *rack*/*slot*/*module*/*port*.

The example indicates port 0 on the RP card that is installed in slot 0.

Step 3 **ipv4 address** *ip-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 1.76.18.150/16 (or)
ipv4 address 1.76.18.150 255.255.0.0
```

Assigns an IP address and subnet mask to the interface.

- Replace *ip-address* with the primary IPv4 address for the interface.
- Replace *mask* with the mask for the associated IP subnet. The network mask can be specified in either of two ways:
- The network mask can be a four-part dotted decimal address. For example, 255.255.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address.
- The network mask can be indicated as a slash (/) and number. For example, /16 indicates that the first 16 bits of the mask are ones, and the corresponding bits of the address are network address.

Step 4 **no ipv6 address autoconfig**

Example:

```
RP/0/RP0/CPU0:router(config-if)# no ipv6 address autoconfig
```

(Optional) Disables IPv6 address on the interface.

Step 5 **mtu** *bytes*

Example:

```
RP/0/RP0/CPU0:router(config-if)# mtu 1488
```

(Optional) Sets the maximum transmission unit (MTU) byte value for the interface. The default is 1514.

- The default is 1514 bytes.
- The range for the Management Ethernet interface Interface **mtu** values is 64 to 1514 bytes.

Step 6 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-if)# no shutdown
```

Removes the shutdown configuration, which removes the forced administrative down on the interface, enabling it to move to an up or down state.

Step 7 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# end
```

or

```
RP/0/RP0/CPU0:router(config-if)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8 **show interfaces MgmtEth** *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router# show interfaces MgmtEth 0/RP0/CPU0/0
```

(Optional) Displays statistics for interfaces on the router.

Example

This example displays advanced configuration and verification of the Management Ethernet interface on the RP:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface MgmtEth 0/RP0/CPU0/0
RP/0/RP0/CPU0:router(config-if)# ipv4 address 1.76.18.150/16
RP/0/RP0/CPU0:router(config-if)# no ipv6 address autoconfig
RP/0/RP0/CPU0:router(config-if)# no shutdown
```

```
RP/0/RP0/CPU0:router(config-if)# commit
RP/0/RP0/CPU0:router:Mar 26 01:09:28.685 :ifmgr[190]:%LINK-3-UPDOWN :Interface
MgmtEth0/RP0/CPU0/0, changed state to Up
RP/0/RP0/CPU0:router(config-if)# end

RP/0/RP0/CPU0:router# show interfaces MgmtEth 0/RP0/CPU0/0

MgmtEth0/RP0/CPU0/0 is up, line protocol is up
  Interface state transitions: 3
  Hardware is Management Ethernet, address is 1005.cad8.4354 (bia 1005.cad8.4354)
  Internet address is 1.76.18.150/16
  MTU 1488 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 1000Mb/s, 1000BASE-T, link type is autonegotiation
  loopback not set,
  Last link flapped 00:00:59
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, output 00:00:02
  Last clearing of "show interface" counters never
  5 minute input rate 4000 bits/sec, 3 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     21826 packets input, 4987886 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 12450 broadcast packets, 8800 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     1192 packets output, 217483 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     3 carrier transitions

RP/0/RP0/CPU0:router# show running-config interface MgmtEth 0/RP0/CPU0/0

interface MgmtEth0/RP0/CPU0/0
 mtu 1488
 ipv4 address 1.76.18.150/16
 ipv6 address 2002::14c:125a/64
 ipv6 enable
!
```

The following example displays VRF configuration and verification of the Management Ethernet interface on the RP with source address:

```
RP/0/RP0/CPU0:router# show run interface MgmtEth 0/RP0/CPU0/0
interface MgmtEth0/RP0/CPU0/0
 vrf httpupload
 ipv4 address 10.8.67.20 255.255.0.0
 ipv6 address 2001:10:8:67::20/48
!

RP/0/RP0/CPU0:router# show run http
Wed Jan 30 14:58:53.458 UTC
http client vrf httpupload
http client source-interface ipv4 MgmtEth0/RP0/CPU0/0

RP/0/RP0/CPU0:router# show run vrf
Wed Jan 30 14:59:00.014 UTC
vrf httpupload
!
```

Verifying Management Ethernet Interface Configuration

Perform this task to verify configuration modifications on the Management Ethernet interfaces.

SUMMARY STEPS

1. **show interfaces MgmtEth** *interface-path-id*
2. **show running-config interface MgmtEth** *interface-path-id*

DETAILED STEPS

Step 1 **show interfaces MgmtEth** *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router# show interfaces MgmtEth 0/RP0/CPU0/0
MgmtEth0/RP0/CPU0/0 is up, line protocol is up
  Interface state transitions: 3
  Hardware is Management Ethernet, address is 4ce1.767c.3b00 (bia 4ce1.767c.3b00)
  Internet address is 1.76.18.150/16
  MTU 1488 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 1000Mb/s, 1000BASE-T, link type is autonegotiation
  loopback not set,
  Last link flapped 11w5d
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, output 02:26:34
  Last clearing of "show interface" counters never
  5 minute input rate 12000 bits/sec, 11 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     12703775975 packets input, 3169461581071 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 12696055435 broadcast packets, 3869832 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     15836861 packets output, 6225384076 bytes, 0 total output drops
     Output 0 broadcast packets, 4 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     3 carrier transitions
```

Displays the Management Ethernet interface configuration.

Step 2 **show running-config interface MgmtEth** *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router# show running-config interface MgmtEth 0/RP0/CPU0/0
interface MgmtEth0/RP0/CPU0/0
 mtu 1488
 vrf mgmt
 ipv4 address 1.76.18.150 255.255.0.0
!
RP/0/RP0/CPU0: router#
```

Displays the running configuration.

CHAPTER 3

Configuring Ethernet Interfaces

This module describes the configuration of Ethernet interfaces.

The following distributed ethernet architecture delivers network scalability and performance, while enabling service providers to offer high-density, high-bandwidth networking solutions.

- 1-Gigabit
- 10-Gigabit
- 25-Gigabit
- 40-Gigabit
- 100-Gigabit

> **Tip** You can programmatically configure and manage the Ethernet interfaces using `openconfig-ethernet-if.yang` and `openconfig-interfaces.yang` OpenConfig data models. To get started with using data models, see the *Programmability Configuration Guide for Cisco NCS 540 Series Routers*.

These solutions are designed to interconnect the router with other systems in point-of-presence (POP)s, including core and edge routers and Layer 2 and Layer 3 switches.

Restrictions

Router does not support configuration of the static mac address.

Configuring Physical Ethernet Interfaces

Restrictions and Important Guidelines

- NC55-MPA-12T-S supports 1G optics in eight ports. The ports are 0 to 3 and 8 to 11.
- NC55-MPA-12T-S supports 10G optics in ports 4 to 7.

Use this procedure to create a basic Ethernet interface configuration.

SUMMARY STEPS

1. **show version**
2. **show interfaces** [**GigE** | **TenGigE** | **TwentyFiveGigE** | **FortyGigE** | **HundredGigE**] *interface-path-id*
3. **configure**
4. **interface** [**GigE** | **TenGigE** | **TwentyFiveGigE** | **FortyGigE** | **HundredGigE**] *interface-path-id*
5. **ipv4 address** *ip-address mask*
6. **mtu** *bytes*
7. **no shutdown**
8. **show interfaces** [**GigE TenGigETwentyFiveGigE TwentyFiveGigE FortyGigE HundredGigE**] *interface-path-id*

DETAILED STEPS

Step 1 **show version**

Example:

```
RP/0/RP0/CPU0:router# show version
```

(Optional) Displays the current software version, and can also be used to confirm that the router recognizes the line card.

Step 2 **show interfaces** [**GigE** | **TenGigE** | **TwentyFiveGigE** | **FortyGigE** | **HundredGigE**] *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router# show interface HundredGigE 0/0/1/0
```

(Optional) Displays the configured interface and checks the status of each interface port.

Step 3 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure terminal
```

Enters global configuration mode.

Step 4 **interface** [**GigE** | **TenGigE** | **TwentyFiveGigE** | **FortyGigE** | **HundredGigE**] *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/0/1/0
```

Enters interface configuration mode and specifies the Ethernet interface name and notation *rack/slot/module/port*. Possible interface types for this procedure are:

- GigE
- 10GigE
- 25GigE

- 40GigE
- 100GigE

Note • The example indicates a 100-Gigabit Ethernet interface in the line card in slot 1.

The supported *interface-path-id* ranges are:

- **GigE** — 0/0/0/0 - 0/0/0/31
- **TenGigE** — 0/0/0/0 - 0/0/0/31
- **TwentyFiveGigE** — 0/0/0/24 - 0/0/0/31
- **FortyGigE** — 0/0/1/0 - 0/0/1/1
- **HundredGigE** — 0/0/1/0 - 0/0/1/1

Step 5 **ipv4 address** *ip-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 172.18.189.38 255.255.255.224
```

Assigns an IP address and subnet mask to the interface.

- Replace *ip-address* with the primary IPv4 address for the interface.
- Replace *mask* with the mask for the associated IP subnet. The network mask can be specified in either of two ways:
- The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address.
- The network mask can be indicated as a slash (/) and number. For example, /8 indicates that the first 8 bits of the mask are ones, and the corresponding bits of the address are network address.

Step 6 **mtu** *bytes*

Example:

```
RP/0/RP0/CPU0:router(config-if)# mtu 2000
```

(Optional) Sets the MTU value for the interface.

- The configurable range for MTU values is 1514 bytes to 9646 bytes.
- The default is 1514 bytes for normal frames and 1518 bytes for 802.1Q tagged frames.

Step 7 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-if)# no shutdown
```

Removes the shutdown configuration, which forces an interface administratively down.

Step 8 **show interfaces** [**GigE TenGigETwentyFiveGigE TwentyFiveGigE FortyGigE HundredGigE**] *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router# show interfaces HundredGigE
0/0/1/0
```

(Optional) Displays statistics for interfaces on the router.

Example

This example shows how to configure an interface for a 100-Gigabit Ethernet line card:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/0/1/0
RP/0/RP0/CPU0:router(config-if)# ipv4 address 172.18.189.38 255.255.255.224

RP/0/RP0/CPU0:router(config-if)# mtu 2000

RP/0/RP0/CPU0:router(config-if)# no shutdown
RP/0/RP0/CPU0:router(config-if)# end
Uncommitted changes found, commit them? [yes]: yes


RP/0/RP0/CPU0:router# show interfaces HundredGigE 0/0/1/0
HundredGigE0/0/1/0 is up, line protocol is up
  Interface state transitions: 1
  Hardware is HundredGigE, address is 6219.8864.e330 (bia 6219.8864.e330)
  Internet address is 3.24.1.1/24
  MTU 9216 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
     reliability 255/255, txload 3/255, rxload 3/255
  Encapsulation ARPA,
  Full-duplex, 100000Mb/s, link type is force-up
  output flow control is off, input flow control is off
  Carrier delay (up) is 10 msec
  loopback not set,
  Last link flapped 10:05:07
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:08:56, output 00:00:00
  Last clearing of "show interface" counters never
  5 minute input rate 1258567000 bits/sec, 1484160 packets/sec
  5 minute output rate 1258584000 bits/sec, 1484160 packets/sec
     228290765840 packets input, 27293508436038 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 15 broadcast packets, 45 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     212467849449 packets output, 25733664696650 bytes, 0 total output drops
     Output 23 broadcast packets, 15732 multicast packets
     39 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions


RP/0/RP0/CPU0:router# show running-config interface HundredGigE 0/0/1/0

interface HundredGigE 0/0/1/0
 mtu 9216

 ipv4 address 3.24.1.1 255.255.255.0
 ipv6 address 3:24:1::1/64
```

```
 flow ipv4 monitor perfv4 sampler fsm ingress
!
```

Information About Configuring Ethernet

This section provides the following information sections:

Default Configuration Values for 100-Gigabit Ethernet

This table describes the default interface configuration parameters that are present when an interface is enabled on a 10-Gigabit Ethernet or 100-Gigabit Ethernet line card.

Note You must use the **shutdown** command to bring an interface administratively down. The interface default is **no shutdown**. When a line card is first inserted into the router, if there is no established preconfiguration for it, the configuration manager adds a shutdown item to its configuration. This shutdown can be removed only be entering the **no shutdown** command.

Table 1: 100-Gigabit Ethernet line card Default Configuration Values

Parameter	Configuration File Entry	Default Value
MTU	**mtu**	• 1514 bytes for normal frames • 1518 bytes for 802.1Q tagged frames. • 1522 bytes for Q-in-Q frames.
MAC address	**mac address**	Hardware burned-in address (BIA)

Ethernet MTU

The Ethernet maximum transmission unit (MTU) is the size of the largest frame, minus the 4-byte frame check sequence (FCS), that can be transmitted on the Ethernet network. Every physical network along the destination of a packet can have a different MTU.

Cisco IOS XR software supports two types of frame forwarding processes:

- Fragmentation for IPV4 packets–In this process, IPv4 packets are fragmented as necessary to fit within the MTU of the next-hop physical network.

Note IPv6 does not support fragmentation.

- MTU discovery process determines largest packet size–This process is available for all IPV6 devices, and for originating IPv4 devices. In this process, the originating IP device determines the size of the largest IPv6 or IPV4 packet that can be sent without being fragmented. The largest packet is equal to the smallest MTU of any network between the IP source and the IP destination devices. If a packet is larger than the smallest MTU of all the networks in its path, that packet will be fragmented as necessary. This process ensures that the originating device does not send an IP packet that is too large.

Jumbo frame support is automatically enable for frames that exceed the standard frame size. The default value is 1514 for standard frames and 1518 for 802.1Q tagged frames. These numbers exclude the 4-byte frame check sequence (FCS).

The following list describes the properties of MTUs:

- Each physical port can have a different MTU.
- Main interface of each bundle can have one MTU value.
- L3 sub-interface (bundle or physical) shares MTU profiles and can have a maximum of 3 unique configured MTUs per NPU.

Note L2 sub-interface MTU is not supported.

Link Layer Discovery Protocol (LLDP)

Cisco Discovery Protocol (CDP) is a device discovery protocol that runs over Layer 2. Layer 2 is also known as the data link layer that runs on all Cisco-manufactured devices, such as routers, bridges, access servers, and switches. CDP allows the network management applications to automatically discover and learn about other Cisco devices that connect to the network.

To support non-Cisco devices and to allow for interoperability between other devices, it also supports the IEEE 802.1AB LLDP. LLDP is also a neighbor discovery protocol that is used for network devices to advertise information about themselves to other devices on the network. This protocol runs over the data link layer, which allows two systems running different network layer protocols to learn about each other.

With LLDP, you can also access the information about a particular physical network connection. If you use a non-Cisco monitoring tool (via SNMP,) LLDP helps you identify the Object Identifiers (OIDs) that the system supports. The following are the supported OIDs:

- 1.0.8802.1.1.2.1.4.1.1.4
- 1.0.8802.1.1.2.1.4.1.1.5
- 1.0.8802.1.1.2.1.4.1.1.6
- 1.0.8802.1.1.2.1.4.1.1.7
- 1.0.8802.1.1.2.1.4.1.1.8
- 1.0.8802.1.1.2.1.4.1.1.9
- 1.0.8802.1.1.2.1.4.1.1.10
- 1.0.8802.1.1.2.1.4.1.1.11

- 1.0.8802.1.1.2.1.4.1.1.12

Enabling LLDP Globally

To run LLDP on the router, you must enable it globally. When you enable LLDP globally, all interfaces that support LLDP are automatically enabled for both transmit and receive operations.

You can override this default operation at the interface to disable receive or transmit operations.

The following table describes the global attributes that you can configure:

Attribute	Default	Range	Description
Holdtime	120	0-65535	Specifies the holdtime (in sec) that are sent in packets
Reinit	2	2-5	Delay (in sec) for LLDP initialization on any interface
Timer	30	5-65534	Specifies the rate at which LLDP packets are sent (in sec)

To enable LLDP globally, complete the following steps:

1. `RP/0/RP0/CPU0:router # configure`
2. `RP/0/RP0/CPU0:router(config) #lldp`
3. `end or commit`

Running configuration

```
RP/0/RP0/CPU0:router-5#show run lldp
Fri Dec 15 20:36:49.132 UTC
lldp
!

RP/0/RP0/CPU0:router#show lldp neighbors
Fri Dec 15 20:29:53.763 UTC
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
SW-NOSTG-I11-PUB.cis Mg0/RP0/CPU0/0      120        N/A            Fa0/28

Total entries displayed: 1

RP/0/RP0/CPU0:router#show lldp neighbors mgmtEth 0/RP0/CPU0/0
Fri Dec 15 20:30:54.736 UTC
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
```

```
SW-NOSTG-I11-PUB.cis Mg0/RP0/CPU0/0      120        N/A              Fa0/28

Total entries displayed: 1
```

CHAPTER 4

Configuring Ethernet OAM

This module describes the configuration of Ethernet Operations, Administration, and Maintenance (OAM) .

Feature History for Configuring Ethernet OAM

Release	Modification

Information About Configuring Ethernet OAM

To configure Ethernet OAM, you should understand the following concepts:

Ethernet Link OAM

Table 2: Feature History Table

Ethernet as a Metro Area Network (MAN) or a Wide Area Network (WAN) technology benefits greatly from the implementation of Operations, Administration and Maintenance (OAM) features. Ethernet link OAM features allow Service Providers to monitor the quality of the connections on a MAN or WAN. Service providers can monitor specific events, . Ethernet link OAM operates on a single, physical link and it can be configured to monitor either side or both sides of that link.

Ethernet link OAM can be configured in the following ways:

- A Link OAM profile can be configured, and this profile can be used to set the parameters for multiple interfaces.
- Link OAM can be configured directly on an interface.

  When an interface is also using a link OAM profile, specific parameters that are set in the profile can be overridden by configuring a different value directly on the interface.

An Ethernet Link OAM profile simplifies the process of configuring EOAM features on multiple interfaces. An Ethernet OAM profile, and all of its features, can be referenced by other interfaces, allowing other interfaces to inherit the features of that Ethernet OAM profile.

Individual Ethernet link OAM features can be configured on individual interfaces without being part of a profile. In these cases, the individually configured features always override the features in the profile.

The preferred method of configuring custom EOAM settings is to create an EOAM profile in Ethernet configuration mode and then attach it to an individual interface or to multiple interfaces.

When an EOAM packet is received on any one of the AC interfaces on which EOAM is not configured, the AC interface multicasts the received EOAM packets to other AC interfaces that are part of EVPN-BD to reach the peer. When an EOAM is enabled on the bundle member in the peer, it punts the packet to the CPU in the peer. Also, the EOAM flaps the bundle member as the local or remote Key of the received EOAM does not match.

These standard Ethernet Link OAM features are supported on the router:

Neighbor Discovery

Neighbor discovery enables each end of a link to learn the OAM capabilities of the other end and establish an OAM peer relationship. Each end also can require that the peer have certain capabilities before it will establish a session. You can configure certain actions to be taken if there is a capabilities conflict or if a discovery process times out, using the **action capabilities-conflict** or **action discovery-timeout** commands.

EFD

Ethernet Fault Detection (EFD) is a mechanism that allows Ethernet OAM protocols, such as CFM, to control the `line protocol` state of an interface.

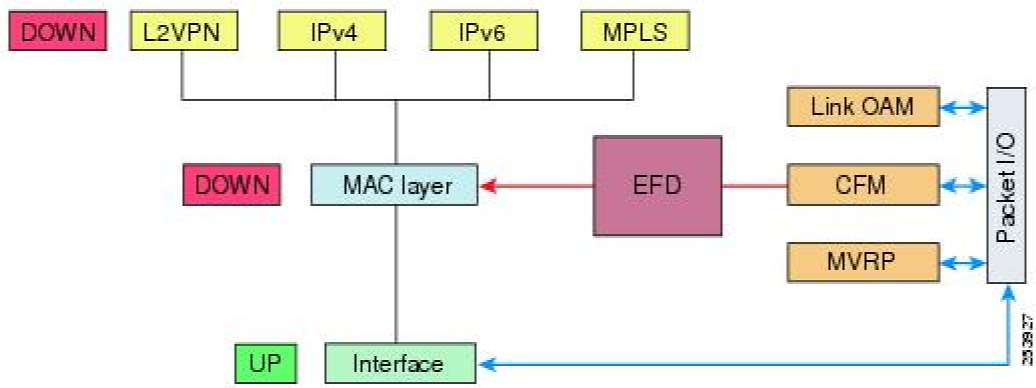
Unlike many other interface types, Ethernet interfaces do not have a line protocol, whose state is independent from that of the interface. For Ethernet interfaces, this role is handled by the physical-layer Ethernet protocol itself, and therefore if the interface is physically up, then it is available 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 traffic flow, 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.

This 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.

Figure 1: CFM Error Detection and EFD Trigger



MIB Retrieval

MIB retrieval enables an OAM peer on one side of an interface to get the MIB variables from the remote side of the link. The MIB variables that are retrieved from the remote OAM peer are READ ONLY.

SNMP Traps

SNMP traps can be enabled or disabled on an Ethernet OAM interface.

Ethernet CFM

Ethernet Connectivity Fault Management (CFM) is a service-level OAM protocol that provides tools for monitoring and troubleshooting end-to-end Ethernet services per VLAN. This includes proactive connectivity monitoring, fault verification, and fault isolation. CFM uses standard Ethernet frames and can be run on any physical media that is capable of transporting Ethernet service frames. Unlike most other Ethernet protocols which are restricted to a single physical link, CFM frames can transmit across the entire end-to-end Ethernet network.

Note Enable a maximum of 32 VLAN ranges per NPU. Else, when you reload the device, all CFM sessions over the 802.1Q VLAN interface might go down. Also, the corresponding bundle interface might go down. If more than 32 VLAN ranges exist on an NPU, remove the additional VLAN ranges and reload the device to address the issue.

CFM is defined in two standards:

- IEEE 802.1ag—Defines the core features of the CFM protocol.
- ITU-T Y.1731—Redefines, but maintains compatibility with the features of IEEE 802.1ag, and defines some additional features.

Ethernet CFM supports these functions of ITU-T Y.1731:

- ETH-CC, ETH-RDI, ETH-LB, ETH-LT—These are equivalent to the corresponding features defined in IEEE 802.1ag.

Note The Linktrace responder procedures defined in IEEE 802.1ag are used rather than the procedures defined in Y.1731; however, these are interoperable.

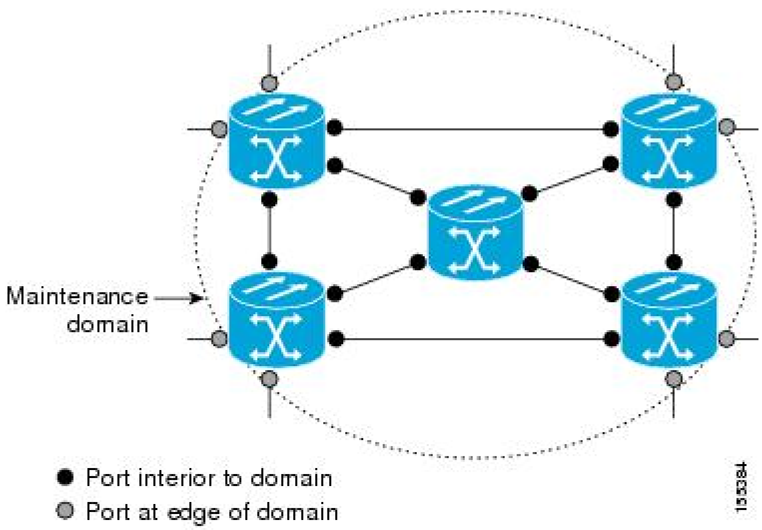
- ETH-AIS—The reception of ETH-LCK messages is also supported.

To understand how the CFM maintenance model works, you need to understand these concepts and features:

Maintenance Domains

A maintenance domain describes 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 interfaces internal to it and at its boundary, as shown in this figure.

Figure 2: CFM Maintenance Domain



A maintenance domain is defined by the bridge ports that are provisioned within it. Domains are assigned maintenance levels, in the range of 0 to 7, by the administrator. The level of the domain is useful in defining the hierarchical relationships of multiple domains.

CFM maintenance domains allow different organizations to use CFM in the same network, but independently. For example, consider a service provider who offers a service to a customer, and to provide that service, they use two other operators in segments of the network. In this environment, CFM can be used in the following ways:

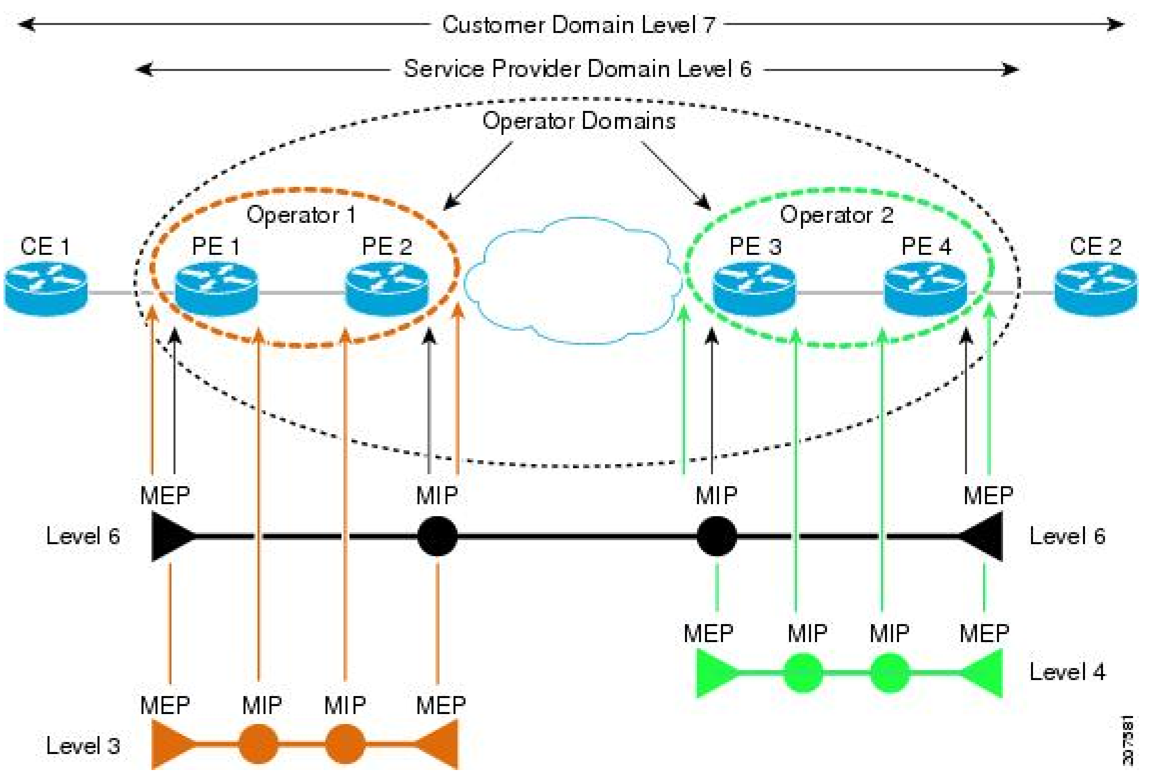
- The customer can use CFM between their CE devices, to verify and manage connectivity across the whole network.
- The service provider can use CFM between their PE devices, to verify and manage the services they are providing.
- Each operator can use CFM within their operator network, to verify and manage connectivity within their network.

Each organization uses a different CFM maintenance domain.

This figure shows an example of the different levels of maintenance domains in a network.

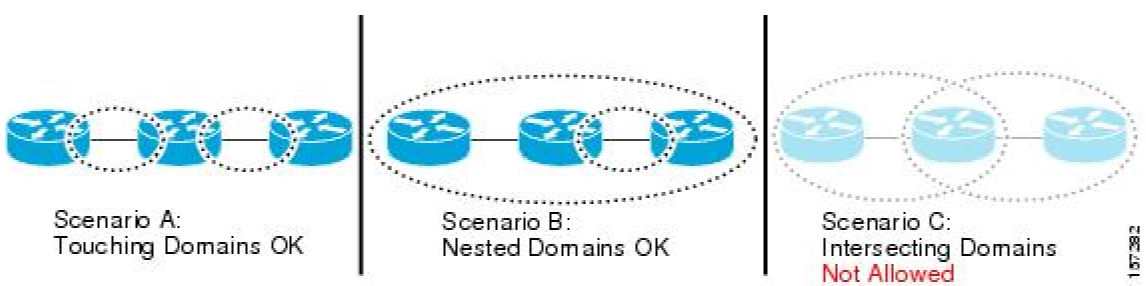
Note In CFM diagrams, the conventions are that triangles represent MEPs, pointing in the direction that the MEP sends CFM frames, and circles represent MIPs.

Figure 3: Different CFM Maintenance Domains Across a Network



To ensure that the CFM frames for each domain do not interfere with each other, each domain is assigned a maintenance level, between 0 and 7. Where domains are nested, as in this example, the encompassing domain must have a higher level than the domain it encloses. In this case, the domain levels must be negotiated between the organizations involved. The maintenance level is carried in all CFM frames that relate to that domain.

CFM maintenance domains may touch or nest, but cannot intersect. This figure illustrates the supported structure for touching and nested domains, and the unsupported intersection of domains.



Services

A CFM service allows an organization to partition its CFM maintenance domain, according to the connectivity within the network. For example, if the network is divided into a number of virtual LANs (VLANs), a CFM service is created for each of these. CFM can then operate independently in each service. It is important that the CFM services match the network topology, so that CFM frames relating to one service cannot be received in a different service. For example, a service provider may use a separate CFM service for each of their customers, to verify and manage connectivity between that customer's end points.

A CFM service is always associated with the maintenance domain that it operates within, and therefore with that domain's maintenance level. All CFM frames relating to the service carry the maintenance level of the corresponding domain.

Note CFM Services are referred to as *Maintenance Associations* in IEEE 802.1ag and as *Maintenance Entity Groups* in ITU-T Y.1731.

Maintenance Points

A CFM Maintenance Point (MP) is an instance of a particular CFM service on a specific interface. CFM only operates on an interface if there is a CFM maintenance point on the interface; otherwise, CFM frames are forwarded transparently through the interface.

A maintenance point is always associated with a particular CFM service, and therefore with a particular maintenance domain at a particular level. Maintenance points generally only process CFM frames at the same level as their associated maintenance domain. Frames at a higher maintenance level are always forwarded transparently, while frames at a lower maintenance level are normally dropped. This helps enforce the maintenance domain hierarchy, and ensures that CFM frames for a particular domain cannot leak out beyond the boundary of the domain.

There are two types of MP:

- Maintenance End Points (MEPs)—Created at the edge of the domain. Maintenance end points (MEPs) are members of a particular service within a domain and are responsible for sourcing and sinking CFM frames. They periodically transmit continuity check messages and receive similar messages from other MEPs within their domain. They also transmit traceroute and loopback messages at the request of the administrator. MEPs are responsible for confining CFM messages within the domain.
- Maintenance Intermediate Points (MIPs)—Created in the middle of the domain. Unlike MEPS, MIPs do allow CFM frames at their own level to be forwarded.

MIP Creation

Unlike MEPs, MIPs are not explicitly configured on each interface. MIPs are created automatically according to the algorithm specified in the CFM 802.1ag standard. The algorithm, in brief, operates as follows for each interface:

- The bridge-domain or cross-connect for the interface is found, and all services associated with that bridge-domain or cross-connect are considered for MIP auto-creation.
- The level of the highest-level MEP on the interface is found. From among the services considered above, the service in the domain with the lowest level that is higher than the highest MEP level is selected. If there are no MEPs on the interface, the service in the domain with the lowest level is selected.

- The MIP auto-creation configuration (**mip auto-create** command) for the selected service is examined to determine whether a MIP should be created.

> **Note** Configuring a MIP auto-creation policy for a service does not guarantee that a MIP will automatically be created for that service. The policy is only considered if that service is selected by the algorithm first.

MEP and CFM Processing Overview

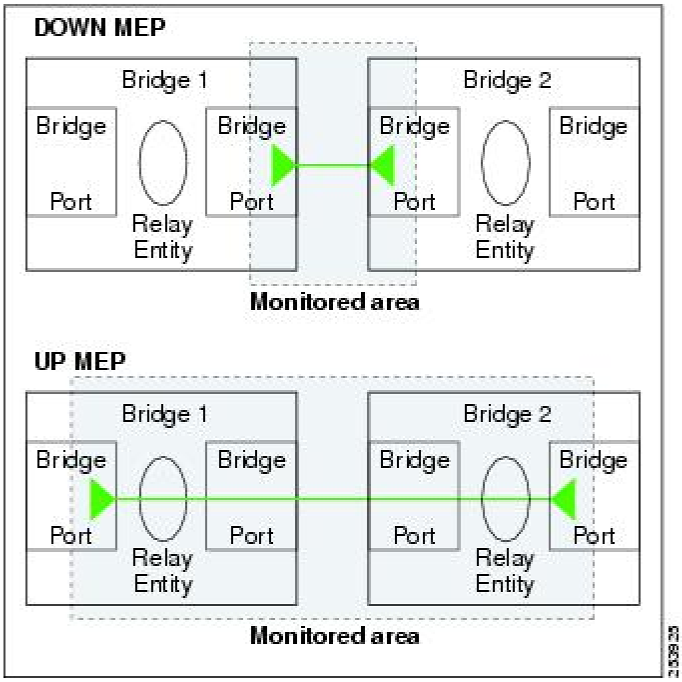
The boundary of a domain is an interface, rather than a bridge or host. Therefore, MEPs can be sub-divided into two categories:

- Down MEPs—Send CFM frames from the interface where they are configured, and process CFM frames received on that interface. Down MEPs transmit AIS messages upward (toward the cross-connect).
- Up MEPs—Send frames into the bridge relay function, as if they had been received on the interface where the MEP is configured. They process CFM frames that have been received on other interfaces, and have been switched through the bridge relay function as if they are going to be sent out of the interface where the MEP is configured. Up MEPs transmit AIS messages downward (toward the wire). However, AIS packets are only sent when there is a MIP configured on the same interface as the MEP and at the level of the MIP.

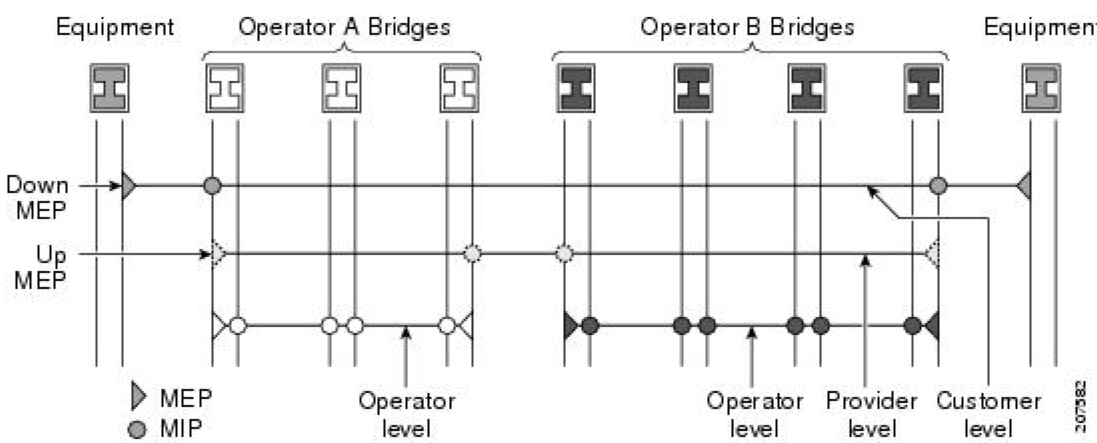
> **Note**
> - The terms *Down MEP* and *Up MEP* are defined in the IEEE 802.1ag and ITU-T Y.1731 standards, and refer to the direction that CFM frames are sent from the MEP. The terms should not be confused with the operational status of the MEP.
> - The router only supports the “Down MEP level < Up MEP level” configuration.

This figure illustrates the monitored areas for Down and Up MEPs.

Figure 4: Monitored Areas for Down and Up MEPs



This figure shows maintenance points at different levels. Because domains are allowed to nest but not intersect (see), a MEP at a low level always corresponds with a MEP or MIP at a higher level. In addition, only a single MIP is allowed on any interface—this is generally created in the lowest domain that exists at the interface and that does not have a MEP.



MIPs and Up MEPs can only exist on switched (Layer 2) interfaces, because they send and receive frames from the bridge relay function. Down MEPs can be created on switched (Layer 2) interfaces.

MEPs continue to operate normally if the interface they are created on is blocked by the Spanning Tree Protocol (STP); that is, CFM frames at the level of the MEP continue to be sent and received, according to the direction of the MEP. MEPs never allow CFM frames at the level of the MEP to be forwarded, so the STP block is maintained.

MIPs also continue to receive CFM frames at their level if the interface is STP blocked, and can respond to any received frames. However, MIPs do not allow CFM frames at the level of the MIP to be forwarded if the interface is blocked.

Note A separate set of CFM maintenance levels is created every time a VLAN tag is pushed onto the frame. Therefore, if CFM frames are received on an interface which pushes an additional tag, so as to "tunnel" the frames over part of the network, the CFM frames will not be processed by any MPs within the tunnel, even if they are at the same level. For example, if a CFM MP is created on an interface with an encapsulation that matches a single VLAN tag, any CFM frames that are received at the interface that have two VLAN tags will be forwarded transparently, regardless of the CFM level.

CFM Protocol Messages

The CFM protocol consists of a number of different message types, with different purposes. All CFM messages use the CFM EtherType, and carry the CFM maintenance level for the domain to which they apply.

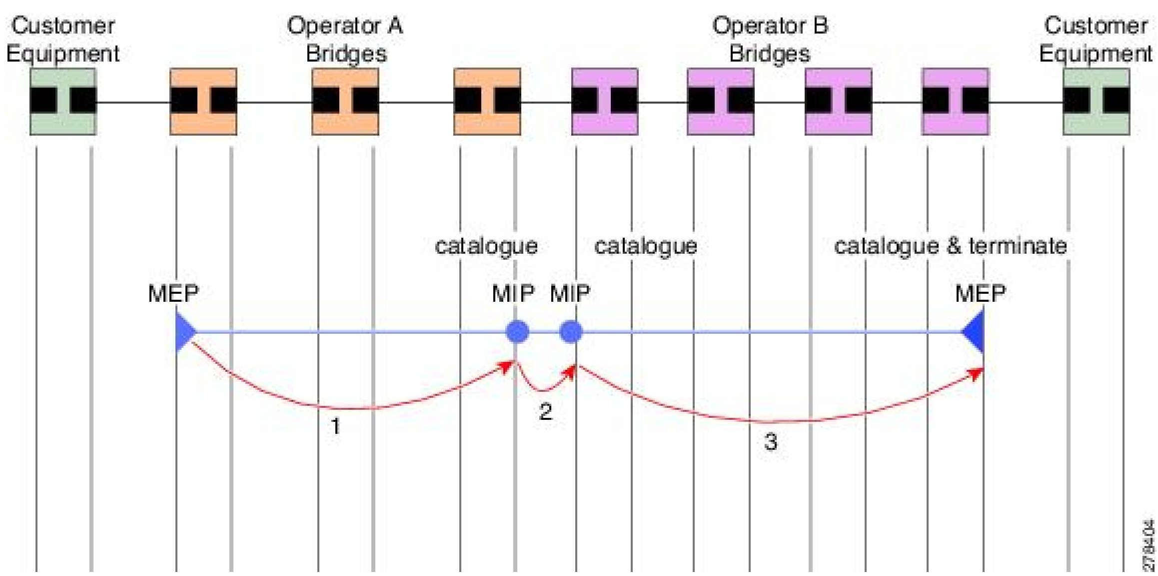
This section describes the following CFM messages:

Continuity Check (IEEE 802.1ag and ITU-T Y.1731)

Continuity Check Messages (CCMs) are "heartbeat" messages exchanged periodically between all the MEPs in a service. Each MEP sends out multicast CCMs, and receives CCMs from all the other MEPs in the service—these are referred to as *peer MEPs*. This allows each MEP to discover its peer MEPs, and to verify that there is connectivity between them.

MIPs also receive CCMs. MIPs use the information to build a MAC learning database that is used when responding to Linktrace. For more information about Linktrace, see the Linktrace (IEEE 802.1ag and ITU-T Y.1731).

Figure 5: Continuity Check Message Flow



All the MEPs in a service must transmit CCMs at the same interval. IEEE 802.1ag defines 7 possible intervals that can be used:

- 3.3ms
- 10ms
- 100ms
- 1s
- 10s
- 1 minute

A MEP detects a loss of connectivity with one of its peer MEPs when some number of CCMs have been missed. This occurs when sufficient time has passed during which a certain number of CCMs were expected, given the CCM interval. This number is called the *loss threshold*, and is usually set to 3.

CFM is supported only on interfaces which have Layer 2 transport feature enabled.

Maintenance Association Identifier (MAID)

CCM messages carry a variety of information that allows different defects to be detected in the service. This information includes:

- A configured identifier for the domain of the transmitting MEP. This is referred to as the Maintenance Domain Identifier (MDID).
- A configured identifier for the service of the transmitting MEP. This is referred to as the Short MA Name (SMAN). Together, the MDID and the SMAN make up the Maintenance Association Identifier (MAID). The MAID must be configured identically on every MEP in the service.
- These are restrictions on the type of MAID that are supported for sessions with time interval of less than 1 minute. The MAID supports two types of formats on offloaded MEPs:
    - No Domain Name Format
        - MD Name Format = 1-NoDomainName
        - Short MA Name Format = 3 - 2 bytes integer value
        - Short MA NAme Length = 2 - fixed length
        - Short MA Name = 2 bytes of integer
    - 1731 Maid Format
        - MD Name Format = 1-NoDomainName
        - MA Name Format(MEGID Format) = 32
        - MEGID Length = 13 - fixed length
        - MEGID(ICCCode) = 6 Bytes
        - MEGID(UMC) = 7 Bytes
        - ITU Carrier Code (ICC) - Number of different configurable ICC code - 15 (for each NPU)
        - Unique MEG ID Code (UMC) - 4

Maintenance Association Identifier (MAID) comprises of the Maintenance Domain Identifier (MDID) and Short MA Name (SMAN).

MDID **only** supports **null** value and SMAN supports ITU Carrier Code (ICC) or a numerical. No other values are supported.

An example for configuring domain ID null is: **ethernet cfm domain SMB level 3 id null**

An example for configuring SMAN is: **ethernet cfm domain SMB level 3 id null service 901234AB xconnect group 99999 p2p 99999 id number 1**

The following table summarizes the supported values and parameters for MDID and SMAN. This table only details the MAID restriction on the hardware offload feature. There is no MAID restriction for software offload or non-offloaded MEPs.

Format	MDID	SMAN	Support	Comment
	No	2 byte integer	Yes	Up to 2000 entries
	No	13 bytes ICCCode (6 bytes) and UMC (7 bytes)	Yes	Up to 15 unique ICC Up to 4K UMC values
48 bytes string based	1-48 bytes of MDID and SMAN		No	Most commonly used

- A configured numeric identifier for the MEP (the MEP ID). Each MEP in the service must be configured with a different MEP ID.
- Dynamic Remote MEPs are not supported for MEPs with less than 1min interval. You must configure MEP CrossCheck for all such MEPS.
- Sequence numbering is not supported for MEPs with less than 1 minute interval.
- In a Remote Defect Indication (RDI), each MEP includes this in the CCMs it is sending, if it has detected a defect relating to the CCMs it is receiving. This notifies all the MEPs in the service that a defect has been detected somewhere in the service.
- The interval at which CCMs are being transmitted.
- CCM Tx/Rx statistics counters are not supported for MEPs with less than1 minute intervals.
- Sender TLV and Cisco Proprietary TLVs are not supported for MEPs with less than 1min intervals.
- The status of the interface where the MEP is operating—for example, whether the interface is up, down, STP blocked, and so on.

Note The status of the interface (up/down) should not be confused with the direction of any MEPs on the interface (Up MEPs/Down MEPs).

These defects can be detected from received CCMs:

- Interval mismatch—The CCM interval in the received CCM does not match the interval that the MEP is sending CCMs.
- Level mismatch—A MEP has received a CCM carrying a lower maintenance level than the MEPs own level.
- Loop—A CCM is received with the source MAC address equal to the MAC address of the interface where the MEP is operating.
- Configuration error—A CCM is received with the same MEP ID as the MEP ID configured for the receiving MEP.
- Cross-connect—A CCM is received with an MAID that does not match the locally configured MAID. This generally indicates a VLAN misconfiguration within the network, such that CCMs from one service are leaking into a different service.
- Peer interface down—A CCM is received that indicates the interface on the peer is down.
- Remote defect indication—A CCM is received carrying a remote defect indication.

Note This defect does not cause the MEP to include a remote defect indication in the CCMs that it is sending.

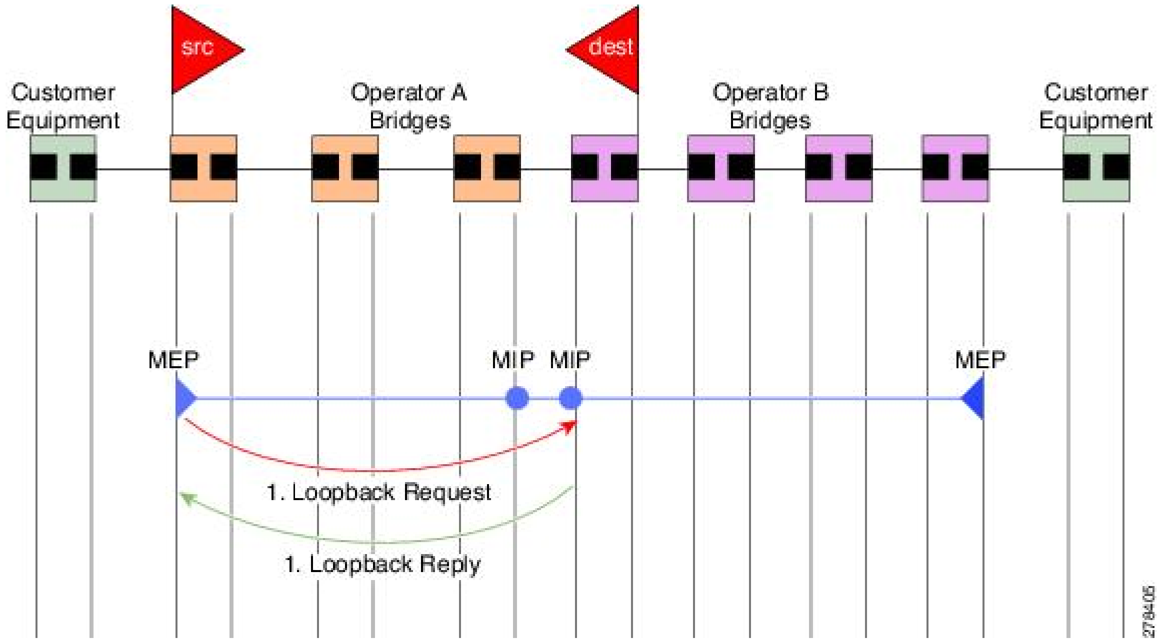
Out-of-sequence CCMs can also be detected by monitoring the sequence number in the received CCMs from each peer MEP. However, this is not considered a CCM defect.

Loopback (IEEE 802.1ag and ITU-T Y.1731)

Loopback Messages (LBM) and Loopback Replies (LBR) are used to verify connectivity between a local MEP and a particular remote MP. At the request of the administrator, a local MEP sends unicast LBMs to the remote MP. On receiving each LBM, the target maintenance point sends an LBR back to the originating MEP. Loopback indicates whether the destination is reachable or not—it does not allow hop-by-hop discovery of the path. It is similar in concept to an ICMP Echo (ping). Since loopback messages are destined for unicast addresses, they are forwarded like normal data traffic, while observing the maintenance levels. At each device that the loopback reaches, if the outgoing interface is known (in the bridge's forwarding database), then the frame is sent out on that interface. If the outgoing interface is not known, then the message is flooded on all interfaces.

This figure shows an example of CFM loopback message flow between a MEP and MIP.

Figure 6: Loopback Messages



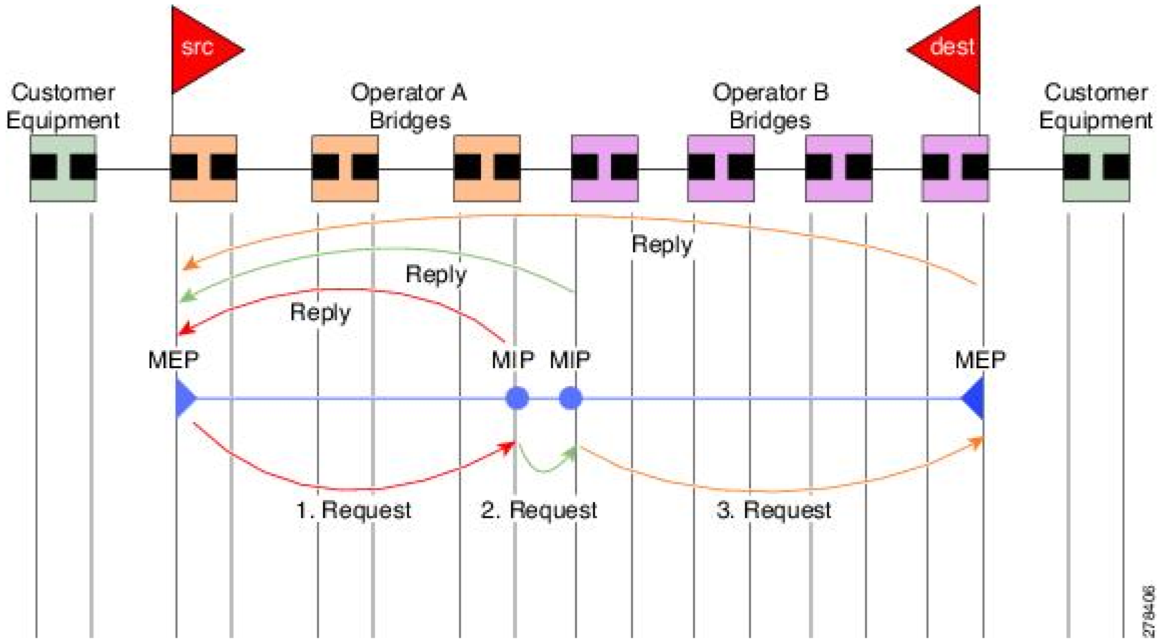
Loopback messages can be padded with user-specified data. This allows data corruption to be detected in the network. They also carry a sequence number which allows for out-of-order frames to be detected.

Linktrace (IEEE 802.1ag and ITU-T Y.1731)

Linktrace Messages (LTM) and Linktrace Replies (LTR) are used to track the path (hop-by-hop) to a unicast destination MAC address. At the request of the operator, a local MEP sends an LTM. Each hop where there is a maintenance point sends an LTR back to the originating MEP. This allows the administrator to discover connectivity data about the path. It is similar in concept to IP traceroute, although the mechanism is different. In IP traceroute, successive probes are sent, whereas CFM Linktrace uses a single LTM which is forwarded by each MP in the path. LTMs are multicast, and carry the unicast target MAC address as data within the frame. They are intercepted at each hop where there is a maintenance point, and either retransmitted or dropped to discover the unicast path to the target MAC address.

This figure shows an example of CFM linktrace message flow between MEPs and MIPs.

Figure 7: Linktrace Message Flow



The linktrace mechanism is designed to provide useful information even after a network failure. This allows it to be used to locate failures, for example after a loss of continuity is detected. To achieve this, each MP maintains a CCM Learning Database. This maps the source MAC address for each received CCM to the interface through which the CCM was received. It is similar to a typical bridge MAC learning database, except that it is based only on CCMs and it times out much more slowly—on the order of days rather than minutes.

Note In IEEE 802.1ag, the CCM Learning Database is referred to as the MIP CCM Database. However, it applies to both MIPs and MEPs.

In IEEE 802.1ag, when an MP receives an LTM message, it determines whether to send a reply using the following steps:

1. The target MAC address in the LTM is looked up in the bridge MAC learning table. If the MAC address is known, and therefore the egress interface is known, then an LTR is sent.
2. If the MAC address is not found in the bridge MAC learning table, then it is looked up in the CCM learning database. If it is found, then an LTR is sent.
3. If the MAC address is not found, then no LTR is sent (and the LTM is not forwarded).

If the target MAC has never been seen previously in the network, the linktrace operation will not produce any results.

> **Note** IEEE 802.1ag and ITU-T Y.1731 define slightly different linktrace mechanisms. In particular, the use of the CCM learning database and the algorithm described above for responding to LTM messages are specific to IEEE 802.1ag. IEEE 802.1ag also specifies additional information that can be included in LTRs. Regardless of the differences, the two mechanisms are interoperable.

Configurable Logging

CFM supports logging of various conditions to syslog. Logging can be enabled independently for each service, and when the following conditions occur:

- New peer MEPs are detected, or loss of continuity with a peer MEP occurs.
- Changes to the CCM defect conditions are detected.
- Cross-check “missing” or “unexpected” conditions are detected.
- AIS condition detected (AIS messages received) or cleared (AIS messages no longer received).
- EFD used to shut down an interface, or bring it back up.

Flexible VLAN Tagging for CFM

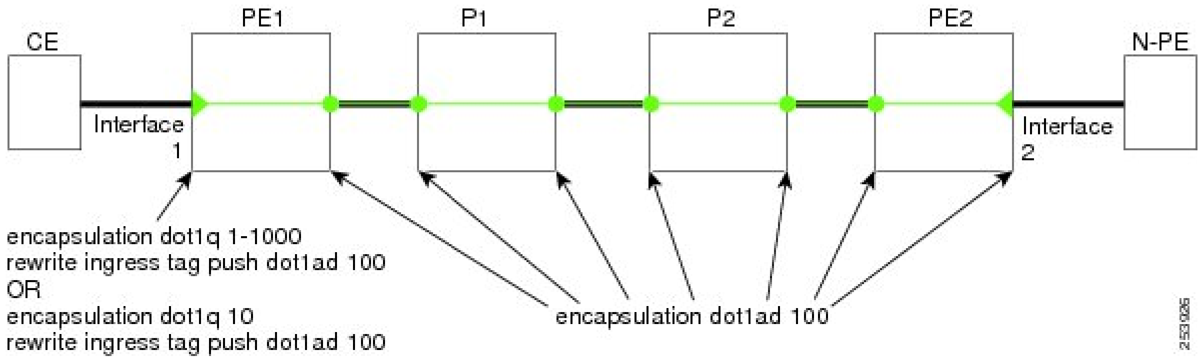
The Flexible VLAN Tagging for CFM feature ensures that CFM packets are sent with the right VLAN tags so that they are appropriately handled as a CFM packet by the remote device. When packets are received by an edge router, they are treated as either CFM packets or data packets, depending on the number of tags in the header. The system differentiates between CFM packets and data packets based on the number of tags in the packet, and forwards the packets to the appropriate paths based on the number of tags in the packet.

CFM frames are normally sent with the same VLAN tags as the corresponding customer data traffic on the interface, as defined by the configured encapsulation and tag rewrite operations. Likewise, received frames are treated as CFM frames if they have the correct number of tags as defined by the configured encapsulation and tag rewrite configuration, and are treated as data frames (that is, they are forwarded transparently) if they have more than this number of tags.

In most cases, this behavior is as desired, since the CFM frames are then treated in exactly the same way as the data traffic flowing through the same service. However, in a scenario where multiple customer VLANs are multiplexed over a single multipoint provider service (for example, N:1 bundling), a different behavior might be desirable.

This figure shows an example of a network with multiple VLANS using CFM.

Figure 8: Service Provider Network With Multiple VLANs and CFM



This figure shows a provider's access network, where the S-VLAN tag is used as the service delimiter. PE1 faces the customer, and PE2 is at the edge of the access network facing the core. N:1 bundling is used, so the interface encapsulation matches a range of C-VLAN tags. This could potentially be the full range, resulting in all:1 bundling. There is also a use case where only a single C-VLAN is matched, but the S-VLAN is nevertheless used as the service delimiter—this is more in keeping with the IEEE model, but limits the provider to 4094 services.

CFM is used in this network with a MEP at each end of the access network, and MIPs on the boxes within the network (if it is native Ethernet). In the normal case, CFM frames are sent by the up MEP on PE1 with two VLAN tags, matching the customer data traffic. This means that at the core interfaces and at the MEP on PE2, the CFM frames are forwarded as if they were customer data traffic, since these interfaces match only on the S-VLAN tag. So, the CFM frames sent by the MEP on PE1 are not seen by any of the other MPs.

Flexible VLAN tagging changes the encapsulation for CFM frames that are sent and received at Up MEPs. Flexible VLAN tagging allows the frames to be sent from the MEP on PE1 with just the S-VLAN tag that represents the provider service. If this is done, the core interfaces will treat the frames as CFM frames and they will be seen by the MIPs and by the MEP on PE2. Likewise, the MEP on PE1 should handle received frames with only one tag, as this is what it will receive from the MEP on PE2.

To ensure that CFM packets from Up MEPs are routed to the appropriate paths successfully, tags may be set to a specific number in a domain service, using the **tags** command. Currently, tags can only be set to one (1).

How to Configure Ethernet OAM

This section provides these configuration procedures:

Configuring Ethernet Link OAM

Custom EOAM settings can be configured and shared on multiple interfaces by creating an EOAM profile in Ethernet configuration mode and then attaching the profile to individual interfaces. The profile configuration does not take effect until the profile is attached to an interface. After an EOAM profile is attached to an interface, individual EOAM features can be configured separately on the interface to override the profile settings when desired.

This section describes how to configure an EOAM profile and attach it to an interface in these procedures:

Configuring an Ethernet OAM Profile

Perform these steps to configure an Ethernet OAM profile.

SUMMARY STEPS

1. **configure**
2. **ethernet oam profile** *profile-name*
3. **link-monitor**
4. **symbol-period window** *window*
5. **symbol-period threshold low** *threshold* **high** *threshold* **symbol-period threshold** { **ppm** [**low** *threshold*] [**high** *threshold*] | **symbols** [**low** *threshold* [**thousand** | **million** | **billion**]] [**high** *threshold* [**thousand** | **million** | **billion**]]}
6. **frame window** *window*
7. **frame threshold low** *threshold* **high** *threshold*
8. **frame-period window** *window*
9. **frame-period threshold lowthreshold**...

   frame-period threshold low*threshold* **high** *threshold* **frame-period threshold** { **ppm** [**low** *threshold*] [**high** *threshold*] | **frames** [**low** *threshold* [**thousand** | **million** | **billion**]] [**high** *threshold* [**thousand** | **million** | **billion**]]}
10. **frame-seconds window** *window*
11. **frame-seconds threshold low** *threshold* **high** *threshold*
12. **exit**
13. **mib-retrieval**
14. **connection timeout** <*timeout*>
15. **hello-interval** {**100ms**|**1s**}
16. **mode** {**active**|**passive**}
17. **require-remote mode** {**active**|**passive**}
18. **require-remote mib-retrieval**
19. **action capabilities-conflict** {**disable** | **efd** | **error-disable-interface**}
20. **action critical-event** {**disable** | **error-disable-interface**}
21. **action discovery-timeout** {**disable** | **efd** | **error-disable-interface**}
22. **action dying-gasp** {**disable** | **error-disable-interface**}
23. **action high-threshold** {**error-disable-interface** | **log**}
24. **action session-down** {**disable** | **efd** | **error-disable-interface**}
25. **action session-up disable**
26. **action uni-directional link-fault** {**disable** | **efd** | **error-disable-interface**}
27. **action wiring-conflict** {**disable** | **efd** | **log**}
28. **uni-directional link-fault detection**
29. **commit**
30. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# configure terminal`	
Step 2	**ethernet oam profile** *profile-name* **Example:** `RP/0/RP0/CPU0:router(config)# ethernet oam profile Profile_1`	Creates a new Ethernet Operations, Administration and Maintenance (OAM) profile and enters Ethernet OAM configuration mode.
Step 3	**link-monitor** **Example:** `RP/0/RP0/CPU0:router(config-eoam)# link-monitor`	Enters the Ethernet OAM link monitor configuration mode.
Step 4	**symbol-period window** *window* **Example:** `RP/0/RP0/CPU0:router(config-eoam-lm)# symbol-period window 60000`	(Optional) Configures the window size (in milliseconds) for an Ethernet OAM symbol-period error event. The IEEE 802.3 standard defines the window size as a number of symbols rather than a time duration. These two formats can be converted either way by using a knowledge of the interface speed and encoding. The range is 1000 to 60000. The default value is 1000.
Step 5	**symbol-period threshold low** *threshold* **high** *threshold* **symbol-period threshold** { **ppm** [**low** *threshold*] [**high** *threshold*] | **symbols** [**low** *threshold* [**thousand** | **million** | **billion**]] [**high** *threshold* [**thousand** | **million** | **billion**]]} **Example:** `RP/0/RP0/CPU0:router(config-eoam-lm)# symbol-period threshold ppm low 1 high 1000000`	(Optional) Configures the thresholds (in symbols) that trigger an Ethernet OAM symbol-period error event. The high threshold is optional and is configurable only in conjunction with the low threshold. The range is 1 to 1000000. The default low threshold is 1.
Step 6	**frame window** *window* **Example:** `RP/0/RP0/CPU0:router(config-eoam-lm)# frame window 6000`	(Optional) Configures the frame window size (in milliseconds) of an OAM frame error event. The range is from 1000 to 60000. The default value is 1000.
Step 7	**frame threshold low** *threshold* **high** *threshold* **Example:** `RP/0/RP0/CPU0:router(config-eoam-lm)# frame threshold low 10000000 high 60000000`	(Optional) Configures the thresholds (in symbols) that triggers an Ethernet OAM frame error event. The high threshold is optional and is configurable only in conjunction with the low threshold. The range is from 0 to 60000000. The default low threshold is 1.
Step 8	**frame-period window** *window* **Example:**	(Optional) Configures the window size (in milliseconds) for an Ethernet OAM frame-period error event. The IEEE

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-eoam-lm)# frame-period window 60000` `RP/0/RP0/CPU0:router(config-eoam-lm)# frame-period window milliseconds 60000`	802.3 standard defines the window size as number of frames rather than a time duration. These two formats can be converted either way by using a knowledge of the interface speed. Note that the conversion assumes that all frames are of the minimum size. The range is from 1000 to 60000. The default value is 1000. **Note** The only accepted values are multiples of the line cardinterface module-specific polling interval, that is, 1000 milliseconds for most line cardsinterface modules.
Step 9	**frame-period threshold low***threshold* **high** *threshold* **frame-period threshold** { **ppm** [**low** *threshold*] [**high** *threshold*] | **frames** [**low** *threshold* [**thousand** | **million** | **billion**]] [**high** *threshold* [**thousand** | **million** | **billion**]]} **Example:** `RP/0/RP0/CPU0:router(config-eoam-lm)# frame-period threshold ppm low 100 high 1000000`	(Optional) Configures the thresholds (in errors per million frames) that trigger an Ethernet OAM frame-period error event. The frame period window is defined in the IEEE specification as a number of received frames, in our implementation it is x milliseconds. The high threshold is optional and is configurable only in conjunction with the low threshold. The range is from 1 to 1000000. The default low threshold is 1. To obtain the number of frames, the configured time interval is converted to a window size in frames using the interface speed. For example, for a 1Gbps interface, the IEEE defines minimum frame size as 512 bits. So, we get a maximum of approximately 1.5 million frames per second. If the window size is configured to be 8 seconds (8000ms) then this would give us a Window of 12 million frames in the specification's definition of Errored Frame Window. The thresholds for frame-period are measured in errors per million frames. Hence, if you configure a window of 8000ms (that is a window of 12 million frames) and a high threshold of 100, then the threshold would be crossed if there are 1200 errored frames in that period (that is, 100 per million for 12 million).
Step 10	**frame-seconds window** *window* **Example:** `RP/0/RP0/CPU0:router(config-eoam-lm)# frame-seconds window 900000`	(Optional) Configures the window size (in milliseconds) for the OAM frame-seconds error event. The range is 10000 to 900000. The default value is 60000. **Note** The only accepted values are multiples of the line cardinterface module-specific polling interval, that is, 1000 milliseconds for most line cardsinterface modules.

	Command or Action	Purpose
Step 11	**frame-seconds threshold low** *threshold* **high** *threshold* **Example:** `RP/0/RP0/CPU0:router(config-eoam-lm)# frame-seconds threshold low 3 high 900`	(Optional) Configures the thresholds (in seconds) that trigger a frame-seconds error event. The high threshold value can be configured only in conjunction with the low threshold value. The range is 1 to 900 The default value is 1.
Step 12	**exit** **Example:** `RP/0/RP0/CPU0:router(config-eoam-lm)# exit`	Exits back to Ethernet OAM mode.
Step 13	**mib-retrieval** **Example:** `RP/0/RP0/CPU0:router(config-eoam)# mib-retrieval`	Enables MIB retrieval in an Ethernet OAM profile or on an Ethernet OAM interface.
Step 14	**connection timeout** *<timeout>* **Example:** `RP/0/RP0/CPU0:router(config-eoam)# connection timeout 30`	Configures the connection timeout period for an Ethernet OAM session. as a multiple of the hello interval. The range is 2 to 30. The default value is 5.
Step 15	**hello-interval** {**100ms**|**1s**} **Example:** `RP/0/RP0/CPU0:router(config-eoam)# hello-interval 100ms`	Configures the time interval between hello packets for an Ethernet OAM session. The default is 1 second (**1s**).
Step 16	**mode** {**active**|**passive**} **Example:** `RP/0/RP0/CPU0:router(config-eoam)# mode passive`	Configures the Ethernet OAM mode. The default is active.
Step 17	**require-remote mode** {**active**|**passive**} **Example:** `RP/0/RP0/CPU0:router(config-eoam)# require-remote mode active`	Requires that active mode or passive mode is configured on the remote end before the OAM session becomes active.
Step 18	**require-remote mib-retrieval** **Example:** `RP/0/RP0/CPU0:router(config-eoam)# require-remote mib-retrieval`	Requires that MIB-retrieval is configured on the remote end before the OAM session becomes active.

	Command or Action	Purpose
Step 19	**action capabilities-conflict** {**disable** | **efd** | **error-disable-interface**} **Example:** `RP/0/RP0/CPU0:router(config-eoam)# action capabilities-conflict efd`	Specifies the action that is taken on an interface when a capabilities-conflict event occurs. The default action is to create a syslog entry. **Note** • If you change the default, the **log** keyword option is available in Interface Ethernet OAM configuration mode to override the profile setting and log the event for the interface when it occurs.
Step 20	**action critical-event** {**disable** | **error-disable-interface**} **Example:** `RP/0/RP0/CPU0:router(config-eoam)# action critical-event error-disable-interface`	Specifies the action that is taken on an interface when a critical-event notification is received from the remote Ethernet OAM peer. The default action is to create a syslog entry. **Note** • If you change the default, the **log** keyword option is available in Interface Ethernet OAM configuration mode to override the profile setting and log the event for the interface when it occurs.
Step 21	**action discovery-timeout** {**disable** | **efd** | **error-disable-interface**} **Example:** `RP/0/RP0/CPU0:router(config-eoam)# action discovery-timeout efd`	Specifies the action that is taken on an interface when a connection timeout occurs. The default action is to create a syslog entry. **Note** • If you change the default, the **log** keyword option is available in Interface Ethernet OAM configuration mode to override the profile setting and log the event for the interface when it occurs.
Step 22	**action dying-gasp** {**disable** | **error-disable-interface**} **Example:** `RP/0/RP0/CPU0:router(config-eoam)# action dying-gasp error-disable-interface`	Specifies the action that is taken on an interface when a dying-gasp notification is received from the remote Ethernet OAM peer. The default action is to create a syslog entry. **Note** • If you change the default, the **log** keyword option is available in Interface Ethernet OAM configuration mode to override the profile setting and log the event for the interface when it occurs.
Step 23	**action high-threshold** {**error-disable-interface** | **log**} **Example:** `RP/0/RP0/CPU0:router(config-eoam)# action high-threshold error-disable-interface`	Specifies the action that is taken on an interface when a high threshold is exceeded. The default is to take no action when a high threshold is exceeded.

	Command or Action	Purpose
		Note • If you change the default, the **disable** keyword option is available in Interface Ethernet OAM configuration mode to override the profile setting and take no action at the interface when the event occurs.
Step 24	**action session-down** {**disable** | **efd** | **error-disable-interface**} **Example:** `RP/0/RP0/CPU0:router(config-eoam)# action session-down efd`	Specifies the action that is taken on an interface when an Ethernet OAM session goes down. **Note** • If you change the default, the **log** keyword option is available in Interface Ethernet OAM configuration mode to override the profile setting and log the event for the interface when it occurs.
Step 25	**action session-up disable** **Example:** `RP/0/RP0/CPU0:router(config-eoam)# action session-up disable`	Specifies that no action is taken on an interface when an Ethernet OAM session is established. The default action is to create a syslog entry. **Note** • If you change the default, the **log** keyword option is available in Interface Ethernet OAM configuration mode to override the profile setting and log the event for the interface when it occurs.
Step 26	**action uni-directional link-fault** {**disable** | **efd** | **error-disable-interface**}	Specifies the action that is taken on an interface when a link-fault notification is received from the remote Ethernet OAM peer. The default action is to create a syslog entry. **Note** • If you change the default, the **log** keyword option is available in Interface Ethernet OAM configuration mode to override the profile setting and log the event for the interface when it occurs.
Step 27	**action wiring-conflict** {**disable** | **efd** | **log**} **Example:** `RP/0/RP0/CPU0:router(config-eoam)# action session-down efd`	Specifies the action that is taken on an interface when a wiring-conflict event occurs. The default is to put the interface into error-disable state. **Note** • If you change the default, the **error-disable-interface** keyword option is available in Interface Ethernet OAM configuration mode to override the profile setting and put the interface into error-disable state when the event occurs.

	Command or Action	Purpose
Step 28	**uni-directional link-fault detection** **Example:** `RP/0/RP0/CPU0:router(config-eoam)# uni-directional link-fault detection`	Enables detection of a local, unidirectional link fault and sends notification of that fault to an Ethernet OAM peer.
Step 29	**commit** **Example:** `RP/0/RP0/CPU0:router(config-if)# commit`	Saves the configuration changes to the running configuration file and remains within the configuration session.
Step 30	**end** **Example:** `RP/0/RP0/CPU0:router(config-if)# end`	Ends the configuration session and exits to the EXEC mode.

Attaching an Ethernet OAM Profile to an Interface

Perform these steps to attach an Ethernet OAM profile to an interface:

SUMMARY STEPS

1. **configure**
2. **interface** [| **HundredGigE**| **TenGigE**] *interface-path-id*
3. **ethernet oam**
4. **profile** *profile-name*
5. **commit**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure terminal`	Enters global configuration mode.
Step 2	**interface** [| **HundredGigE**| **TenGigE**] *interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/0`	Enters interface configuration mode and specifies the Ethernet interface name and notation *rack/slot/module/port*.
Step 3	**ethernet oam** **Example:**	Enables Ethernet OAM and enters interface Ethernet OAM configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-if)# ethernet oam`	
Step 4	**profile** *profile-name* **Example:** `RP/0/RP0/CPU0:router(config-if-eoam)# profile Profile_1`	Attaches the specified Ethernet OAM profile (*profile-name*), and all of its configuration, to the interface.
Step 5	**commit** **Example:** `RP/0/RP0/CPU0:router(config-if)# commit`	Saves the configuration changes to the running configuration file and remains within the configuration session.
Step 6	**end** **Example:** `RP/0/RP0/CPU0:router(config-if)# end`	Ends the configuration session and exits to the EXEC mode.

Configuring Ethernet OAM at an Interface and Overriding the Profile Configuration

Using an EOAM profile is an efficient way of configuring multiple interfaces with a common EOAM configuration. However, if you want to use a profile but also change the behavior of certain functions for a particular interface, then you can override the profile configuration. To override certain profile settings that are applied to an interface, you can configure that command in interface Ethernet OAM configuration mode to change the behavior for that interface.

In some cases, only certain keyword options are available in interface Ethernet OAM configuration due to the default settings for the command. For example, without any configuration of the **action** commands, several forms of the command have a default behavior of creating a syslog entry when a profile is created and applied to an interface. Therefore, the **log** keyword is not available in Ethernet OAM configuration for these commands in the profile because it is the default behavior. However, the **log** keyword is available in Interface Ethernet OAM configuration if the default is changed in the profile configuration so you can retain the action of creating a syslog entry for a particular interface.

To see all of the default Ethernet OAM configuration settings, see the Verifying the Ethernet OAM Configuration.

To configure Ethernet OAM settings at an interface and override the profile configuration, perform these steps:

SUMMARY STEPS

1. **configure**
2. **interface** [HundredGigE | TenGigE] *interface-path-id*
3. **ethernet oam**
4. *interface-Ethernet-OAM-command*
5. **commit**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure terminal`	Enters global configuration mode.
Step 2	**interface** [HundredGigE | TenGigE] *interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/0`	Enters interface configuration mode and specifies the Ethernet interface name and notation *rack/slot/module/port*. **Note** • The example indicates an 8-port 10-Gigabit Ethernet interface in modular services card slot 1.
Step 3	**ethernet oam** **Example:** `RP/0/RP0/CPU0:router(config-if)# ethernet oam`	Enables Ethernet OAM and enters interface Ethernet OAM configuration mode.
Step 4	*interface-Ethernet-OAM-command* **Example:** `RP/0/RP0/CPU0:router(config-if-eoam)# action capabilities-conflict error-disable-interface`	Configures a setting for an Ethernet OAM configuration command and overrides the setting for the profile configuration, where *interface-Ethernet-OAM-command* is one of the supported commands on the platform in interface Ethernet OAM configuration mode.
Step 5	**commit** **Example:** `RP/0/RP0/CPU0:router(config-if)# commit`	Saves the configuration changes to the running configuration file and remains within the configuration session.
Step 6	**end** **Example:** `RP/0/RP0/CPU0:router(config-if)# end`	Ends the configuration session and exits to the EXEC mode.

Verifying the Ethernet OAM Configuration

Use the **show ethernet oam configuration** command to display the values for the Ethernet OAM configuration for a particular interface, or for all interfaces. The following example shows the default values for Ethernet OAM settings:

```
RP/0/RP0/CPU0:router# show ethernet oam configuration
Thu Aug  5 22:07:06.870 DST
GigabitEthernet0/0/0/0:
  Hello interval:                                          1s
  Mib retrieval enabled:                                    N
  Uni-directional link-fault detection enabled:             N
  Configured mode:                                     Active
  Connection timeout:                                       5
  Symbol period window:                                     0
```

```
Symbol period low threshold:                                    1
Symbol period high threshold:                                None
Frame window:                                                1000
Frame low threshold:                                            1
Frame high threshold:                                        None
Frame period window:                                         1000
Frame period low threshold:                                     1
Frame period high threshold:                                 None
Frame seconds window:                                       60000
Frame seconds low threshold:                                    1
Frame seconds high threshold:                                None
High threshold action:                                       None
Link fault action:                                            Log
Dying gasp action:                                            Log
Critical event action:                                        Log
Discovery timeout action:                                     Log
Capabilities conflict action:                                 Log
Wiring conflict action:                             Error-Disable
Session up action:                                            Log
Session down action:                                          Log
Require remote mode:                                       Ignore
Require remote MIB retrieval:                                   N
```

Configuring Ethernet CFM

Note CFM is not supported for the following:

- L3 Interfaces and Sub-Interfaces
- Bundle Member Ports
- EVPN-FXC
- Bridge Domain
- VPLS

Configuring a CFM Maintenance Domain

To configure a CFM maintenance domain, perform the following steps:

SUMMARY STEPS

1. **configure**
2. **ethernet cfm**
3. **traceroute cache hold-time** *minutes* **size** *entries*
4. **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]]
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ethernet cfm** **Example:** `RP/0/RP0/CPU0:router(config)# ethernet cfm`	Enters Ethernet Connectivity Fault Management (CFM) configuration mode.
Step 3	**traceroute cache hold-time** *minutes* **size** *entries* **Example:** `RP/0/RP0/CPU0:router(config-cfm)# traceroute cache hold-time 1 size 3000`	(Optional) Sets the maximum limit of traceroute cache entries or the maximum time limit to hold the traceroute cache entries. The default is 100 minutes and 100 entries.
Step 4	**domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] **Example:** `RP/0/RP0/CPU0:router(config-cfm)# domain Domain_One level 1 id string D1`	Creates and names a container for all domain configurations and enters CFM domain configuration mode. The level must be specified. The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn)# commit`	Saves configuration changes. • When you use the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Services for a CFM Maintenance Domain

You can configure up to 2000 CFM services for a maintenance domain. To configure services for a CFM maintenance domain, perform the following steps:

SUMMARY STEPS

1. **configure**
2. **ethernet cfm**
3. **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]]
4. **service** *service-name* {**down-meps | xconnect group** *xconnect-group-name* **m2mp | p2p** *xconnect-name*}[**id** [**icc-based** *icc-string umc-string*] | [[**number** *number*]
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ethernet cfm** **Example:** `RP/0/RP0/CPU0:router(config)# ethernet cfm`	Enters Ethernet CFM configuration mode.
Step 3	**domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] **Example:** `RP/0/RP0/CPU0:router(config-cfm)# domain Domain_One level 1 id string D1`	Creates and names a container for all domain configurations at a specified maintenance level, and enters CFM domain configuration mode. The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.
Step 4	**service** *service-name* {**down-meps | xconnect group** *xconnect-group-name* **m2mp | p2p** *xconnect-name*}[**id** [**icc-based** *icc-string umc-string*] | [[**number** *number*] **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn)# service ABC xconnect group X1 p2p ADB`	Configures and associates a service with the domain and enters CFM domain service configuration mode. You can specify that the service is used only for down MEPs, or associate the service with a bridge domain where MIPs and up MEPs will be created. The **id** sets the short MA name.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# commit`	Saves configuration changes. • When you use the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before`

	Command or Action	Purpose

		`exiting(yes/no/cancel)?`
		`[cancel]:`
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		• Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Enabling and Configuring Continuity Check for a CFM Service

To configure Continuity Check for a CFM service, complete the following steps:

SUMMARY STEPS

1. **configure**
2. **ethernet cfm**
3. **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]]
4. **service** *service-name* {**down-meps | xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id** [**icc-based** *icc-string umc-string*] | [[**number** *number*]
5. **continuity-check interval** *time* [**loss-threshold** *threshold*]
6. **continuity-check archive hold-time** *minutes*
7. **continuity-check loss auto-traceroute**
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ethernet cfm** **Example:** `RP/0/RP0/CPU0:router(config)# ethernet cfm`	Enters Ethernet Connectivity Fault Management (CFM) configuration mode.

	Command or Action	Purpose
Step 3	**domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] **Example:** `RP/0/RP0/CPU0:router(config-cfm)# domain Domain_One level 1 id string D1`	Creates and names a container for all domain configurations and enters the CFM domain configuration mode. The level must be specified. The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.
Step 4	**service** *service-name* {**down-meps | xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id** [**icc-based** *icc-string umc-string*] | [[**number** *number*] **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn)# service ABC xconnect group X1 p2p ADB`	Configures and associates a service with the domain and enters CFM domain service configuration mode. You can specify that the service is used only for down MEPs, or associate the service with a bridge domain or xconnect where MIPs and up MEPs will be created. The **id** sets the short MA name.
Step 5	**continuity-check interval** *time* [**loss-threshold** *threshold*] **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# continuity-check interval 100m loss-threshold 10`	(Optional) Enables Continuity Check and specifies the time interval at which CCMs are transmitted or to set the threshold limit for when a MEP is declared down.
Step 6	**continuity-check archive hold-time** *minutes* **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# continuity-check archive hold-time 100`	(Optional) Configures how long information about peer MEPs is stored after they have timed out.
Step 7	**continuity-check loss auto-traceroute** **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# continuity-check loss auto-traceroute`	(Optional) Configures automatic triggering of a traceroute when a MEP is declared down.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# commit`	Saves configuration changes. • When you use the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

	Command or Action	Purpose
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Automatic MIP Creation for a CFM Service

For more information about the algorithm for creating MIPs, see the **MIP Creation** section.

To configure automatic MIP creation for a CFM service, complete the following steps:

SUMMARY STEPS

1. **configure**
2. **ethernet cfm**
3. **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]]
4. **service** *service-name* {**down-meps** | **xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id** [**icc-based***icc-string umc-string*] | [**number** *number*]
5. **mip auto-create** {**all** | **lower-mep-only**} {**ccm-learning**}
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ethernet cfm** **Example:** `RP/0/RP0/CPU0:router# ethernet cfm`	Enters the Ethernet Connectivity Fault Management (CFM) configuration mode.
Step 3	**domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] **Example:** `RP/0/RP0/CPU0:router(config-cfm)# domain Domain_One level 1 id string D1`	Creates and names a container for all domain configurations and enters the CFM domain configuration mode. The level must be specified. The only supported option is **id [null]** for less than 1min interval MEPS. The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association

	Command or Action	Purpose
		identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.
Step 4	**service** *service-name* {**down-meps | xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id** [**icc-based***icc-string umc-string*] | [**number** *number*] **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn)# service ABC xconnect group X1 p2p ADB`	Configures and associates a service with the domain and enters CFM domain service configuration mode. You can specify that the service is used only for down MEPs, or associate the service with a bridge domain where MIPs and up MEPs will be created. The **id** sets the short MA name.
Step 5	**mip auto-create** {**all** | **lower-mep-only**} {**ccm-learning**} **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# mip auto-create all ccm-learning`	(Optional) Enables the automatic creation of MIPs in a bridge domain. **ccm-learning** option enables CCM learning for MIPs created in this service. This must be used only in services with a relatively long CCM interval of at least 100 ms. CCM learning at MIPs is disabled by default.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# commit`	Saves configuration changes. • When you use the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Cross-Check on a MEP for a CFM Service

To configure cross-check on a MEP for a CFM service and specify the expected set of MEPs, complete the following steps:

SUMMARY STEPS

1. **configure**
2. **ethernet cfm**

3. **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]]
4. **service** *service-name* {**bridge group** *bridge-domain-group* **bridge-domain** *bridge-domain-name* | **down-meps** | **xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id** [**icc-based** *icc-string umc-string*] | [**string** *text*] | [**number** *number*] | [**vlan-id** *id-number*] | [**vpn-id** *oui-vpnid*]]
5. **mep crosscheck**
6. **mep-id** *mep-id-number* [**mac-address** *mac-address*]
7. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ethernet cfm** **Example:** `RP/0/RP0/CPU0:router# ethernet cfm`	Enters the Ethernet Connectivity Fault Management (CFM) configuration mode.
Step 3	**domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] **Example:** `RP/0/RP0/CPU0:router(config-cfm)# domain Domain_One level 1 id string D1`	Creates and names a container for all domain configurations and enters the CFM domain configuration mode. The level must be specified. The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.
Step 4	**service** *service-name* {**bridge group** *bridge-domain-group* **bridge-domain** *bridge-domain-name* | **down-meps** | **xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id** [**icc-based** *icc-string umc-string*] | [**string** *text*] | [**number** *number*] | [**vlan-id** *id-number*] | [**vpn-id** *oui-vpnid*]] **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn)# service Bridge_Service bridge group BD1 bridge-domain B1`	Configures and associates a service with the domain and enters CFM domain service configuration mode. You can specify that the service is used only for down MEPs, or associate the service with a bridge domain or xconnect where MIPs and up MEPs will be created. The **id** sets the short MA name.
Step 5	**mep crosscheck** **Example:** `RP/0/RP0/CPU0:router(config-cfm-xcheck)# mep crosscheck mep-id 10`	Enters CFM MEP crosscheck configuration mode.
Step 6	**mep-id** *mep-id-number* [**mac-address** *mac-address*] **Example:**	Enables cross-check on a MEP.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cfm-xcheck)# mep-id 10`	**Note** • Repeat this command for every MEP that you want included in the expected set of MEPs for cross-check.
Step 7	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cfm-xcheck)# commit`	Saves configuration changes. • When you use the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Other Options for a CFM Service

To configure other options for a CFM service, complete the following steps:

SUMMARY STEPS

1. **configure**
2. **ethernet cfm**
3. **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]]
4. **service** *service-name* {**bridge group** *bridge-domain-group* **bridge-domain** *bridge-domain-name* | **down-meps | xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id** [**icc-based** *icc-string umc-string*] | [**string** *text*] | [**number** *number*] | [**vlan-id** *id-number*] | [**vpn-id** *oui-vpnid*]]
5. **maximum-meps** *number*
6. **log** {**ais**|**continuity-check errors**|**continuity-check mep changes**|**crosscheck errors**|**efd**}
7. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ethernet cfm** **Example:** `RP/0/RP0/CPU0:router# ethernet cfm`	Enters the Ethernet Connectivity Fault Management (CFM) configuration mode.
Step 3	**domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] **Example:** `RP/0/RP0/CPU0:router(config-cfm)# domain Domain_One level 1 id string D1`	Creates and names a container for all domain configurations and enters the CFM domain configuration mode. The level must be specified. The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.
Step 4	**service** *service-name* {**bridge group** *bridge-domain-group* **bridge-domain** *bridge-domain-name* | **down-meps** | **xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id** [**icc-based** *icc-string umc-string*] | [**string** *text*] | [**number** *number*] | [**vlan-id** *id-number*] | [**vpn-id** *oui-vpnid*]] **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn)# service Bridge_Service bridge group BD1 bridge-domain B1`	Configures and associates a service with the domain and enters CFM domain service configuration mode. You can specify that the service is used only for down MEPs, or associate the service with a bridge domain or xconnect where MIPs and up MEPs will be created. The **id** sets the short MA name.
Step 5	**maximum-meps** *number* **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# maximum-meps 1000`	(Optional) Configures the maximum number (2 to 8190) of MEPs across the network, which limits the number of peer MEPs recorded in the database.
Step 6	**log** {**ais**|**continuity-check errors**|**continuity-check mep changes**|**crosscheck errors**|**efd**} **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# log continuity-check errors`	(Optional) Enables logging of certain types of events.
Step 7	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# commit`	Saves configuration changes. • When you use the **end** command, the system prompts you to commit changes:

	Command or Action	Purpose
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring CFM MEPs

SUMMARY STEPS

1. **configure**
2. **interface** {**HundredGigE** \| **TenGigE**} *interface-path-id*
3. **interface** {**HundredGigE** \| **TenGigE** \| **Bundle-Ether**} *interface-path-id***.***subinterface*
4. **interface** {**HundredGigE** \| **TenGigE**} *interface-path-id*
5. **ethernet cfm**
6. **mep domain** *domain-name* **service** *service-name* **mep-id** *id-number*
7. **cos** *cos*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** {**HundredGigE** | **TenGigE**} *interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/1`	Type of Ethernet interface on which you want to create a MEP. Enter **HundredGigE** or **TenGigE** and the physical interface or virtual interface. **Note** • Use the **show interfaces** command to see a list of all interfaces currently configured on the router.

	Command or Action	Purpose
Step 3	**interface** {**HundredGigE** | **TenGigE** | **Bundle-Ether**} *interface-path-id***.***subinterface* **Example:** `RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/1.1`	Type of Ethernet interface on which you want to create a MEP. Enter **HundredGigE, TenGigE,** or **Bundle-Ether** and the physical interface or virtual interface followed by the subinterface path ID. Naming convention is *interface-path-id***.***subinterface*. The period in front of the subinterface value is required as part of the notation.
Step 4	**interface** {**HundredGigE** | **TenGigE**} *interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/1`	Type of Ethernet interface on which you want to create a MEP. Enter **HundredGigE** or **TenGigE** and the physical interface or virtual interface. **Note** • Use the **show interfaces** command to see a list of all interfaces currently configured on the router.
Step 5	**ethernet cfm** **Example:** `RP/0/RP0/CPU0:router(config-if)# ethernet cfm`	Enters interface Ethernet CFM configuration mode.
Step 6	**mep domain** *domain-name* **service** *service-name* **mep-id** *id-number* **Example:** `RP/0/RP0/CPU0:router(config-if-cfm)# mep domain Dm1 service Sv1 mep-id 1`	Creates a maintenance end point (MEP) on an interface and enters interface CFM MEP configuration mode.
Step 7	**cos** *cos* **Example:** `RP/0/RP0/CPU0:router(config-if-cfm-mep)# cos 7`	(Optional) Configures the class of service (CoS) (from 0 to 7) for all CFM packets generated by the MEP on an interface. If not configured, the CoS is inherited from the Ethernet interface. **Note** For Ethernet interfaces, the CoS is carried as a field in the VLAN tag. Therefore, CoS only applies to interfaces where packets are sent with VLAN tags. If the **cos (CFM)** command is executed for a MEP on an interface that does not have a VLAN encapsulation configured, it will be ignored.

Step 8	**end** or **commit**
	Example:
	`RP/0/RP0/CPU0:router(config-if-cfm-mep)# commit`
		Saves configuration changes.
		• When you use the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`

	Command or Action	Purpose
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Y.1731 AIS

This section has the following step procedures:

Configuring AIS in a CFM Domain Service

Use the following procedure to configure Alarm Indication Signal (AIS) transmission for a CFM domain service and configure AIS logging.

SUMMARY STEPS

1. **configure**
2. **ethernet cfm**
3. **domain** *name* **level** *level*
4. **service** *name* **bridge group** *name* **bridge-domain** *name*
5. **service** *name* **xconnect group** *xconnect-group-name* **p2p** *xconnect-name*
6. **ais transmission** [**interval** {**1s**|**1m**}][**cos** *cos*]
7. **log ais**
8. **no domain** *name***level** *level*
9. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ethernet cfm** **Example:**	Enters Ethernet CFM global configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config)# ethernet cfm`	
Step 3	**domain** *name* **level** *level* **Example:** `RP/0/RP0/CPU0:router(config-cfm)# domain D1 level 1`	Specifies the domain and domain level.
Step 4	**service** *name* **bridge group** *name* **bridge-domain** *name* **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn)# service S1 bridge group BG1 bridge-domain BD2`	Specifies the service, bridge group, and bridge domain.
Step 5	**service** *name* **xconnect group** *xconnect-group-name* **p2p** *xconnect-name* **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn)# service S1 xconnect group XG1 p2p X2`	Specifies the service and cross-connect group and name.
Step 6	**ais transmission** [**interval** {**1s**|**1m**}][**cos** *cos*] **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# ais transmission interval 1m cos 7`	Configures Alarm Indication Signal (AIS) transmission for a Connectivity Fault Management (CFM) domain service.
Step 7	**log ais** **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# log ais`	Configures AIS logging for a Connectivity Fault Management (CFM) domain service to indicate when AIS or LCK packets are received.
Step 8	**no domain** *name***level** *level* **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# no domain D1 level 1`	Disables the domain and domain level.
Step 9	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-sla-prof-stat-cfg)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

	Command or Action	Purpose
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring AIS on a CFM Interface

To configure AIS on a CFM interface, perform the following steps:

SUMMARY STEPS

1. **configure**
2. **interface gigabitethernet** *interface-path-id*
3. **ethernet cfm**
4. **ais transmission up interval 1m cos** *cos*
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface gigabitethernet** *interface-path-id* **Example:** `RP/0/RP0/CPU0:router# interface TenGigE 0/0/0/2`	Enters interface configuration mode.
Step 3	**ethernet cfm** **Example:** `RP/0/RP0/CPU0:router(config)# ethernet cfm`	Enters Ethernet CFM interface configuration mode.
Step 4	**ais transmission up interval 1m cos** *cos* **Example:** `RP/0/RP0/CPU0:router(config-if-cfm)# ais transmission up interval 1m cos 7`	Configures Alarm Indication Signal (AIS) transmission on a Connectivity Fault Management (CFM) interface.

	Command or Action	Purpose
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-sla-prof-stat-cfg)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Flexible VLAN Tagging for CFM

Use this procedure to set the number of tags in CFM packets in a CFM domain service.

SUMMARY STEPS

1. **configure**
2. **ethernet cfm**
3. **domain** *name* **level** *level*
4. **service** *name* **bridge group** *name* **bridge-domain** *name*
5. **tags** *number*
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ethernet cfm** **Example:**	Enters Ethernet CFM global configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config)# ethernet cfm`	
Step 3	**domain** *name* **level** *level* **Example:** `RP/0/RP0/CPU0:router(config-cfm)# domain D1 level 1`	Specifies the domain and domain level.
Step 4	**service** *name* **bridge group** *name* **bridge-domain** *name* **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn)# service S2 bridge group BG1 bridge-domain BD2`	Specifies the service, bridge group, and bridge domain.
Step 5	**tags** *number* **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# tags 1`	Specifies the number of tags in CFM packets. Currently, the only valid value is 1.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Verifying the CFM Configuration

To verify the CFM configuration, use one or more of the following commands:

show ethernet cfm configuration-errors [**domain** *domain-name*] [**interface** *interface-path-id*]	Displays information about errors that are preventing configured CFM operations from becoming active, as well as any warnings that have occurred.
show ethernet cfm local maintenance-points domain *name* [**service** *name]* | **interface** *type interface-path-id*] [**mep** | **mip**]	Displays a list of local maintenance points.

Note After you configure CFM, the error message, *cfmd[317]: %L2-CFM-5-CCM_ERROR_CCMS_MISSED : Some received CCMs have not been counted by the CCM error counters*, may display. This error message does not have any functional impact and does not require any action from you.

Troubleshooting Tips

To troubleshoot problems within the CFM network, perform these steps:

SUMMARY STEPS

1. To verify connectivity to a problematic MEP, use the **ping ethernet cfm** command as shown in this example:
2. If the results of the **ping ethernet cfm** command show a problem with connectivity to the peer MEP, use the **traceroute ethernet cfm** command to help further isolate the location of the problem as shown in the following example:

DETAILED STEPS

Step 1 To verify connectivity to a problematic MEP, use the **ping ethernet cfm** command as shown in this example:

```
RP/0/RP0/CPU0:router# ping ethernet cfm domain D1 service S1 mep-id 16 source
interface TenGigE 0/0/0/1

Type escape sequence to abort.
Sending 5 CFM Loopbacks, timeout is 2 seconds -
Domain foo (level 2), Service foo
Source: MEP ID 1, interface TenGigE0/0/0/1
Target: 0001.0002.0003 (MEP ID 16):
  Running (5s) ...
Success rate is 60.0 percent (3/5), round-trip min/avg/max = 1251/1349/1402 ms
Out-of-sequence: 0.0 percent (0/3)
Bad data: 0.0 percent (0/3)
Received packet rate: 1.4 pps
```

Step 2 If the results of the **ping ethernet cfm** command show a problem with connectivity to the peer MEP, use the **traceroute ethernet cfm** command to help further isolate the location of the problem as shown in the following example:

```
RP/0/RP0/CPU0:router# traceroute ethernet cfm domain D1 service S1 mep-id 16 source
interface TenGigE 0/0/0/2

Traceroutes in domain D1 (level 4), service S1
```

```
Source: MEP-ID 1, interface TenGigE0/0/0/2
================================================================================
Traceroute at 2009-05-18 12:09:10 to 0001.0203.0402,
TTL 64, Trans ID 2:

Hop Hostname/Last             Ingress MAC/name       Egress MAC/Name        Relay
--- ------------------------ ---------------------- ---------------------- -----
  1 ios                      0001.0203.0400 [Down]                         FDB
    0000-0001.0203.0400      TenGigE0/0/0/2
  2 abc                                             0001.0203.0401 [Ok]    FDB
    ios                                             Not present
  3 bcd                      0001.0203.0402 [Ok]                           Hit
    abc                      TenGigE0/0
Replies dropped: 0
```

If the target was a MEP, verify that the last hop shows "Hit" in the Relay field to confirm connectivity to the peer MEP.

If the Relay field contains "MPDB" for any of the hops, then the target MAC address was not found in the bridge MAC learning table at that hop, and the result is relying on CCM learning. This result can occur under normal conditions, but it can also indicate a problem. If you used the **ping ethernet cfm** command before using the **traceroute ethernet cfm** command, then the MAC address should have been learned. If "MPDB" is appearing in that case, then this indicates a problem at that point in the network.

Configuration Examples for Ethernet OAM

This section provides the following configuration examples:

Configuration Examples for EOAM Interfaces

This section provides the following configuration examples:

Configuring an Ethernet OAM Profile Globally: Example

This example shows how to configure an Ethernet OAM profile globally:

```
configure terminal
 ethernet oam profile Profile_1
  link-monitor
   symbol-period window 60000
   symbol-period threshold ppm low 10000000 high 60000000
   frame window 60
   frame threshold ppm low 10000000 high 60000000
   frame-period window 60000
   frame-period threshold ppm low 100 high 12000000
   frame-seconds window 900000
   frame-seconds threshold low 3 high 900
   exit
  mib-retrieval
  connection timeout 30
  require-remote mode active
  require-remote mib-retrieval
  action dying-gasp error-disable-interface
  action critical-event error-disable-interface
  action discovery-timeout error-disable-interface
  action session-down error-disable-interface
  action capabilities-conflict error-disable-interface
  action wiring-conflict error-disable-interface
```

```
  action remote-loopback error-disable-interface
  commit
```

Configuring Ethernet OAM Features on an Individual Interface: Example

This example shows how to configure Ethernet OAM features on an individual interface:

```
configure terminal
 interface TenGigE 0/0/0/0
  ethernet oam
   link-monitor
    symbol-period window 60000
    symbol-period threshold ppm low 10000000 high 60000000
    frame window 60
    frame threshold ppm low 10000000 high 60000000
    frame-period window 60000
    frame-period threshold ppm low 100 high 12000000
    frame-seconds window 900000
    frame-seconds threshold low 3 high 900
    exit
   mib-retrieval
   connection timeout 30
   require-remote mode active
   require-remote mib-retrieval
   action link-fault  error-disable-interface
   action dying-gasp error-disable-interface
   action critical-event error-disable-interface
   action discovery-timeout error-disable-interface
   action session-down error-disable-interface
   action capabilities-conflict error-disable-interface
   action wiring-conflict error-disable-interface
   action remote-loopback error-disable-interface
   commit
```

Configuring Ethernet OAM Features to Override the Profile on an Individual Interface: Example

This example shows the configuration of Ethernet OAM features in a profile followed by an override of that configuration on an interface:

```
configure terminal
 ethernet oam profile Profile_1
  mode passive
  action dying-gasp disable
  action critical-event disable
  action discovery-timeout disable
  action session-up disable
  action session-down disable
  action capabilities-conflict disable
  action wiring-conflict disable
  action remote-loopback disable
  action uni-directional link-fault error-disable-interface
  commit

configure terminal
 interface TenGigE 0/0/0/0
  ethernet oam
   profile Profile_1
    mode active
    action dying-gasp log
    action critical-event log
    action discovery-timeout log
    action session-up log
```

```
    action session-down log
    action capabilities-conflict log
    action wiring-conflict log
    action remote-loopback log
    action uni-directional link-fault log
    uni-directional link-fault detection
    commit
```

Clearing Ethernet OAM Statistics on an Interface: Example

This example shows how to clear Ethernet OAM statistics on an interface:

```
RP/0/RP0/CPU0:router# clear ethernet oam statistics interface gigabitethernet 0/0/0/1
```

Enabling SNMP Server Traps on a Router: Example

This example shows how to enable SNMP server traps on a router:

```
configure terminal
 snmp-server traps ethernet oam events
```

Configuration Examples for Ethernet CFM

This section includes the following examples:

Ethernet CFM Domain Configuration: Example

This example shows how to configure a basic domain for Ethernet CFM:

```
configure
 ethernet cfm
  traceroute cache hold-time 1 size 3000
  domain Domain_One level 1 id string D1
  commit
```

Ethernet CFM Service Configuration: Example

This example shows how to create a service for an Ethernet CFM domain:

```
  service Bridge_Service bridge group BD1 bridge-domain B1
  service Cross_Connect_1 xconnect group XG1 p2p X1
  commit
```

Flexible Tagging for an Ethernet CFM Service Configuration: Example

This example shows how to set the number of tags in CFM packets from down MEPs in a CFM domain service:

```
configure
 ethernet cfm
  domain D1 level 1
   service S2 bridge group BG1 bridge-domain BD2
    tags 1
    commit
```

Continuity Check for an Ethernet CFM Service Configuration: Example

This example shows how to configure continuity-check options for an Ethernet CFM service:

```
   continuity-check archive hold-time 100
   continuity-check loss auto-traceroute
   continuity-check interval 100ms loss-threshold 10
   commit
```

MIP Creation for an Ethernet CFM Service Configuration: Example

This example shows how to enable MIP auto-creation for an Ethernet CFM service:

```
RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# mip auto-create all
RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# commit
```

Cross-check for an Ethernet CFM Service Configuration: Example

This example shows how to configure cross-check for MEPs in an Ethernet CFM service:

```
   mep crosscheck
    mep-id 10
    mep-id 20
    commit
```

Other Ethernet CFM Service Parameter Configuration: Example

This example shows how to configure other Ethernet CFM service options:

```
    maximum-meps 4000
    log continuity-check errors
    commit
    exit
   exit
  exit
```

MEP Configuration: Example

This example shows how to configure a MEP for Ethernet CFM on an interface:

```
 interface TenGigE 0/0/0/1
  ethernet cfm
  mep domain Dm1 service Sv1 mep-id 1
  commit
```

Ethernet CFM Show Command: Examples

These examples show how to verify the configuration of Ethernet Connectivity Fault Management (CFM):

Example 1

This example shows how to display all the maintenance points that have been created on an interface:

```
RP/0/RP0/CPU0:router# show ethernet cfm local maintenance-points

Domain/Level         Service             Interface         Type   ID   MAC
-------------------- ------------------- ----------------- ------ ---- --------
fig/5                bay                 Gi0/10/0/12       Dn MEP    2 44:55:66
fig/5                bay                 Gi0/0/1/0         MIP         55:66:77
fred/3               barney              Gi0/1/0/0         Dn MEP    5 66:77:88!
```

Example 2

This example shows how to display all the CFM configuration errors on all domains:

```
RP/0/RP0/CPU0:router# show ethernet cfm configuration-errors

Domain fig (level 5), Service bay
* MIP creation configured using bridge-domain blort, but bridge-domain blort does not exist.

 * An Up MEP is configured for this domain on interface TenGigE0/0/0/3 and an Up MEP is
also configured for domain blort, which is at the same level (5).
 * A MEP is configured on interface TenGigE0/0/0/1 for this domain/service, which has CC
interval 100ms, but the lowest interval supported on that interface is 1s
```

Example 3

This example shows how to display operational state for local maintenance end points (MEPs):

```
RP/0/RP0/CPU0:router# show ethernet cfm local meps

 A - AIS received                I - Wrong interval
 R - Remote Defect received      V - Wrong Level
 L - Loop (our MAC received)     T - Timed out (archived)
 C - Config (our ID received)    M - Missing (cross-check)
 X - Cross-connect (wrong MAID)  U - Unexpected (cross-check)
 P - Peer port down

Domain foo (level 6), Service bar
   ID Interface (State)        Dir MEPs/Err RD Defects AIS
----- ------------------------ --- -------- -- ------- ---
  100 Gi1/1/0/1 (Up)       Up      0/0   N  A       L7

Domain fred (level 5), Service barney
   ID Interface (State)        Dir MEPs/Err RD Defects AIS
----- ------------------------ --- -------- -- ------- ---
    2 Gi0/1/0/0 (Up)       Up      3/2   Y  RPC     L6
Domain foo (level 6), Service bar
   ID Interface (State)        Dir MEPs/Err RD Defects AIS
----- ------------------------ --- -------- -- ------- ---
  100 Gi1/1/0/1 (Up)       Up      0/0   N  A

Domain fred (level 5), Service barney
   ID Interface (State)        Dir MEPs/Err RD Defects AIS
----- ------------------------ --- -------- -- ------- ---
    2 Gi0/1/0/0 (Up)       Up      3/2   Y  RPC
```

Example 4

This example shows how to display operational state of other maintenance end points (MEPs) detected by a local MEP:

```
RP/0/RP0/CPU0:router# show ethernet cfm peer meps

Flags:
 > - Ok                           I - Wrong interval
 R - Remote Defect received       V - Wrong level
 L - Loop (our MAC received)      T - Timed out
 C - Config (our ID received)     M - Missing (cross-check)
 X - Cross-connect (wrong MAID)   U - Unexpected (cross-check)
```

```
Domain fred (level 7), Service barney
Down MEP on TenGigE0/0/0/1, MEP-ID 2
================================================================================
St    ID MAC address    Port    Up/Downtime   CcmRcvd SeqErr   RDI Error
-- ----- -------------- ------- ----------- --------- ------ ----- -----
 >     1 0011.2233.4455 Up      00:00:01         1234      0     0     0
R>     4 4455.6677.8899 Up      1d 03:04         3456      0   234     0
L      2 1122.3344.5566 Up      3w 1d 6h         3254      0     0  3254
C      2 7788.9900.1122 Test    00:13            2345      6    20  2345
X      3 2233.4455.6677 Up      00:23              30      0     0    30
I      3 3344.5566.7788 Down    00:34           12345      0   300  1234
V      3 8899.0011.2233 Blocked 00:35              45      0     0    45
 T     5 5566.7788.9900         00:56              20      0     0     0
M      6                                            0      0     0     0
U>     7 6677.8899.0011 Up      00:02             456      0     0     0

Domain fred (level 7), Service fig
Down MEP on TenGigE0/0/0/12, MEP-ID 3
================================================================================
St    ID MAC address    Port    Up/Downtime  CcmRcvd SeqErr   RDI Error
-- ----- -------------- ------- ----------- -------- ------ ----- -----
 >     1 9900.1122.3344 Up      03:45            4321      0     0     0
```

Example 5

This example shows how to display operational state of other maintenance end points (MEPs) detected by a local MEP with details:

```
RP/0/RP0/CPU0:router#  show ethernet cfm peer meps detail
Domain dom3 (level 5), Service ser3
Down MEP on TenGigE0/0/0/1 MEP-ID 1
================================================================================
Peer MEP-ID 10, MAC 0001.0203.0403
   CFM state: Wrong level, for 00:01:34
   Port state: Up
   CCM defects detected:    V - Wrong Level
   CCMs received: 5
     Out-of-sequence:             0
     Remote Defect received:      5
     Wrong Level:                 0
     Cross-connect (wrong MAID):  0
     Wrong Interval:              5
     Loop (our MAC received):     0
     Config (our ID received):    0
Last CCM received 00:00:06 ago:
     Level: 4, Version: 0, Interval: 1min
     Sequence number: 5, MEP-ID: 10
     MAID: String: dom3, String: ser3
     Port status: Up, Interface status: Up


Domain dom4 (level 2), Service ser4
Down MEP on TenGigE0/0/0/2 MEP-ID 1
================================================================================
Peer MEP-ID 20, MAC 0001.0203.0402
   CFM state: Ok, for 00:00:04
   Port state: Up
   CCMs received: 7
     Out-of-sequence:             1
     Remote Defect received:      0
     Wrong Level:                 0
     Cross-connect (wrong MAID):  0
     Wrong Interval:              0
```

```
      Loop (our MAC received):     0
   Config (our ID received):    0
 Last CCM received 00:00:04 ago:
      Level: 2, Version: 0, Interval: 10s
      Sequence number: 1, MEP-ID: 20
      MAID: String: dom4, String: ser4
      Chassis ID: Local: ios; Management address: 'Not specified'
      Port status: Up, Interface status: Up

 Peer MEP-ID 21, MAC 0001.0203.0403
    CFM state: Ok, for 00:00:05
    Port state: Up
    CCMs received: 6
      Out-of-sequence:             0
      Remote Defect received:      0
      Wrong Level:                 0
      Cross-connect (wrong MAID):  0
      Wrong Interval:              0
      Loop (our MAC received):     0
      Config (our ID received):    0
 Last CCM received 00:00:05 ago:
      Level: 2, Version: 0, Interval: 10s
      Sequence number: 1, MEP-ID: 21
      MAID: String: dom4, String: ser4
      Port status: Up, Interface status: Up


 Peer MEP-ID 601, MAC 0001.0203.0402
    CFM state: Timed Out (Standby), for 00:15:14, RDI received
    Port state: Down
    CCM defects detected:    Defects below ignored on local standby MEP
                             I - Wrong Interval
                             R - Remote Defect received
                             T - Timed Out
                             P - Peer port down
    CCMs received: 2
      Out-of-sequence:             0
      Remote Defect received:      2
      Wrong Level:                 0

      Wrong Interval:              2
      Loop (our MAC received):     0
      Config (our ID received):    0
    Last CCM received 00:15:49 ago:
      Level: 2, Version: 0, Interval: 10s
      Sequence number: 1, MEP-ID: 600
      MAID: DNS-like: dom5, String: ser5
      Chassis ID: Local: ios; Management address: 'Not specified'
      Port status: Up, Interface status: Down
```

AIS for CFM Configuration: Examples

Example 1

This example shows how to configure Alarm Indication Signal (AIS) transmission for a CFM domain service:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# ethernet cfm
RP/0/RP0/CPU0:router(config-cfm)# domain D1 level 1
RP/0/RP0/CPU0:router(config-cfm-dmn)# service S1 bridge group BG1 bridge-domain BD2
RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# ais transmission interval 1m cos 7
```

```
RP/0/RP0/CPU0:routerconfigure
RP/0/RP0/CPU0:router(config)# ethernet cfm
RP/0/RP0/CPU0:router(config-cfm)# domain D1 level 1
RP/0/RP0/CPU0:router(config-cfm-dmn)# service Cross_Connect_1 xconnect group XG1 p2p X1
RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# ais transmission interval 1m cos 7
```

Example 2

This example shows how to configure AIS logging for a Connectivity Fault Management (CFM) domain service to indicate when AIS or LCK packets are received:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# ethernet cfm
RP/0/RP0/CPU0:router(config-cfm)# domain D1 level 1
RP/0/RP0/CPU0:router(config-cfm-dmn)# service S2 bridge group BG1 bridge-domain BD2
RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# log ais

RP/0/RP0/CPU0:routerconfigure
RP/0/RP0/CPU0:router(config)# ethernet cfm
RP/0/RP0/CPU0:router(config-cfm)# domain D1 level 1
RP/0/RP0/CPU0:router(config-cfm-dmn)# service Cross_Connect_1 xconnect group XG1 p2p X1
RP/0/RP0/CPU0:router(config-cfm-dmn-svc)# log ais
```

This example shows how to configure AIS transmission on a CFM interface.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/2
RP/0/RP0/CPU0:router(config-if)# ethernet cfm
RP/0/RP0/CPU0:router(config-if-cfm)# ais transmission up interval 1m cos 7
```

AIS for CFM Show Commands: Examples

This section includes the following examples:

show ethernet cfm interfaces ais Command: Example

This example shows how to display the information published in the Interface AIS table:

```
RP/0/RP0/CPU0:router# show ethernet cfm interfaces ais

Defects (from at least one peer MEP):
 A - AIS received                I - Wrong interval
 R - Remote Defect received      V - Wrong Level
 L - Loop (our MAC received)     T - Timed out (archived)
 C - Config (our ID received)    M - Missing (cross-check)
 X - Cross-connect (wrong MAID)  U - Unexpected (cross-check)
 P - Peer port down              D - Local port down

                               Trigger                   Transmission
                         AIS  ---------    Via    ---------------------------
Interface (State)        Dir  L Defects  Levels   L Int Last started  Packets
------------------------ ---  - -------  -------  - --- ------------ --------
TenGigE0/0/0/0 (Up)       Dn   5 RPC      6        7 1s  01:32:56 ago     5576
TenGigE0/0/0/0 (Up)       Up   0 M        2,3      5 1s  00:16:23 ago      983
TenGigE0/0/0/1 (Dn)       Up     D                 7 60s 01:02:44 ago     3764
TenGigE0/0/0/2 (Up)       Dn   0 RX       1!
```

show ethernet cfm local meps Command: Examples

Example 1: Default

This example shows how to display statistics for local maintenance end points (MEPs):

```
RP/0/RP0/CPU0:router# show ethernet cfm local meps

 A - AIS received                I - Wrong interval
 R - Remote Defect received      V - Wrong Level
 L - Loop (our MAC received)     T - Timed out (archived)
 C - Config (our ID received)    M - Missing (cross-check)
 X - Cross-connect (wrong MAID)  U - Unexpected (cross-check)
 P - Peer port down

Domain foo (level 6), Service bar
   ID Interface (State)        Dir MEPs/Err RD Defects AIS
----- ------------------------ --- -------- -- ------- ---
  100 Gi1/1/0/1 (Up)       Up     0/0   N  A       7

Domain fred (level 5), Service barney
   ID Interface (State)        Dir MEPs/Err RD Defects AIS
----- ------------------------ --- -------- -- ------- ---
    2 Gi0/1/0/0 (Up)       Up     3/2   Y  RPC     6
```

Example 2: Domain Service

This example shows how to display statistics for MEPs in a domain service:

```
RP/0/RP0/CPU0:router# show ethernet cfm local meps domain foo service bar detail

Domain foo (level 6), Service bar
Down MEP on TenGigE0/0/0/1, MEP-ID 100
================================================================================
  Interface state: Up      MAC address: 1122.3344.5566
  Peer MEPs: 0 up, 0 with errors, 0 timed out (archived)

  CCM generation enabled:  No
  AIS generation enabled:  Yes (level: 7, interval: 1s)
  Sending AIS:             Yes (started 01:32:56 ago)
  Receiving AIS:           Yes (from lower MEP, started 01:32:56 ago)

Domain fred (level 5), Service barney
Down MEP on TenGigE0/0/0/1, MEP-ID 2
================================================================================
  Interface state: Up      MAC address: 1122.3344.5566
  Peer MEPs: 3 up, 2 with errors, 0 timed out (archived)
  Cross-check defects: 0 missing, 0 unexpected

  CCM generation enabled:  Yes (Remote Defect detected: Yes)
  CCM defects detected:    R - Remote Defect received
                           P - Peer port down
                           C - Config (our ID received)
  AIS generation enabled:  Yes (level: 6, interval: 1s)
  Sending AIS:             Yes (to higher MEP, started 01:32:56 ago)
  Receiving AIS:           No
```

Example 4: Detail

This example shows how to display detailed statistics for MEPs in a domain service:

```
RP/0/RP0/CPU0:router# show ethernet cfm local meps detail

Domain foo (level 6), Service bar
Down MEP on TenGigE0/0/0/1, MEP-ID 100
================================================================================
  Interface state: Up      MAC address: 1122.3344.5566
  Peer MEPs: 0 up, 0 with errors, 0 timed out (archived)

  CCM generation enabled:  No
  AIS generation enabled:  Yes (level: 7, interval: 1s)
  Sending AIS:             Yes (started 01:32:56 ago)
  Receiving AIS:           Yes (from lower MEP, started 01:32:56 ago)

Domain fred (level 5), Service barney
Down MEP on TenGigE0/0/0/1, MEP-ID 2
================================================================================
  Interface state: Up      MAC address: 1122.3344.5566
  Peer MEPs: 3 up, 2 with errors, 0 timed out (archived)
  Cross-check defects: 0 missing, 0 unexpected

  CCM generation enabled:  Yes (Remote Defect detected: Yes)
  CCM defects detected:    R - Remote Defect received
                           P - Peer port down
                           C - Config (our ID received)
  AIS generation enabled:  Yes (level: 6, interval: 1s)
  Sending AIS:             Yes (to higher MEP, started 01:32:56 ago)
  Receiving AIS:           No
```

show ethernet cfm local meps detail Command: Example

Use the **show ethernet cfm local meps detail** command to display MEP-related EFD status information. This example shows that EFD is triggered for MEP-ID 100:

```
RP/0/RP0/CPU0:router# show ethernet cfm local meps detail

Domain foo (level 6), Service bar
Down MEP on TenGigE0/0/0/1, MEP-ID 100
================================================================================
  Interface state: Up      MAC address: 1122.3344.5566
  Peer MEPs: 0 up, 0 with errors, 0 timed out (archived)
  Cross-check errors: 2 missing, 0 unexpected

  CCM generation enabled:  No
  AIS generation enabled:  Yes (level: 7, interval: 1s)
  Sending AIS:             Yes (started 01:32:56 ago)
  Receiving AIS:           Yes (from lower MEP, started 01:32:56 ago)
  EFD triggered:           Yes

Domain fred (level 5), Service barney
Down MEP on TenGigE0/0/0/1, MEP-ID 2
================================================================================
  Interface state: Up      MAC address: 1122.3344.5566
  Peer MEPs: 3 up, 0 with errors, 0 timed out (archived)
  Cross-check errors: 0 missing, 0 unexpected

  CCM generation enabled:  Yes (Remote Defect detected: No)
  AIS generation enabled:  Yes (level: 6, interval: 1s)
  Sending AIS:             No
  Receiving AIS:           No
  EFD triggered:           No
```

> **Note** You can also verify that EFD has been triggered on an interface using the **show interfaces** and **show interfaces brief** commands. When an EFD trigger has occurred, these commands will show the interface status as *up* and the line protocol state as *down*.

CFM Over Bundles

CFM over bundle supports the following:

- CFM Maintenance Points—Up Maintenance-association End Points (MEP), Down MEP, and MIP, which includes L2 bundle main and sub-interfaces.
- CCM interval of 100 microsecond, 1second, 10 seconds, and 1 minute. CCM interval of 10 minutes is supported only in the versions earlier than IOS XR 7.3.2.
- RP OIR/VM reload, without impacting learned CFM peer MEPs.
- Process restart without impacting CFM sessions.
- CFM MEPs on bundle interfaces as software-offloaded-MEPs with all possible rewrite and encapsulation combinations supported by L2 sub-interfaces.
- CCM learning on MIP over bundle interfaces. CCM database learning supports investigation of one CCM out of 50 that goes over MIP.
- Static and dynamic MEPs.

Restrictions for Configuration of CFM on Bundles

Following are the restrictions for configuring CFM over bundle member interfaces:

- Only Layer 2 bundle Ethernet interfaces and sub-interfaces are supported except for those matching the VLAN tag `any`.
- CCM interval of 3.3 milliseconds and 10 milliseconds are not supported.
- CCM interval of 10 minutes is not supported from IOS XR 7.3.2.
- Supports 5000 pps rates of CCM traffic for bundle interfaces.
- Ethernet CFM is not supported with MEP that are configured on default and untagged encapsulated sub-interfaces that are part of a single physical interface.

Y.1731 Performance Monitoring

Y.1731 Performance Monitoring (PM) provides a standard Ethernet PM function that includes measurement of Ethernet frame delay, frame delay variation, frame loss, and frame throughput measurements. This is specified by the ITU-T Y-1731 standard and interpreted by the Metro Ethernet Forum (MEF) standards group.

The router supports the following:

- Delay Measurement (DM)

- Synthetic Loss Measurement (SLM)

Two-Way Delay Measurement for Scalability

Use the Ethernet frame delay measurement to measure frame delay and frame delay variations. The system measures the Ethernet frame delay by using the Delay Measurement Message (DMM) method.

Restrictions for Configuring Two-Way Delay Measurement

Follow the guidelines and restrictions listed here when you configure two-way delay measurement:

- Y.1731 PM is not supported for One-Way DMM in release prior to XR Release 6.6.25.

Configuring Two-Way Delay Measurement

Perform the following steps to configure two-way delay measurement:

RP/0/RP0/CPU0:router **(config)** # **ethernet sla**

```
profile DMM type cfm-delay-measurement
  probe
   send burst every 5 seconds packet count 5 interval 1 seconds
  !
  schedule
   every 1 minutes for 40 seconds
  !
  statistics
   measure round-trip-delay
    buckets size 1 probes
    buckets archive 5
   !
   measure round-trip-jitter
    buckets size 1 probes
    buckets archive 1
   !
!
!
!
interface TenGigE0/0/0/10.1 l2transport
encapsulation dot1q 1
ethernet cfm
  mep domain DOWN0 service s10 mep-id 2001
   sla operation profile DMM target mep-id 6001
  !
```

Configuring an On-Demand Ethernet SLA Operation for CFM Delay Measurement

To configure an on-demand Ethernet SLA operation for CFM delay measurement, use this command in privileged EXEC configuration mode:

RP/0/RP0/CPU0:router (config) #

```
ethernet sla on-demand operation type cfm-synthetic-loss-measurement probe domain D1 source
 interface TenGigE 0/6/1/0 target mac-address 2.3.4
```

Running Configuration

```
RP/0/RP0/CPU0:router# show ethernet cfm peer meps
Mon Sep 11 12:09:44.534 UTC
```

```
Flags:
> - Ok                            I - Wrong interval
R - Remote Defect received        V - Wrong level
L - Loop (our MAC received)       T - Timed out
C - Config (our ID received)      M - Missing (cross-check)
X - Cross-connect (wrong MAID)    U - Unexpected (cross-check)
* - Multiple errors received      S - Standby

Domain UP6 (level 6), Service s6
Up MEP on FortyGigE0/0/1/2.1 MEP-ID 1
================================================================================
St    ID MAC Address    Port    Up/Downtime   CcmRcvd SeqErr   RDI Error
-- ----- -------------- ------- ----------- --------- ------ ----- -----
 >  4001 70e4.227c.2865 Up      00:01:27            0      0     0     0

Domain DOWN0 (level 0), Service s10
Down MEP on TenGigE0/0/0/10.1 MEP-ID 2001
================================================================================
St    ID MAC Address    Port    Up/Downtime   CcmRcvd SeqErr   RDI Error
-- ----- -------------- ------- ----------- --------- ------ ----- -----
 >  6001 70e4.227c.287a Up      00:02:11            0      0     0     0
RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router# show running-config
Mon Sep 11 12:10:18.467 UTC
Building configuration...
!! IOS XR Configuration version = 6.4.1.14
!! Last configuration change at Mon Sep 11 12:08:16 2017 by root
!
logging console disable
telnet vrf default ipv4 server max-servers 10
username root
group root-lr
group cisco-support
secret 5 $1$QJT3$94M5/wK5J0v/lpAu/wz31/
!
line console
exec-timeout 0 0
!
ethernet cfm
domain UP6 level 6 id null
  service s6 xconnect group g1 p2p p1 id number 6
   mip auto-create all ccm-learning
   continuity-check interval 1s
   mep crosscheck
    mep-id 4001
   !
  !
!
domain DOWN0 level 0 id null
 service s10 down-meps id number 10
   continuity-check interval 1s
   mep crosscheck
    mep-id 6001
   !
  !
!
!
profile DMM type cfm-delay-measurement
  probe
   send burst every 5 seconds packet count 5 interval 1 seconds
  !
  schedule
   every 1 minutes for 40 seconds
  !
```

```
  statistics
   measure round-trip-delay
    buckets size 1 probes
    buckets archive 5
   !
   measure round-trip-jitter
    buckets size 1 probes
    buckets archive 1
   !
interface MgmtEth0/RP0/CPU0/0
shutdown
!
interface TenGigE0/0/0/0
shutdown
!
interface TenGigE0/0/0/1
shutdown
!
interface TenGigE0/0/0/2
shutdown
!
interface TenGigE0/0/0/3
shutdown
!
interface TenGigE0/0/0/4
shutdown
!
interface TenGigE0/0/0/5
shutdown
!
interface TenGigE0/0/0/6
shutdown
!
interface TenGigE0/0/0/7
shutdown
!
interface TenGigE0/0/0/8
shutdown
!
interface TenGigE0/0/0/9
shutdown
!
interface TenGigE0/0/0/10.1 l2transport
encapsulation dot1q 1
ethernet cfm
  mep domain DOWN0 service s10 mep-id 2001
   sla operation profile DMM target mep-id 6001
   sla operation profile test-slm target mep-id 6001
  !
!
!
interface TenGigE0/0/0/11
shutdown
!
interface TenGigE0/0/0/12
shutdown
!
interface TenGigE0/0/0/13
shutdown
!
interface TenGigE0/0/0/14
shutdown
!
interface TenGigE0/0/0/15
```

```
shutdown
!
interface TenGigE0/0/0/16
shutdown
!
interface TenGigE0/0/0/17
shutdown
!
interface TenGigE0/0/0/18
shutdown
!
interface TenGigE0/0/0/19
shutdown
!
interface TenGigE0/0/0/20
shutdown
!
interface TenGigE0/0/0/21
shutdown
!
interface TenGigE0/0/0/22
shutdown
!
interface TenGigE0/0/0/23
shutdown
!
interface TenGigE0/0/0/24
shutdown
!
interface TenGigE0/0/0/25
shutdown
!
interface TenGigE0/0/0/26
shutdown
!
interface TenGigE0/0/0/27
shutdown
!
interface TenGigE0/0/0/28
shutdown
!
interface TenGigE0/0/0/29
shutdown
!
interface TenGigE0/0/0/30
shutdown
!
!
interface TenGigE0/0/0/31
 shutdown
!
interface TenGigE0/0/0/32
 shutdown
!
interface TenGigE0/0/0/33
 shutdown
!
interface TenGigE0/0/0/34
 shutdown
!
interface TenGigE0/0/0/35
 shutdown
!
interface TenGigE0/0/0/36
```

```
 shutdown
!
interface TenGigE0/0/0/37
 shutdown
!
interface TenGigE0/0/0/38
 shutdown
!
interface TenGigE0/0/0/39
 shutdown
!
interface TenGigE0/0/1/0/1
 shutdown
!
interface TenGigE0/0/1/0/2
 shutdown
!
interface TenGigE0/0/1/0/3
 shutdown
!
controller Optics0/0/1/0
 breakout 4x10
!
interface HundredGigE0/0/1/1
 shutdown
!
interface FortyGigE0/0/1/2.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain UP6 service s6 mep-id 1
    sla operation profile DMM target mep-id 6001
    sla operation profile test-slm target mep-id 6001
  !
 !
!
l2vpn
 xconnect group g1
  p2p p1
   interface TenGigE0/0/0/10.1
   interface FortyGigE0/0/1/2.1
  !
 !
!
end
```

Verification

```
One-way Delay (Source->Dest)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 10
    Min: 1912765.961ms; Max: 1912765.961ms; Mean: 1912765.961ms; StdDev: -2147483.648ms


One-way Delay (Dest->Source)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
```

```
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 10
    Min: -1912765.952ms; Max: -1912765.951ms; Mean: -1912765.951ms; StdDev: -2147483.648ms


Round Trip Jitter
~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 9
    Min: 0.000ms; Max: 0.001ms; Mean: 0.000ms; StdDev: 0.000ms


One-way Jitter (Source->Dest)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 9
    Min: 0.000ms; Max: 0.000ms; Mean: 0.000ms; StdDev: 0.000ms


One-way Jitter (Dest->Source)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 9
    Min: 0.000ms; Max: 0.001ms; Mean: 0.000ms; StdDev: 0.000ms


RP/0/RP0/CPU0:ios#ethernet sla on-demand operation type cfm-syn probe domain DOWN0 source
interface tenGigE 0/0/0/10.1 target mep-id 6001
Mon Sep 11 12:12:39.259 UTC
Warning: Burst configuration is present and so this profile cannot be represented in the
MEF-SOAM-PM-MIB configuration tables. However, the statistics are still collected
On-demand operation 2 succesfully created
 / - Completed - statistics will be displayed shortly.
RP/0/RP0/CPU0:ios#show ethernet sla statistics on-demand id 2

Mon Sep 11 12:13:24.825 UTC
Source: Interface TenGigE0/0/0/10.1, Domain DOWN0
Destination: Target MEP-ID 6001
================================================================================
On-demand operation ID #2, packet type 'cfm-synthetic-loss-measurement'
Started at 12:12:41 UTC Mon 11 September 2017, runs once for 10s
Frame Loss Ratio calculated every 10s

One-way Frame Loss (Source->Dest)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:12:41 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 100; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 1
    Min: 0.000%; Max: 0.000%; Mean; 0.000%; StdDev: 0.000%; Overall: 0.000%
```

```
One-way Frame Loss (Dest->Source)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:12:41 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 100; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 1
    Min: 0.000%; Max: 0.000%; Mean; 0.000%; StdDev: 0.000%; Overall: 0.000%

RP/0/RP0/CPU0:ios#show ethernet cfm local meps verbose
Mon Sep 11 12:13:04.461 UTC
Domain UP6 (level 6), Service s6
Up MEP on FortyGigE0/0/1/2.1 MEP-ID 1
================================================================================
  Interface state: Up      MAC address: 008a.960f.c4a8
  Peer MEPs: 1 up, 0 with errors, 0 timed out (archived)
  Cross-check errors: 0 missing, 0 unexpected

  CCM generation enabled:  Yes, 1s (Remote Defect detected: No)
                           CCM processing offloaded to hardware
  AIS generation enabled:  No
  Sending AIS:             No
  Receiving AIS:           No

  No packets sent/received

Domain DOWN0 (level 0), Service s10
Down MEP on TenGigE0/0/0/10.1 MEP-ID 2001
================================================================================
  Interface state: Up      MAC address: 008a.960f.c428
  Peer MEPs: 1 up, 0 with errors, 0 timed out (archived)
  Cross-check errors: 0 missing, 0 unexpected

  CCM generation enabled:  Yes, 1s (Remote Defect detected: No)
                           CCM processing offloaded to hardware
  AIS generation enabled:  No
  Sending AIS:             No
  Receiving AIS:           No

  Packet        Sent      Received
  ------  ----------  ---------------------------------------------------------
  DMM             10             0
  DMR              0            10
  SLM            100             0
  SLR              0           100
```

Synthetic Loss Measurement

The synthetic loss measurement mechanism defined in Y.1731 can only be used in point-to-point networks, and only works when there is sufficient flow of data traffic. The difficulties with the Y.1731 loss measurement mechanism was recognized across the industry and hence an alternative mechanism has been defined and standardized for measuring loss of traffic.

This alternative mechanism does not measure the loss of the actual data traffic, but instead injects synthetic CFM frames and measures the loss of these synthetic frames. You can perform a statistical analysis to give an approximation of the loss of data traffic. This technique is called Synthetic Loss Measurement (SLM). SLM has been included in the latest version of the Y.1731 standard. Use SLA to perform the following measurements:

- One-way loss (Source to Destination)
- One-way loss (Destination to Source)

SLM supports the following:

- All Layer 2 transport interfaces, such as physical, bundle interfaces, Layer2 sub-interfaces, pseudowire Head-end interfaces or attachment circuits. Transport network can be EVPN or BGP-MPLS.
- Up and Down MEPs.
- Transparent passing of the SLM packets through the MIP without punting it to the software.

Configuring Synthetic Loss Measurement

The following section describes how you can configure Synthetic Loss Measurement:

RP/0/RP0/CPU0:router **(config)# ethernet sla**

```
profile test-slm type cfm-synthetic-loss-measurement
  probe
   send packet every 1 seconds
   synthetic loss calculation packets 24
  !
  schedule
   every 3 minutes for 120 seconds
  !
  statistics
   measure one-way-loss-sd
    buckets size 1 probes
    buckets archive 5
   !
   measure one-way-loss-ds
    buckets size 1 probes
    buckets archive 5
!
!
!
!
interface TenGigE0/0/0/10.1 l2transport
encapsulation dot1q 1
ethernet cfm
  mep domain DOWN0 service s10 mep-id 2001
   sla operation profile test-slm target mep-id 6001
  !
```

Configuring an On-Demand Ethernet SLA Operation for CFM Synthetic Loss Measurement

To configure an on-demand Ethernet SLA operation for CFM synthetic loss measurement, use this command in privileged EXEC configuration mode:

RP/0/RP0/CPU0:router (config) # **ethernet sla on-demand operation type cfm-synthetic-loss-measurement probe domain D1 source interface TenGigE 0/0/0/0 target mac-address 2.3.4**

Running Configuration

```
RP/0/RP0/CPU0:router# show ethernet sla statistics on-demand id 1
Mon Sep 11 12:12:00.699 UTC
```

```
Source: Interface TenGigE0/0/0/10.1, Domain DOWN0
Destination: Target MEP-ID 6001
================================================================================
On-demand operation ID #1, packet type 'cfm-delay-measurement'
RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router# show running-config
Mon Sep 11 12:10:18.467 UTC
Building configuration...
!! IOS XR Configuration version = 6.4.1.14I
!! Last configuration change at Mon Sep 11 12:08:16 2017 by root
!
logging console disable
telnet vrf default ipv4 server max-servers 10
username root
group root-lr
group cisco-support
secret 5 $1$QJT3$94M5/wK5J0v/lpAu/wz31/
!
line console
exec-timeout 0 0
!
ethernet cfm
domain UP6 level 6 id null
  service s6 xconnect group g1 p2p p1 id number 6
   mip auto-create all ccm-learning
   continuity-check interval 1s
   mep crosscheck
    mep-id 4001
   !
  !
!
domain DOWN0 level 0 id null
 service s10 down-meps id number 10
   continuity-check interval 1s
   mep crosscheck
    mep-id 6001
   !
  !
!
!
profile test-slm type cfm-synthetic-loss-measurement
  probe
   send packet every 1 seconds
   synthetic loss calculation packets 24
  !
  schedule
   every 3 minutes for 120 seconds
  !
  statistics
   measure one-way-loss-sd
    buckets size 1 probes
    buckets archive 5
   !
   measure one-way-loss-ds
    buckets size 1 probes
    buckets archive 5
!

interface TenGigE0/0/0/10.1 l2transport
encapsulation dot1q 1
ethernet cfm
  mep domain DOWN0 service s10 mep-id 2001
   sla operation profile DMM target mep-id 6001
   sla operation profile test-slm target mep-id 6001
```

```
  !
!
!

interface FortyGigE0/0/1/2.1 l2transport
 encapsulation dot1q 1
 ethernet cfm
  mep domain UP6 service s6 mep-id 1
    sla operation profile DMM target mep-id 6001
    sla operation profile test-slm target mep-id 6001
  !
 !
!
l2vpn
 xconnect group g1
  p2p p1
   interface TenGigE0/0/0/10.1
   interface FortyGigE0/0/1/2.1
  !
 !
!
end
```

Verification

```
Round Trip Delay
~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 10
    Min: 0.009ms; Max: 0.010ms; Mean: 0.009ms; StdDev: 0.000ms


One-way Delay (Source->Dest)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 10
    Min: 1912765.961ms; Max: 1912765.961ms; Mean: 1912765.961ms; StdDev: -2147483.648ms


One-way Delay (Dest->Source)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 10
    Min: -1912765.952ms; Max: -1912765.951ms; Mean: -1912765.951ms; StdDev: -2147483.648ms


Round Trip Jitter
~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
```

```
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 9
    Min: 0.000ms; Max: 0.001ms; Mean: 0.000ms; StdDev: 0.000ms


One-way Jitter (Source->Dest)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 9
    Min: 0.000ms; Max: 0.000ms; Mean: 0.000ms; StdDev: 0.000ms


One-way Jitter (Dest->Source)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 probes per bucket

Bucket started at 12:11:19 UTC Mon 11 September 2017 lasting 10s
    Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
                   Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
    Result count: 9
    Min: 0.000ms; Max: 0.001ms; Mean: 0.000ms; StdDev: 0.000ms
```

CHAPTER 5

Integrated Routing and Bridging

The BVI is a virtual interface within the router that acts like a normal routed interface. The BVI does not support bridging itself, but acts as a gateway for the corresponding bridge-domain to a routed interface within the router.

Aside from supporting a configurable MAC address, a BVI supports only Layer 3 attributes, and has the following characteristics:

- Uses a MAC address taken from the local chassis MAC address pool, unless overridden at the BVI interface.
- Is configured as an interface type using the **interface bvi** command and uses an IPv4 address that is in the same subnet as the hosts on the segments of the bridged domain. The BVI also supports secondary addresses.
- The BVI identifier is independent of the bridge-domain identifier. These identifiers do not need to correlate like they do in Cisco IOS software.
- Is associated to a bridge group using the **routed interface bvi** command.

Supported Features on a BVI

- These interface commands are supported on a BVI:
  - **arp purge-delay**
  - arp timeout
  - **bandwidth** (The default is 10 Gbps and is used as the cost metric for routing protocols for the BVI)
  - **ipv4**

  - **ipv6**
  - **mac-address**
  - **shutdown**
- MTU configuration under BVI interface is not supported.

BVI Interface and Line Protocol States

Like typical interface states on the router, a BVI has both an Interface and Line Protocol state.

- The BVI interface state is Up when the following occurs:
  - The BVI interface is created.
  - The bridge-domain that is configured with the **routed interface bvi** command has at least one available active bridge port (Attachment circuit [AC] or pseudowire [PW]).

Note A BVI will be moved to the Down state if all of the bridge ports (Ethernet flow points [EFPs]) associated with the bridge domain for that BVI are down. However, the BVI will remain up if at least one bridgeport is up, even if all EFPs are down.

- These characteristics determine when the the BVI line protocol state is up:
  - The bridge-domain is in Up state.
  - The BVI IP address is not in conflict with any other IP address on another active interface in the router.

Prerequisites for Configuring IRB

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

Before configuring IRB, be sure that these tasks and conditions are met:

- Know the IP addressing and other Layer 3 information to be configured on the bridge virtual interface (BVI).
- Complete MAC address planning if you decide to override the common global MAC address for all BVIs.
- Be sure that the BVI network address is being advertised by running static or dynamic routing on the BVI interface.

Restrictions for Configuring IRB

Before configuring IRB, consider these restrictions:

- Only one BVI can be configured in any bridge domain.
- The same BVI can not be configured in multiple bridge domains.
- MTU configuration and fragmentation of packets is not supported on BVI interfaces.
- The following areas are *not* supported on the BVI:
    - Access Control Lists (ACLs). However, Layer 2 ACLs can be configured on each Layer 2 port of the bridge domain.
    - IP fast reroute (FRR)
    - TI-LFA
    - SR
    - LDP
    - NetFlow
    - MoFRR
    - Quality of Service (QoS)
    - Traffic mirroring
    - Unnumbered interface for BVI
    - Video monitoring (Vidmon)
    - IRB with 802.1ah (BVI and Provider Backbone Bridge (PBB) should not be configured in the same bridge domain).
    - PIM snooping. (Need to use selective flood.)
    - VRF-aware DHCP relay
- The following areas are *not* supported on the Layer2 bridging (with BVI):
    - Static mac entry configuration in Bridge.
    - Mac ageing configuration at global config mode.
    - MAC Learning Disable.
    - Vlan rewrite.
- QOS configuration on BVI interface is not supported for egress.
- Label allocation mode per-CE with BVI is not supported in an access network along with PE-CE protocols enabled.

How to Configure IRB

This section includes the following configuration tasks:

Configuring the Bridge Group Virtual Interface

To configure a BVI, complete the following steps.

Configuration Guidelines

Consider the following guidelines when configuring the BVI:

- The BVI must be assigned an IPv4 or IPv6 address that is in the same subnet as the hosts in the bridged segments.

SUMMARY STEPS

1. **configure**
2. **interface bvi** *identifier*
3. **arp purge-delay** *seconds*
4. **arp timeout** *seconds*
5. **bandwidth** *rate*
6. **end** or **commit**

DETAILED STEPS

Step 1 **configure**

Example:

```
Router# configure
```

Enters the global configuration mode.

Step 2 **interface bvi** *identifier*

Example:

```
Router(config)# interface bvi 1
```

Specifies or creates a BVI, where *identifier* is a number from 1 to 65535.

Step 3 **arp purge-delay** *seconds*

Example:

```
Router(config-if)#arp purge-delay 120
```

(Optional) Specifies the amount of time (in *seconds*) to delay purging of Address Resolution Protocol (ARP) table entries when the interface goes down.

The range is 1 to 65535. By default purge delay is not configured.

Step 4 **arp timeout** *seconds*

Example:

```
Router(config-if)# arp timeout 12200
```

(Optional) Specifies how long dynamic entries learned on the interface remain in the ARP cache.

The range is 30 to 2144448000 seconds. The default is 14,400 seconds (4 hours).

Step 5 **bandwidth** *rate*

Example:

```
Router(config-if)# bandwidth 1000000
```

(Optional) Specifies the amount of bandwidth (in kilobits per second) to be allocated on the interface. This number is used as the cost metric in routing protocols for the BVI.

The range is 0 to 4294967295. The default is 10000000 (10 Gbps).

Step 6 **end** or **commit**

Example:

```
Router(config-if)# end
```

or

```
Router(config-if)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?
  [cancel]:
  ```

  Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

  Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

  Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Layer 2 AC Interfaces

To configure the Layer 2 AC interfaces for routing by a BVI, complete the following steps.

SUMMARY STEPS

1. **configure**
2. **interface** [**HundredGigE** | **TenGigE**] **l2transport**
3. **end** or **commit**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface** [**HundredGigE** | **TenGigE**] **l2transport**

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/0.1 l2transport
```

Enables Layer 2 transport mode on a Gigabit Ethernet or 10-Gigabit Ethernet interface or subinterface and enters interface or subinterface configuration mode.

Step 3 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# end
```

or

```
RP/0/RP0/CPU0:router(config-if)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Bridge Group and Assigning Interfaces to a Bridge Domain

To configure a bridge group and assign interfaces to a bridge domain, complete the following steps.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **interface** [**HundredGigE | TenGigE**
6. **end** or **commit**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group 10
```

Creates a bridge group and enters L2VPN bridge group configuration mode.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain BD_1
```

Creates a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **interface** [**HundredGigE | TenGigE**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# interface HundredGigE 0/0/1/0.1
```

Associates the 100-Gigabit Ethernet or 10-Gigabit Ethernet interface with the specified bridge domain and enters L2VPN bridge group bridge domain attachment circuit configuration mode.

Repeat this step for as many interfaces as you want to associate with the bridge domain.

Step 6 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)# end
```

or

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Associating the BVI as the Routed Interface on a Bridge Domain

To associate the BVI as the routed interface on a bridge domain, complete the following steps.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **routed interface bvi** *identifier*
6. **end** or **commit**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group BG_test
```

Creates a bridge group and enters L2VPN bridge group configuration mode.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain 1
```

Creates a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **routed interface bvi** *identifier*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# routed interface bvi 1
```

Associates the specified BVI as the routed interface for the interfaces assigned to the bridge domain.

Step 6 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# end
```

or

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Displaying Information About a BVI

To display information about BVI status and packet counters, use the following commands:

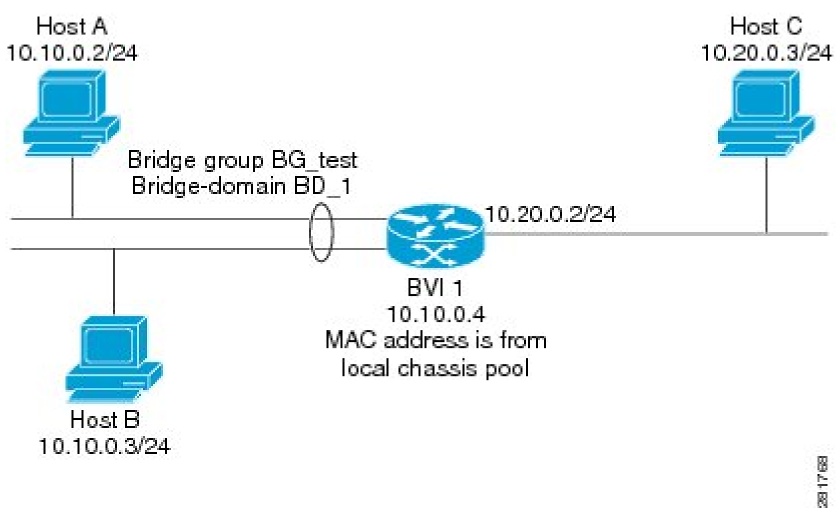
show interfaces bvi *identifier* [**accounting** | **brief** | **description** | **detail**]	Displays interface status, line protocol state, and packet counters for the specified BVI.
show adjacency bvi *identifier* [**detail** | **remote**]	Displays packet and byte transmit counters per adjacency to the specified BVI.
show l2vpn bridge-domain detail	Displays the reason that a BVI is down.

Additional Information on IRB

Packet Flows Using IRB

This figure shows a simplified functional diagram of an IRB implementation to describe different packet flows between Host A, B, and C. In this example, Host C is on a network with a connection to the same router. In reality, another router could be between Host C and the router shown.

Figure 9: IRB Packet Flows Between Hosts



When IRB is configured on a router, the following processing happens:

- ARP requests are resolved between the hosts and BVI that are part of the bridge domain.

- All packets from a host on a bridged interface go to the BVI if the destination MAC address matches the BVI MAC address. Otherwise, the packets are bridged.
- For packets destined for a host on a routed network, the BVI forwards the packets to the routing engine before sending them out a routed interface.
- All packets either from or destined to a host on a bridged interface go to the BVI first (unless the packet is destined for a host on the bridge domain).
- For packets that are destined for a host on a segment in the bridge domain that come in to the router on a routed interface, the BVI forwards the packet to the bridging engine, which forwards it through the appropriate bridged interface.

Packet Flows When Host A Sends to Host B on the Bridge Domain

When Host A sends data to Host B in the bridge domain on the 10.10.0.0 network, no routing occurs. The hosts are on the same subnet and the packets are bridged between their segment interfaces on the router.

Packet Flows When Host A Sends to Host C From the Bridge Domain to a Routed Interface

Using host information from this figure, the following occurs when Host A sends data to Host C from the IRB bridging domain to the routing domain:

- Host A sends the packet to the BVI (as long any ARP request the is resolved between the host and the BVI). The packet has the following information:
    - Source MAC address of host A.
    - Destination MAC address of the BVI.
- Since Host C is on another network and needs to be routed, the BVI forwards the packet to the routed interface with the following information:
    - IP source MAC address of Host A (10.10.0.2) is changed to the MAC address of the BVI (10.10.0.4).
    - IP destination address is the IP address of Host C (10.20.0.3).
- Interface 10.20.0.2 sees receipt of a packet from the routed BVI 10.10.0.4. The packet is then routed through interface 10.20.0.2 to Host C.

Packet Flows When Host C Sends to Host B From a Routed Interface to the Bridge Domain

Using host information from this figure, the following occurs when Host C sends data to Host B from the IRB routing domain to the bridging domain:

- The packet comes into the routing domain with the following information:
    - MAC source address—MAC of Host C.
    - MAC destination address—MAC of the 10.20.0.2 ingress interface.

- IP source address—IP address of Host C (10.20.0.3).
- IP destination address—IP address of Host B (10.10.0.3).

- When interface 10.20.0.2 receives the packet, it looks in the routing table and determines that the packet needs to be forwarded to the BVI at 10.10.0.4.
- The routing engine captures the packet that is destined for the BVI and forwards it to the BVI's corresponding bridge domain. The packet is then bridged through the appropriate interface if the destination MAC address for Host B appears in the bridging table, or is flooded on all interfaces in the bridge group if the address is not in the bridging table.

Configuration Examples for IRB

This section provides the following configuration examples:

Basic IRB Configuration: Example

The following example shows how to perform the most basic IRB configuration:

```
! Configure the BVI and its IPv4 address
!
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)#interface bvi 1
RP/0/RP0/CPU0:router(config-if)#ipv4 address 10.10.0.4 255.255.255.0
RP/0/RP0/CPU0:router(config-if))# exit
!
! Configure the Layer 2 AC interface
!
RP/0/RP0/CPU0:router(config)#interface HundredGigE 0/0/1/0 l2transport
RP/0/RP0/CPU0:router(config-if))# exit
!
! Configure the L2VPN bridge group and bridge domain and assign interfaces
!
RP/0/RP0/CPU0:router(config)#l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#bridge group 10
RP/0/RP0/CPU0:router(config-l2vpn-bg)#bridge-domain 1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#interface HundredGigE 0/0/1/0
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-if)# exit
!
! Associate a BVI to the bridge domain
!
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# routed interface bvi 1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# commit
```

IPv4 Addressing on a BVI Supporting Multiple IP Networks: Example

The following example shows how to configure secondary IPv4 addresses on a BVI that supports bridge domains for the 10.10.10.0/24, 10.20.20.0/24, and 10.30.30.0/24 networks. In this example, the BVI must have an address on each of the bridge domain networks:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)#interface bvi 1
```

```
RP/0/RP0/CPU0:router(config-if)#ipv4 address 10.10.10.4 255.255.255.0
RP/0/RP0/CPU0:router(config-if)#ipv4 address 10.20.20.4 255.255.255.0 secondary
RP/0/RP0/CPU0:router(config-if)#ipv4 address 10.30.30.4 255.255.255.0 secondary
RP/0/RP0/CPU0:router(config-if))# commit
```

IRB With BVI and VRRP Configuration: Example

This example shows a partial router configuration for the relevant configuration areas for IRB support of a BVI and VRRP:

Note VRRPv6 is also supported.

```
l2vpn
 bridge group IRB
  bridge-domain IRB-EDGE
   interface TenGigE0/0/0/8
!
   routed interface BVI 100
!
interface TenGigE0/0/0/8
  l2transport
!
interface BVI 100
 ipv4 address 10.21.1.1 255.255.255.0
!
router vrrp
 interface BVI 100
 address-family ipv4
 vrrp 1
 address 10.21.1.100
 priority 100
 !
```

CHAPTER 6

Configuring Link Bundling

The Link Bundling feature allows you to group multiple point-to-point links together into one logical link and provide higher bidirectional bandwidth, redundancy, and load balancing between two routers. A virtual interface is assigned to the bundled link. The component links can be dynamically added and deleted from the virtual interface.

The virtual interface is treated as a single interface on which one can configure an IP address and other software features used by the link bundle. Packets sent to the link bundle are forwarded to one of the links in the bundle.

A link bundle is simply a group of ports that are bundled together and act as a single link. The advantages of link bundles are as follows:

- Multiple links can span several line cards to form a single interface. Thus, the failure of a single link does not cause a loss of connectivity.
- Bundled interfaces increase bandwidth availability, because traffic is forwarded over all available members of the bundle. Therefore, traffic can flow on the available links if one of the links within a bundle fails. Bandwidth can be added without interrupting packet flow.

Cisco IOS XR software supports the following method of forming bundles of Ethernet interfaces:

- IEEE 802.3ad—Standard technology that employs a Link Aggregation Control Protocol (LACP) to ensure that all the member links in a bundle are compatible. Links that are incompatible or have failed are automatically removed from a bundle.

Compatible Characteristics of Ethernet Link Bundles

This list describes the properties of ethernet link bundles:

- The router supports mixed speed bundles. Mixed speed bundles allow member links of different bandwidth to be configured as active members in a single bundle. The ratio of the bandwidth for bundle members must not exceed 10. Also, the total weight of the bundle must not exceed 64. For example, 100Gbps link and 10Gbps links can be active members in a bundle and load-balancing on member links is based on bandwidth weightage.
- The weight of each bundle member is the ratio of its bandwidth to the lowest bandwidth member. Total weight of the bundle is the sum of weights or relative bandwidth of each bundle member. Since the weight for a bundle member is greater than or equal to 1 and less than or equal to 10, the total member of links in a bundle is less than 64 in mixed bundle case.
- Any type of Ethernet interfaces can be bundled, with or without the use of LACP (Link Aggregation Control Protocol).
- A single router can support maximum number of bundle interfaces. Link bundles of only physical interfaces are supported. Following are maximum numbers of bundle interfaces supported on NCS 540 variants:

  Medium Density XR NCS 540 - N540-24Z8Q2C-SYS, N540-28Z4C-SYS, N540X-ACC-SYS, N540-ACC-SYS

  Medium Density XR NCS 540 - N540-28Z4C-SYS-A, N540-28Z4C-SYS-D, N540X-16Z4G8Q2C-A, N540X-16Z4G8Q2C-D, N540X-16Z8Q2C-D, N540-12Z20G-SYS-A,N540-12Z20G-SYS-D, N540X-12Z16G-SYS-A, N540X-12Z16G-SYS-D

  Small Density XR NCS 540 - N540X-6Z18G-SYS-A, N540X-6Z18G-SYS-D, N540X-8Z16G-SYS-A, N540X-8Z16G-SYS-D

Table 3: Bundle Interfaces on NCS 540 Routers

Supported Features	Medium Density XR NCS 540		
Bundle Interfaces	256		
Maximum bundle members	64		
Bundle sub-interfaces	1024		
Layer2 Bundle Interfaces	1023		
hw-module profile bundle-scale command	Supported		

- The **hw-module profile bundle-scale <256/512/1024>** command is supported only on the following NCS 540 router variants:

  N540-24Z8Q2C-SYS, N540-28Z4C-SYS, N540X-ACC-SYS, N540-ACC-SYS

  The total number of supported bundle members with HQoS profile on Layer2 and Layer3 interfaces:

  - **hw-module profile bundle-scale 256** —Total bundle interfaces + total bundle sub-interfaces is 256
  - **hw-module profile bundle-scale 512** — Total bundle interfaces + total bundle sub-interfaces is 512

    - **hw-module profile bundle-scale 1024** — Total bundle interfaces + total bundle sub-interfaces is 1024

- Physical layer and link layer configuration are performed on individual member links of a bundle.
- Configuration of network layer protocols and higher layer applications is performed on the bundle itself.
- IPv4 and IPv6 addressing is supported on ethernet link bundles.
- A bundle can be administratively enabled or disabled.
- Each individual link within a bundle can be administratively enabled or disabled.
- Ethernet link bundles are created in the same way as Ethernet channels, where the user enters the same configuration on both end systems.
- QoS is supported and is applied proportionally on each bundle member.
- In case static MAC address is configured on a bundle-ether interface, the following limitations are applied:
    - Locally generated packets, such as ICMP, BGP, and so on, going out from the interface have the source MAC address as the statically configured MAC address.
    - Transit (forwarded) packets going out of the interface do not have the configured static MAC as source MAC address. In such a scenario, the upper 36-bits come from the system MAC address (or the original/dynamic MAC address) and the lower 12-bits come from the MAC address configured on the bundle. To check the dynamic pool of MAC addresses included, use the `show ethernet mac-allocation detail` command.

      For example, if the dynamic MAC address was 008A.9624.48D8 and the configured static MAC address is 0011.2222.ABCD. Then, the source MAC for transit (forwarded) traffic will be 008A.9624.4BCD.

Note This limitation can cause traffic blackholing for the transit traffic, in case there is L2 ACL applied for security purpose. In such case, it is necessary to add permit statement for both MAC addresses in the L2 ACL.

- Load balancing (the distribution of data between member links) is done by flow instead of by packet. Data is distributed to a link in proportion to the bandwidth of the link in relation to its bundle.
- All links within a single bundle must terminate on the same two systems.
- Bundled interfaces are point-to-point.
- A link must be in the up state before it can be in distributing state in a bundle.
- Only physical links can be bundle members.
- Multicast traffic is load balanced over the members of a bundle. For a given flow, the internal processes selects the member link, and the traffic for the flow is sent over that member.

Information About Configuring Link Bundling

To configure link bundling, you must understand the following concepts:

IEEE 802.3ad Standard

The IEEE 802.3ad standard typically defines a method of forming Ethernet link bundles.

For each link configured as bundle member, the following information is exchanged between the systems that host each end of the link bundle:

- A globally unique local system identifier
- An identifier (operational key) for the bundle of which the link is a member
- An identifier (port ID) for the link
- The current aggregation status of the link

This information is used to form the link aggregation group identifier (LAG ID). Links that share a common LAG ID can be aggregated. Individual links have unique LAG IDs.

The system identifier distinguishes one router from another, and its uniqueness is guaranteed through the use of a MAC address from the system. The bundle and link identifiers have significance only to the router assigning them, which must guarantee that no two links have the same identifier, and that no two bundles have the same identifier.

The information from the peer system is combined with the information from the local system to determine the compatibility of the links configured to be members of a bundle.

The MAC address of the first link attached to a bundle becomes the MAC address of the bundle itself. The bundle uses this MAC address until that link (the first link attached to the bundle) is detached from the bundle, or until the user configures a different MAC address. The bundle MAC address is used by all member links when passing bundle traffic. Any unicast or multicast addresses set on the bundle are also set on all the member links.

Note We recommend that you avoid modifying the MAC address, because changes in the MAC address can affect packet forwarding.

Link Bundle Configuration Overview

The following steps provide a general overview of the link bundle configuration. Keep in mind that a link must be cleared of all previous network layer configuration before it can be added to a bundle:

1. In global configuration mode, create a link bundle. To create an Ethernet link bundle, enter the **interface Bundle-Ether** command.
2. Assign an IP address and subnet mask to the virtual interface using the **ipv4 address** command.
3. Add interfaces to the bundle you created in Step 1 with the **bundle id** command in the interface configuration submode.

You can add up to 32 links to a single bundle.

4. You can optionally implement 1:1 link protection for the bundle by setting the **bundle maximum-active links** command to 1. Performing this configuration causes the highest-priority link in the bundle to become active and the second-highest-priority link to become the standby. (The link priority is based on the value of the **bundle port-priority** command.) If the active link fails, the standby link immediately becomes the active link.

Note A link is configured as a member of a bundle from the interface configuration submode for that link.

Link Switchover

By default, a maximum of 64 links in a bundle can actively carry traffic. If one member link in a bundle fails, traffic is redirected to the remaining operational member links.

You can optionally implement 1:1 link protection for a bundle by setting the **bundle maximum-active links** command to 1. By doing so, you designate one active link and one or more dedicated standby links. If the active link fails, a switchover occurs and a standby link immediately becomes active, thereby ensuring uninterrupted traffic.

If the active and standby links are running LACP, you can choose between an IEEE standard-based switchover (the default) or a faster proprietary optimized switchover. If the active and standby links are not running LACP, the proprietary optimized switchover option is used.

Regardless of the type of switchover you are using, you can disable the wait-while timer, which expedites the state negotiations of the standby link and causes a faster switchover from a failed active link to the standby link.

LACP Fallback

The LACP Fallback feature allows an active LACP interface to establish a Link Aggregation Group (LAG) port-channel before the port-channel receives the Link Aggregation and Control Protocol (LACP) protocol data units (PDU) from its peer. With the LACP Fallback feature configured, the router allows the server to bring up the LAG, before receiving any LACP PDUs from the server, and keeps one port active. This allows the server to establish a connection to PXE server over one Ethernet port, download its boot image and then continue the booting process. When the server boot process is complete, the server fully forms an LACP port-channel.

Configuring Ethernet Link Bundles

This section describes how to configure an Ethernet link bundle.

Note In order for an Ethernet bundle to be active, you must perform the same configuration on both connection endpoints of the bundle.

Tip You can programmatically perform the configuration using `openconfig-if-aggregate.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco NCS 540 Series Routers*.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address mask*
4. **bundle minimum-active bandwidth** *kbps*
5. **bundle minimum-active links** *links*
6. **bundle maximum-active links** *links* [**hot-standby**]
7. **exit**
8. **interface TenGigE** interface-path-id
9. **bundle id** *bundle-id* [**mode** {**active** | **on** | **passive**}]
10. **bundle port-priority** *priority*
11. **no shutdown**
12. **exit**
13. **bundle id** *bundle-id* [**mode** {**active** | **passive** | **on**}]
14. **exit**
15. **exit**
16. Perform Step 1 through Step 15 on the remote end of the connection.
17. **show bundle Bundle-Ether** *bundle-id*
18. **show lacp Bundle-Ether** *bundle-id*

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 3
```

Creates a new Ethernet link bundle with the specified bundle-id. The range is 1 to 65535.

Step 3 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
```

Assigns an IP address and subnet mask to the virtual interface using the **ipv4 address** configuration subcommand.

Note • Only a Layer 3 bundle interface requires an IP address.

Step 4 **bundle minimum-active bandwidth** *kbps*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000
```

(Optional) Sets the minimum amount of bandwidth required before a user can bring up a bundle.

Step 5 **bundle minimum-active links** *links*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2
```

(Optional) Sets the number of active links required before you can bring up a specific bundle.

Step 6 **bundle maximum-active links** *links* [**hot-standby**]

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle maximum-active links 1 hot-standby
```

(Optional) Implements 1:1 link protection for the bundle, which causes the highest-priority link in the bundle to become active and the second-highest-priority link to become the standby. Also, specifies that a switchover between active and standby LACP-enabled links is implemented per a proprietary optimization.

The **bundle port-priority** command determines the priority of the active and standby links for the bundle.

Step 7 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# exit
```

Exits interface configuration submode for the Ethernet link bundle.

Step 8 **interface TenGigE** interface-path-id

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/1
```

Enters interface configuration mode for the specified interface.

Enter the **TenGigE** keyword to specify the interface type. Replace the *interface-path-id* argument with the node-id in the *rack*/*slot*/*module* format.

Step 9 **bundle id** *bundle-id* [**mode** {**active** | **on** | **passive**}]

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle id 3 mode on
```

Adds the link to the specified bundle.

To enable active or passive LACP on the bundle, include the optional **mode active** or **mode passive** keywords in the command string.

To add the link to the bundle without LACP support, include the optional **mode on** keywords with the command string.

Note • If you do not specify the **mode** keyword, the default mode is **on (**LACP is not run over the port).

Step 10 **bundle port-priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle port-priority 1
```

(Optional) If you set the **bundle maximum-active links** command to 1, you must also set the priority of the active link to the highest priority (lowest value) and the standby link to the second-highest priority (next lowest value). For example, you can set the priority of the active link to 1 and the standby link to 2.

Step 11 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-if)# no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 12 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# exit
```

Exits interface configuration submode for the Ethernet interface.

Step 13 **bundle id** *bundle-id* [**mode** {**active** | **passive** | **on**}]

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/1/0

RP/0/RP0/CPU0:router(config-if)# bundle id 3

RP/0/RP0/CPU0:router(config-if)# bundle port-priority 2

RP/0/RP0/CPU0:router(config-if)# no shutdown

RP/0/RP0/CPU0:router(config-if)# exit

RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/1/0

RP/0/RP0/CPU0:router(config-if)# bundle id 3

RP/0/RP0/CPU0:router(config-if)# no shutdown

RP/0/RP0/CPU0:router(config-if)# exit
```

(Optional) Repeat Step 8 through Step 11 to add more links to the bundle.

Step 14 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# exit
```

Exits interface configuration mode.

Step 15 **exit**

Example:

```
RP/0/RP0/CPU0:router(config)# exit
```

Exits global configuration mode.

Step 16 Perform Step 1 through Step 15 on the remote end of the connection.

Brings up the other end of the link bundle.

Step 17 **show bundle Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router# show bundle Bundle-Ether 3
```

(Optional) Shows information about the specified Ethernet link bundle.

Step 18 **show lacp Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router# show lacp Bundle-Ether 3
```

(Optional) Shows detailed information about LACP ports and their peers.

Configuring LACP Fallback

This section describes how to configure the LACP Fallback feature.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address mask*
4. **end** or **commit**
5. **show bundle infrastructure database ma bdl-info Bundle-e1010 | inc***text*
6. **show bundle infrastructure database ma bdl-info Bundle-e1015 | inc***text*

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 3
```

Creates and names a new Ethernet link bundle.

Step 3 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle lacp-fallback timeout 4
```

Enables the LACP Fallback feature.

Step 4 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# commit
```

Saves configuration changes.

Step 5 **show bundle infrastructure database ma bdl-info Bundle-e1010 | inc***text*

Example:

```
RP/0/RP0/CPU0:router# show bundle infrastructure database ma bdl-info Bundle-e1010 | inc "fallback"
```

(Optional) Shows the MA information of the bundle manager.

Step 6 **show bundle infrastructure database ma bdl-info Bundle-e1015 | inc***text*

Example:

```
RP/0/RP0/CPU0:router# show bundle infrastructure database ma bdl-info Bundle-e1015 | inc "fallback"
```

(Optional) Shows the MA information of the bundle manager.

VLANs on an Ethernet Link Bundle

802.1Q VLAN subinterfaces can be configured on 802.3ad Ethernet link bundles. Keep the following information in mind when adding VLANs on an Ethernet link bundle:

Note The memory requirement for bundle VLANs is slightly higher than standard physical interfaces.

To create a VLAN subinterface on a bundle, include the VLAN subinterface instance with the **interface Bundle-Ether** command, as follows:

interface Bundle-Ether *interface-bundle-id.subinterface*

After you create a VLAN on an Ethernet link bundle, all VLAN subinterface configuration is supported on that link bundle.

VLAN subinterfaces can support multiple Layer 2 frame types and services, such as Ethernet Flow Points - EFPs) and Layer 3 services.

Layer 2 EFPs are configured as follows:

```
interface bundle-ether instance.subinterface l2transport. encapsulation dot1q xxxxx
```

Layer 3 VLAN subinterfaces are configured as follows:

```
interface bundle-ether instance.subinterface, encapsulation dot1q xxxxx
```

Note The difference between the Layer 2 and Layer 3 interfaces is the **l2transport** keyword. Both types of interfaces use **dot1q encapsulation**.

Configuring VLAN over Bundles

This section describes how to configure a VLAN bundle. The creation of a VLAN bundle involves three main tasks:

SUMMARY STEPS

1. Create an Ethernet bundle.
2. Create VLAN subinterfaces and assign them to the Ethernet bundle.
3. Assign Ethernet links to the Ethernet bundle.

DETAILED STEPS

Step 1 Create an Ethernet bundle.

Step 2 Create VLAN subinterfaces and assign them to the Ethernet bundle.

Step 3 Assign Ethernet links to the Ethernet bundle.

These tasks are describe in detail in the procedure that follows.

Note In order for a VLAN bundle to be active, you must perform the same configuration on both ends of the bundle connection.

SUMMARY STEPS

1. **configure**

2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address mask*
4. **bundle minimum-active bandwidth** *kbps*
5. **bundle minimum-active links** *links*
6. **bundle maximum-active links** *links* [**hot-standby**]
7. **exit**
8. **interface Bundle-Ether** *bundle-id.vlan-id*
9. **encapsulation dot1q***vlan-id*
10. **ipv4 address** *ipv4-address mask*
11. **no shutdown**
12. **exit**
13. Repeat Step 9 through Step 12 to add more VLANS to the bundle you created in Step 2.
14. **end** or **commit**
15. **exit**
16. **exit**
17. **configure**
18. **interface** {**TenGigE** | **FortyGigE** | **HundredGigE**}*interface-path-id*

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router#(config)# interface Bundle-Ether 3
```

Creates and names a new Ethernet link bundle.

Step 3 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
```

Assigns an IP address and subnet mask to the virtual interface using the **ipv4 address** configuration subcommand.

Step 4 **bundle minimum-active bandwidth** *kbps*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000
```

(Optional) Sets the minimum amount of bandwidth required before a user can bring up a bundle.

Step 5 **bundle minimum-active links** *links*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2
```

(Optional) Sets the number of active links required before you can bring up a specific bundle.

Step 6 **bundle maximum-active links** *links* [**hot-standby**]

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle maximum-active links 1 hot-standby
```

(Optional) Implements 1:1 link protection for the bundle, which causes the highest-priority link in the bundle to become active and the second-highest-priority link to become the standby. Also, specifies that a switchover between active and standby LACP-enabled links is implemented per a proprietary optimization.

The **bundle port-priority** command determines the priority of the active and standby links for the bundle.

Step 7 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# exit
```

Exits the interface configuration submode.

Step 8 **interface Bundle-Ether** *bundle-id.vlan-id*

Example:

```
RP/0/RP0/CPU0:router#(config)# interface Bundle-Ether 3.1
```

Creates a new VLAN, and assigns the VLAN to the Ethernet bundle you created in Step 2.

Replace the *bundle-id* argument with the *bundle-id* you created in Step 2.

Replace the *vlan-id* with a subinterface identifier.

Range is from 1 to 4094 inclusive (0 and 4095 are reserved).

Range is from 1 to 4093 inclusive (0, 4094, and 4095 are reserved).

Note When you include the *.vlan-id* argument with the **interface Bundle-Ether** *bundle-id* command, you enter subinterface configuration mode.

Step 9 **encapsulation dot1q***vlan-id*

Example:

```
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 100
```

Sets the Layer 2 encapsulation of an interface.

Step 10 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router#(config-subif)# ipv4 address 10.1.2.3/24
```

Assigns an IP address and subnet mask to the subinterface.

Step 11 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router#(config-subif)# no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 12 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# exit
```

Exits subinterface configuration mode for the VLAN subinterface.

Step 13 Repeat Step 9 through Step 12 to add more VLANS to the bundle you created in Step 2.

(Optional) Adds more subinterfaces to the bundle.

Step 14 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# end
```

or

```
RP/0/RP0/CPU0:router(config-subif)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before
  exiting(yes/no/cancel)?
  [cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 15 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# end
```

Exits interface configuration mode.

Step 16 **exit**

Example:

```
RP/0/RP0/CPU0:router(config)# exit
```

Exits global configuration mode.

Step 17 **configure**

Example:

```
RP/0/RP0/CPU0:router # configure
```

Enters global configuration mode.

Step 18 **interface** {**TenGigE** | **FortyGigE** | **HundredGigE**}*interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode for the Ethernet interface you want to add to the Bundle.

Enter the **GigabitEthernet** or **TenGigE** keyword to specify the interface type. Replace the *interface-path-id* argument with the node-id in the rack/slot/module format.

Note A VLAN bundle is not active until you add an Ethernet interface on both ends of the link bundle.

LACP Short Period Time Intervals

As packets are exchanged across member links of a bundled interface, some member links may slow down or time-out and fail. LACP packets are exchanged periodically across these links to verify the stability and reliability of the links over which they pass. The configuration of short period time intervals, in which LACP packets are sent, enables faster detection and recovery from link failures.

Short period time intervals are configured as follows:

- In milliseconds
- In increments of 100 milliseconds
- In the range 100 to 1000 milliseconds
- The default is 1000 milliseconds (1 second)
-
- Up to 1280 packets per second (pps)

After 6missed packets, the link is detached from the bundle.

When the short period time interval is *not* configured, LACP packets are transmitted over a member link every 30 seconds by default.

When the short period time interval is configured, LACP packets are transmitted over a member link once every 1000 milliseconds (1 second) by default. Optionally, both the transmit and receive intervals can be configured to less than 1000 milliseconds, independently or together, in increments of 100 milliseconds (100, 200, 300, and so on).

When you configure a custom LACP short period *transmit* interval at one end of a link, you must configure the same time period for the *receive* interval at the other end of the link.

Note You must always configure the *transmit* interval at both ends of the connection before you configure the *receive* interval at either end of the connection. Failure to configure the *transmit* interval at both ends first results in route flapping (a route going up and down continuously). When you remove a custom LACP short period, you must do it in reverse order. You must remove the *receive* intervals first and then the *transmit* intervals.

Configuring the Default LACP Short Period Time Interval

This section describes how to configure the default short period time interval for sending and receiving LACP packets on a Gigabit Ethernet interface. This procedure also enables the LACP short period.

SUMMARY STEPS

1. **configure**
2. **interface HundredGigE***interface-path*
3. **bundle id** *number* **mode active**
4. **lacp period short**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface HundredGigE***interface-path*

Example:

```
RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/0/1/0
```

Creates a Gigabit Ethernet interface and enters interface configuration mode.

Step 3 **bundle id** *number* **mode active**

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle id 1 mode active
```

Specifies the bundle interface and puts the member interface in active mode.

Step 4 **lacp period short**

Example:

```
RP/0/RP0/CPU0:router(config-if)# lacp period short
```

Configures a short period time interval for the sending and receiving of LACP packets, using the default time period of 1000 milliseconds or 1 second.

Example

This example shows how to configure the LACP short period time interval to the default time of 1000 milliseconds (1 second):

```
config
interface HundredGigE 0/0/1/0
   bundle id 1 mode active
   lacp period short
   commit
```

The following example shows how to configure custom LACP short period transmit and receive intervals to *less than* the default of 1000 milliseconds (1 second):

```
config
interface HundredGigE 0/0/1/0
   bundle id 1 mode active
   lacp period short
   commit


config
interface HundredGigE 0/0/1/0
    lacp period short transmit 100
   commit


config
interface HundredGigE 0/0/1/0
   lacp period short receive 100
   commit
```

Configuring Custom LACP Short Period Time Intervals

This section describes how to configure custom short period time intervals (less than 1000 milliseconds) for sending and receiving LACP packets on a Gigabit Ethernet interface.

Note You must always configure the *transmit* interval at both ends of the connection before you configure the *receive* interval at either end of the connection. Failure to configure the *transmit* interval at both ends first results in route flapping (a route going up and down continuously). When you remove a custom LACP short period, you must do it in reverse order. You must remove the *receive* intervals first and then the *transmit* intervals.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address mask*
4. **bundle minimum-active bandwidth** *kbps*
5. **bundle minimum-active links** *links*
6. **bundle maximum-active links** *links*
7. **exit**
8. **interface Bundle-Ether** *bundle-id.vlan-id*
9. **dot1q vlan** *vlan-id*
10. **ipv4 address** *ipv4-address mask*
11. **no shutdown**
12. **exit**
13. Repeat Step 7 through Step 12 to add more VLANs to the bundle you created in Step 2.
14. **exit**
15. **exit**
16. **show ethernet trunk bundle-ether** *instance*
17. **configure**
18. **interface** {**HundredGigE** } *interface-path-id*
19. **bundle id** *bundle-id* [**mode** {**active** | **on** | **passive**}]
20. **no shutdown**
21. Repeat Step 19 through Step 21 to add more Ethernet interfaces to the VLAN bundle.
22. Perform Step 1 through Step 23 on the remote end of the VLAN bundle connection.
23. **show bundle Bundle-Ether** *bundle-id* [**reasons**]
24. **show ethernet trunk bundle-ether** *instance*

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 3
```

Creates and names a new Ethernet link bundle.

Step 3 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
```

Assigns an IP address and subnet mask to the virtual interface using the **ipv4 address** configuration subcommand.

Step 4 **bundle minimum-active bandwidth** *kbps*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000
```

(Optional) Sets the minimum amount of bandwidth required before a user can bring up a bundle.

Step 5 **bundle minimum-active links** *links*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2
```

(Optional) Sets the number of active links required before you can bring up a specific bundle.

Step 6 **bundle maximum-active links** *links*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle maximum-active links 1
```

(Optional) Designates one active link and one link in standby mode that can take over immediately for a bundle if the active link fails (1:1 protection).

Note

- The default number of active links allowed in a single bundle is 8.
- If the **bundle maximum-active** command is issued, then only the highest-priority link within the bundle is active. The priority is based on the value from the **bundle port-priority** command, where a lower value is a higher priority. Therefore, we recommend that you configure a higher priority on the link that you want to be the active link.

Step 7 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# exit
```

Exits the interface configuration submode.

Step 8 **interface Bundle-Ether** *bundle-id.vlan-id*

Example:

```
RP/0/RP0/CPU0:router#(config)# interface Bundle-Ether 3.1
```

Creates a new VLAN, and assigns the VLAN to the Ethernet bundle you created in Step 2.

Replace the *bundle-id* argument with the *bundle-id* you created in Step 2.

Replace the *vlan-id* with a subinterface identifier. Range is from 1 to 4093 inclusive (0, 4094, and 4095 are reserved).

Note

- When you include the *vlan-id* argument with the **interface Bundle-Ether** *bundle-id* command, you enter subinterface configuration mode.

Step 9 **dot1q vlan** *vlan-id*

Example:

```
RP/0/RP0/CPU0:router(config-subif)# dot1q vlan 10
```

Assigns a VLAN to the subinterface.

Replace the *vlan-id* argument with a subinterface identifier. Range is from 1 to 4093 inclusive (0, 4094, and 4095 are reserved).

Step 10 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 10.1.2.3/24
```

Assigns an IP address and subnet mask to the subinterface.

Step 11 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 12 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# exit
```

Exits subinterface configuration mode for the VLAN subinterface.

Step 13 Repeat Step 7 through Step 12 to add more VLANs to the bundle you created in Step 2.

(Optional) Adds more subinterfaces to the bundle.

Step 14 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# exit
```

Exits interface configuration mode.

Step 15 **exit**

Example:

```
RP/0/RP0/CPU0:router(config)# exit
```

Exits global configuration mode.

Step 16 **show ethernet trunk bundle-ether** *instance*

Example:

```
RP/0/RP0/CPU0:router# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Step 17 **configure**

Example:

```
RP/0/RP0/CPU0:router # configure
```

Enters global configuration mode.

Step 18 **interface** {**HundredGigE** } *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/0/1/0
```

Enters the interface configuration mode for the Ethernet interface you want to add to the Bundle.

Enter the **HundredGigE** keyword to specify the interface type. Replace the *interface-path-id* argument with the node-id in the rack/slot/module format.

Note
- A VLAN bundle is not active until you add an Ethernet interface on both ends of the link bundle.

Step 19 **bundle id** *bundle-id* [**mode** {**active** | **on** | **passive**}]

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle-id 3
```

Adds an Ethernet interface to the bundle you configured in Step 2 through Step 13.

To enable active or passive LACP on the bundle, include the optional **mode active** or **mode passive** keywords in the command string.

To add the interface to the bundle without LACP support, include the optional **mode on** keywords with the command string.

Step 20 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-if)# no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 21 Repeat Step 19 through Step 21 to add more Ethernet interfaces to the VLAN bundle.

—

Step 22 Perform Step 1 through Step 23 on the remote end of the VLAN bundle connection.

Brings up the other end of the link bundle.

Step 23 **show bundle Bundle-Ether** *bundle-id* [**reasons**]

Example:

```
RP/0/RP0/CPU0:router# show bundle Bundle-Ether 3 reasons
```

(Optional) Shows information about the specified Ethernet link bundle.

The **show bundle Bundle-Ether** command displays information about the specified bundle. If your bundle has been configured properly and is carrying traffic, the State field in the **show bundle Bundle-Ether** command output will show the number “4,” which means the specified VLAN bundle port is “distributing.”

Step 24 **show ethernet trunk bundle-ether** *instance*

Example:

```
RP/0/RP0/CPU0:router# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

CHAPTER 7

Configuring Traffic Mirroring

This module describes the configuration of the traffic mirroring feature. Traffic mirroring is sometimes called port mirroring, or switched port analyzer (SPAN).

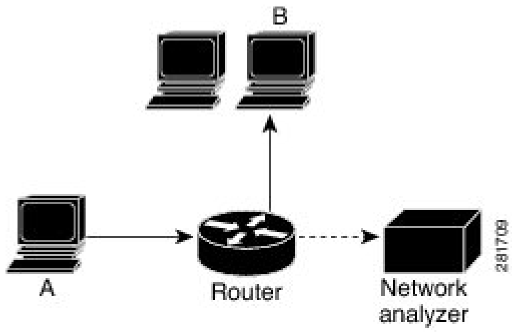
Introduction to Traffic Mirroring

Traffic mirroring, sometimes called port mirroring or Switched Port Analyzer (SPAN), is a Cisco proprietary feature that enables you to monitor network traffic passing in or out of a set of ports. You can then pass this traffic to a destination port on the same router.

Traffic mirroring copies traffic from one or more source ports and sends the copied traffic to one or more destinations for analysis by a network analyzer or other monitoring device. Traffic mirroring does not affect the flow of traffic on the source interfaces or sub-interfaces. It allows the mirrored traffic to be sent to a destination interface or sub-interface.

For example, you can attach a traffic analyzer to the router and capture Ethernet traffic that is sent by host A to host B.

Figure 10: Traffic Mirroring Operation



When local traffic mirroring is enabled, the traffic analyzer gets directly attached to the port that is configured to receive a copy of every packet that host A sends. This port is called a traffic mirroring port.

Note

- From Release 7.2.1, traffic mirroring is introduced on Cisco NCS 5700 line cards.
- From Release 7.4.2, you can mirror incoming (Rx) and outgoing (Tx) traffic from the source ports to separate destinations on Cisco NC57 line cards. During a session, you can configure one destination port for incoming traffic and one for outgoing traffic.

Traffic Mirroring Types

The following types of traffic mirroring are supported:

- **Local traffic mirroring**: This is the most basic form of traffic mirroring. The network analyzer or sniffer is attached directly to the destination interface. In other words, all monitored ports are located on the same router as the destination port.
- **Remote traffic mirroring**: The network analyzer is reached through a GRE tunnel over an IP network.

Note A copy of every packet includes the Layer 2 header if the ethernet keyword is configured. As this renders the mirrored packets unroutable, the end point of the GRE tunnel must be the network analyzer.

- **ACL-based traffic mirroring**: Traffic is mirrored based on the configuration of the interface ACL.

  You can mirror traffic based on the definition of an interface access control list. When you are mirroring Layer 3 traffic, the ACL is configured using the **ipv4 access-list** or the **ipv6 access-list** command with the **capture** option. The **permit** and **deny** commands determine the behavior of regular traffic. The **capture** option designates the packet is to be mirrored to the destination port, and it is supported only on permit type of access control entries (ACEs).

Note Prior to Release 6.5.1, ACL-based traffic mirroring required the use of UDK (User-Defined TCAM Key) with the **enable-capture** option so that the **capture** option can be configured in the ACL.

- **Encapsulated remote SPAN (ERSPAN)**: ERSPAN enables generic routing encapsulation (GRE) for all captured traffic and allows it to be extended across Layer 3 domains.
- **SPAN to File**: SPAN to File is an extension of the pre-existing SPAN feature that allows network packets to be mirrored to a file instead of an interface. This helps in the analysis of the packets at a later stage.
- **File Mirroring**: File mirroring feature enables the router to copy files or directories automatically from `/harddisk:/mirror` location in active RP to `/harddisk:/mirror` location in standby RP or RSP without user intervention or EEM scripts.

ERSPAN

Encapsulated Remote Switched Port Analyzer (ERSPAN) transports mirrored traffic over an IP network. The traffic is encapsulated at the source router and is transferred across the network. The packet is decapsulated at the destination router and then sent to the destination interface.

ERSPAN involves mirroring traffic through a GRE tunnel to a remote site. For more information on configuring the GRE tunnel that is used as the destination for the monitor sessions, see the chapter *Configuring GRE Tunnels*.

Restrictions

Generic Restrictions

The following are the generic restriction(s) related to traffic mirroring:

- Partial mirroring and sampled mirroring are not supported.
- Sub-interface configured as source interface is not supported on SPAN.
- The destination bundle interfaces flap when:
  - both the mirror source and destination are bundle interfaces in LACP mode and
  - mirror packets next-hop is a router or a switch instead of a traffic analyzer.

  This behavior is observed due to a mismatch of LACP packets on the next-hop bundle interface due to the mirroring of LACP packets on the source bundle interface.
- Both SPAN and ERSPAN features cannot be configured on a router simultaneously. Either SPAN or ERSPAN feature can be configured on the same router.

SPAN Restrictions

The following restrictions apply to SPAN:

- SPAN only supports port-level source interfaces.

- SPAN over pseudowire is not supported on the NCS 540 Routers.

ERSPAN Restrictions

The following restrictions apply to ERSPAN:

- The value of ERSPAN session-ID is always zero. IOS XR Command for configuring ERPAN is not available.
- ERSPAN next-hop must have ARP resolved. Any other traffic or protocol will trigger ARP.
- ERSPAN cannot travel over MPLS.
  - Additional routers may encapsulate in MPLS.
- ERSPAN decapsulation is not supported.
- ERSPAN does not work if the GRE next hop is reachable over sub-interface. For ERSPAN to work, the next hop must be reachable over the main interface.

SPAN-ACL Restrictions

The following restrictions apply to SPAN-ACL:

- SPAN-ACL is only supported in the Rx direction, that is, in the ingress direction v4 or v6 ACL.
- MPLS traffic cannot be captured with SPAN-ACL.
  - ACL for any MPLS traffic is not supported.

Configure Traffic Mirroring

These tasks describe how to configure traffic mirroring:

Configure Remote Traffic Mirroring

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **monitor-session** *session-name*

Example:

```
RP/0/RP0/CPU0:router(config)# monitor-session mon1 ethernet
RP/0/RP0/CPU0:router(config-mon)#
```

Defines a monitor session and enters monitor session configuration mode.

Step 3 **destination interface** *tunnel-ip*

Example:

```
RP/0/RP0/CPU0:router(config-mon)# destination interface tunnelip3
```

Specifies the destination subinterface to which traffic is replicated.

Step 4 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-mon)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits monitor session configuration mode and returns to global configuration mode.

Step 5 **interface** *type number*

Example:

```
RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/0/1/0
```

Enters interface configuration mode for the specified source interface. The interface number is entered in *rack*/*slot*/*module*/*port* notation. For more information about the syntax for the router, use the question mark (?) online help function.

Step 6 **monitor-session** *session-name* **ethernet direction rx-onlyport-only**

Example:

```
RP/0/RP0/CPU0:router(config-if)# monitor-session mon1 ethernet
direction rx-only port-only
```

Specifies the monitor session to be used on this interface. Use the **direction** keyword to specify that only ingress or egress traffic is mirrored.

Step 7 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# end
```

or

```
RP/0/RP0/CPU0:router(config-if)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting (yes/no/cancel)?
  [cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8 **show monitor-session [session-name]** **status** [**detail**] [**error**]

Example:

```
RP/0/RP0/CPU0:router# show monitor-session
```

Displays information about the traffic mirroring session.

Example

This example shows the basic configuration for traffic mirroring with physical interfaces.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# monitor-session ms1
RP/0/RP0/CPU0:router(config-mon)# destination interface HundredGigE0/2/0/15
RP/0/RP0/CPU0:router(config-mon)# commit
```

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface TenGigE0/2/0/19
RP/0/RP0/CPU0:router(config-if)# monitor-session ms1 ethernet direction rx-only port-level
RP/0/RP0/CPU0:router(config-if)# commit
```

This example shows sample output of the show monitor-session command with the status keyword:

```
RP/0/RSP0/CPU0:router# show monitor-session status
Monitor-session cisco-rtp1
Destination interface HundredGigE 0/5/0/38
================================================================================
Source Interface Dir Status
--------------------- ---- ----------------------------------------------------
TenGigE0/5/0/4 Both Operational
TenGigE0/5/0/17 Both Operational
RP/0/RSP0/CPU0:router# show monitor-session status detail
Monitor-session sess1
Destination interface is not configured
Source Interfaces
-----------------
TenGigE0/1/0/0
Direction: Both
ACL match: Disabled
Portion: Full packet
Status: Not operational (destination interface not known).
TenGigE0/1/0/1
Direction: Both
ACL match: Disabled
Portion: First 100 bytes
```

```
RP/0/RSP0/CPU0:router# show monitor-session status error
Monitor-session ms1
Destination interface TenGigE0/2/0/15 is not configured
```

```
=================================================================================
Source Interface Dir Status
--------------------- ---- ----------------------------------------------------
Monitor-session ms2
Destination interface is not configured
=================================================================================
Source Interface Dir Status
--------------------- ---- ----------------------------------------------------
RP/0/RP0/CPU0:router# show monitor-session test status
Monitor-session test (ipv4)
Destination Nexthop 255.254.254.4
==========================================================================================
Source Interface Dir Status
------------------------------------------------------------------------------------------
Gi0/0/0/2.2 Rx Not operational (source same as destination)
Gi0/0/0/2.3 Rx Not operational (Destination not active)
Gi0/0/0/2.4 Rx Operational
Gi0/0/0/4 Rx Error: see detailed output for explanation
RP/0/RP0/CPU0:router# show monitor-session test status error
Monitor-session test
Destination Nexthop ipv4 address 255.254.254.4
===============================================================
Source Interface Status
---------------------------------------------------------------
Gi0/0/0/4 < Error: FULL Error Details >
```

Attaching the Configurable Source Interface

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface** *type number*

Example:

```
RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/0/1/0
```

Enters interface configuration mode for the specified source interface. The interface number is entered in *rack*/*slot*/*module*/*port* notation. For more information about the syntax for the router, use the question mark (?) online help function.

Step 3 **ipv4 access-group** *acl-name* {**ingress** | **egress**}

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 access-group acl1 ingress
```

Controls access to an interface.

Step 4 **monitor-session** *session-name* **ethernet direction rx-onlyport-level acl**

Example:

```
RP/0/RP0/CPU0:router(config-if)# monitor-session mon1 ethernet direction rx-only port-level acl
RP/0/RP0/CPU0:router(config-if-mon)#
```

Attaches a monitor session to the source interface and enters monitor session configuration mode.

Note **rx-only** specifies that only ingress traffic is replicated.

Step 5 **acl**

Example:

```
RP/0/RP0/CPU0:router(config-if-mon)# acl
```

Specifies that the traffic mirrored is according to the defined ACL.

Note If an ACL is configured by name, then this step overrides any ACL that may be configured on the interface.

Step 6 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-if-mon)# exit
RP/0/RP0/CPU0:router(config-if)#
```

Exits monitor session configuration mode and returns to interface configuration mode.

Step 7 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# end
```

or

```
RP/0/RP0/CPU0:router(config-if)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting (yes/no/cancel)?
  [cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8 **show monitor-session [session-name]** **status** [**detail**] [**error**]

Example:

```
RP/0/RP0/CPU0:router# show monitor-session status
```

Displays information about the monitor session.

Configuring UDF-Based ACL for Traffic Mirroring

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**udf** *udf-name* **header** {**inner** | **outer**} {**l2** | **l3** | **l4**} **offset** *offset-in-bytes* **length** *length-in-bytes*
	Example:
	`RP/0/RP0/CPU0:router(config)# udf udf3 header outer l4 offset 0 length 1 (config-mon)#`
	Example:
	`RP/0/RP0/CPU0:router(config)# udf udf3 header inner l4 offset 10 length 2 (config-mon)#`
	Example:
	`RP/0/RP0/CPU0:router(config)# udf udf3 header outer l4 offset 50 length 1 (config-mon)#`
		Configures individual UDF definitions. You can specify the name of the UDF, the networking header from which offset, and the length of data to be extracted.
		The **inner** or **outer** keywords indicate the start of the offset from the unencapsulated Layer 3 or Layer 4 headers, or if there is an encapsulated packet, they indicate the start of offset from the inner L3/L4.
		Note The maximum offset allowed, from the start of any header, is 63 bytes
		The **length** keyword specifies, in bytes, the length from the offset. The range is from 1 to 4.

Step 3	**ipv4 access-list** *acl-name* **Example:** `RP/0/RP0/CPU0:router(config))# ipv4 access-list acl1`	Creates ACL and enters IP ACL configuration mode. The length of the *acl-name* argument can be up to 64 characters.

Step 4	**permit** *regular-ace-match-criteria* **udf** *udf-name1 value1 ... udf-name8 value8*
	Example:
	`RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit ipv4 any any udf udf1 0x1234 0xffff udf3 0x56 0xff capture`
	`RP/0/RP0/CPU0:router(config-ipv4-acl)# 30 permit ipv4 any any dscp af11 udf udf5 0x22 0x22 capture`
		Configures ACL with UDF match.

	Command or Action	Purpose
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-ipv4-acl)# exit`	Exits IP ACL configuration mode and returns to global configuration mode.
Step 6	**interface***type number* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/0/1/0`	Configures interface and enters interface configuration mode.
Step 7	**ipv4 access-group** *acl-name* **ingress** **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 access-group acl1 ingress`	Applies access list to an interface.
Step 8	**commit** **Example:** `RP/0/RP0/CPU0:router(config-if)# commit`	Applies access list to an interface.

Additional Information on Traffic Mirroring

Traffic Mirroring Terminology

- Ingress Traffic — Traffic that comes into the router.
- Egress Traffic — Traffic that goes out of the router.
- Source (SPAN) interface — An interface that is monitored using the SPAN feature.
- Source port—A port that is monitored with the use of traffic mirroring. It is also called a monitored port.
- Destination port—A port that monitors source ports, usually where a network analyzer is connected. It is also called a monitoring port.
- Monitor session—A designation for a collection of SPAN configurations consisting of a single destination and, potentially, one or many source interfaces.

Characteristics of Source Port

A source port, also called a monitored port, is a routed port that you monitor for network traffic analysis. In a single traffic mirroring session, you can monitor source port traffic. The Cisco NCS540 Series router support a maximum of up to 800 source ports.

A source port has these characteristics:

- It can be any data port type, such as Bundle Interface, 100 Gigabit Ethernet, or 10 Gigabit Ethernet.

> **Note**
> - Bridge group virtual interfaces (BVIs) are not supported.
> - Bundle members cannot be used as source ports.

- Each source port can be monitored in only one traffic mirroring session.
- When a port is used as a source port, the same port cannot be used as a destination port.
- Each source port can be configured with a direction (ingress, egress, or both) to monitor local traffic mirroring. Remote traffic mirroring is supported both in the ingress and egress directions. For bundles, the monitored direction applies to all physical ports in the group.

Characteristics of Monitor Session

A monitor session is a collection of traffic mirroring configurations consisting of a single destination and, potentially, many source interfaces. For any given monitor session, the traffic from the source interfaces (called *source ports*) is sent to the monitoring port or destination port. If there are more than one source port in a monitoring session, the traffic from the several mirrored traffic streams is combined at the destination port. The result is that the traffic that comes out of the destination port is a combination of the traffic from one or more source ports.

Monitor sessions have these characteristics:

- A single router can have a maximum of four monitor sessions. However, both SPAN and CFM share common mirror profiles. If you configure SPAN and CFM together on the router, the maximum number of monitor sessions may reduce to two.
- A single monitor session can have only one destination port.
- A single destination port can belong to only one monitor session.
- A monitor session can have a maximum of 800 source ports, as long as the maximum number of source ports from all monitoring sessions does not exceed 800.

Characteristics of Destination Port

Each session must have a destination port that receives a copy of the traffic from the source ports.

A destination port has these characteristics:

- A destination port must reside on the same router as the source port for local traffic mirroring. For remote mirroring, the destination is always a GRE tunnel.
- A destination port for local mirroring can be any Ethernet physical port, EFP, GRE tunnel interface, or bundle interface. It can be a Layer 2 or Layer 3 transport interface.

> **Note** Bundle members cannot be used as destination ports.

- A destination port on router cannot be a VLAN subinterface.
- At any one time, a destination port can participate in only one traffic mirroring session. A destination port in one traffic mirroring session cannot be a destination port for a second traffic mirroring session. In other words, no two monitor sessions can have the same destination port.
- A destination port cannot also be a source port.

Traffic Mirroring Configuration Examples

This section contains examples of how to configure traffic mirroring:

Traffic Mirroring with Physical Interfaces (Local): Example

This example shows the basic configuration for traffic mirroring with physical interfaces.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# monitor-session ms1
RP/0/RP0/CPU0:router(config-mon)# destination interface HundredGigE0/0/1/0
RP/0/RP0/CPU0:router(config-mon)# commit

RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface TenGigE0/0/0/0
RP/0/RP0/CPU0:router(config-if)# monitor-session ms1 ethernet direction rx-only port-level
RP/0/RP0/CPU0:router(config-if)# commit
```

Viewing Monitor Session Status: Example

This example shows sample output of the **show monitor-session** command with the **status** keyword:

```
RP/0/RP0/CPU0:router# show monitor-session status

Monitor-session cisco-rtp1
Destination interface HundredGigE 0/0/1/0
================================================================================
Source Interface   Dir  Status
--------------------- ---- ----------------------------------------------------
TenGigE0/0/0/4       Both Operational
TenGigE0/0/0/17      Both Operational

RP/0/RSP0/CPU0:router# show monitor-session status detail

Monitor-session sess1
 Destination interface is not configured
 Source Interfaces
 -----------------
 TenGigE0/0/0/0
  Direction: Both
  ACL match: Disabled
  Portion:  Full packet
  Status:  Not operational (destination interface not known).
 TenGigE0/0/0/1
  Direction: Both
  ACL match: Disabled
```

```
  Portion:  First 100 bytes

RP/0/RP0/CPU0:router# show monitor-session status error

Monitor-session ms1
Destination interface TenGigE0/0/0/15 is not configured
================================================================================
Source Interface   Dir  Status
--------------------- ---- ----------------------------------------------------

Monitor-session ms2
Destination interface is not configured
================================================================================
Source Interface   Dir  Status
--------------------- ---- ----------------------------------------------------


RP/0/RP0/CPU0:router# show monitor-session test status

Monitor-session test (ipv4)

Destination Nexthop 255.254.254.4
=========================================================================================
Source Interface   Dir         Status
-----------------------------------------------------------------------------------------
Gi0/0/0/2.2     Rx    Not operational (source same as destination)
Gi0/0/0/2.3      Rx    Not operational (Destination not active)
Gi0/0/0/2.4      Rx    Operational
Gi0/0/0/4       Rx    Error: see detailed output for explanation

RP/0/RP0/CPU0:router# show monitor-session test status error

Monitor-session test
Destination Nexthop ipv4 address 255.254.254.4
===============================================================
Source Interface     Status
---------------------------------------------------------------
Gi0/0/0/4    < Error: FULL Error Details >
```

Troubleshooting Traffic Mirroring

When you encounter any issue with traffic mirroring, begin troubleshooting by checking the output of the **show monitor-session status** command. This command displays the recorded state of all sessions and source interfaces:

```
# show monitor-session status
Monitor-session 5
rx destination interface tunnel-ip5
tx destination is not specified
================================================================================
Source Interface  Dir  Status
--------------------- ---- ----------------------------------------------------
Te0/0/0/23 (port) Rx   Operational
```

In the preceding example, the line marked as `<Session status>` can indicate one of these configuration errors:

Session Status	Explanation
Session is not configured globally	The session does not exist in global configuration. Review the **sho** command output and ensure that a session with a correct name has configured.
Destination interface <intf> (<down-state>)	The destination interface is not in Up state in the Interface Manage can verify the state using the **show interfaces** command. Check th configuration to determine what might be keeping the interface from up (for example, a sub-interface needs to have an appropriate encaps configured).

The <Source interface status> can report these messages:

Source Interface Status	Explanation
Operational	Everything appears to be working correctly in traffic mirroring PI. follow up with the platform teams in the first instance, if mirroring operating as expected.
Not operational (Session is not configured globally)	The session does not exist in global configuration. Check the **show** command output to ensure that a session with the right name has b configured.
Not operational (destination not known)	The session exists, but it either does not have a destination interface s or the destination interface named for the session does not exist. For e if the destination is a sub-interface that has not been created.
Not operational (source same as destination)	The session exists, but the destination and source are the same inte traffic mirroring does not work.
Not operational (destination not active)	The destination interface or pseudowire is not in the Up state. See corresponding *Session status* error messages for suggested resoluti
Not operational (source state <down-state>)	The source interface is not in the Up state. You can verify the state the **show interfaces** command. Check the configuration to see wha be keeping the interface from coming up (for example, a sub-interfa to have an appropriate encapsulation configured).
Error: see detailed output for explanation	Traffic mirroring has encountered an error. Run the **show monitor status detail** command to display more information.

The **show monitor-session status detail** command displays full details of the configuration parameters and any errors encountered. For example:

RP/0/RP0/CPU0:router **show monitor-session status detail**

```
Monitor-session sess1
 Destination interface is not configured
 Source Interfaces
 -----------------
 TenGigE0/0/0/1
  Direction: Both
  ACL match: Disabled
  Portion:  Full packet
```

```
  Status:  Not operational (destination interface not known)
 TenGigE0/0/0/2
  Direction: Both
  ACL match: Disabled
  Portion:  First 100 bytes
  Status: Not operational (destination interface not known). Error: 'Viking SPAN PD' detected
 the 'warning' condition 'PRM connection
          creation failure'.
Monitor-session foo
 Destination next-hop TenGigE 0/0/0/0
 Source Interfaces
 -----------------
 TenGigE 0/0/0/1.100:
  Direction: Both
  Status:  Operating
 TenGigE 0/0/0/2.200:
  Direction: Tx
  Status:  Error: <blah>

Monitor session bar
 No destination configured
 Source Interfaces
 -----------------
 TenGigE 0/0/0/3.100:
  Direction: Rx
  Status:  Not operational(no destination)
```

Here are additional trace and debug commands:

```
RP/0/RP0/CPU0:router# show monitor-session platform trace ?

 all   Turn on all the trace
 errors Display errors
 events Display interesting events

RP/0/RP0/CPU0:router# show monitor-session trace ?

 process Filter debug by process

#



RP/0/RP0/CPU0:router# debug monitor-session process all

RP/0/RP0/CPU0:router# debug monitor-session process ea

RP/0/RP0/CPU0:router# debug monitor-session process ma

RP/0/RP0/CPU0:router# show monitor-session process mgr

 detail  Display detailed output
 errors  Display only attachments which have errors
 internal Display internal monitor-session information
 |     Output Modifiers

RP/0/RP0/CPU0:router# show monitor-session status

RP/0/RP0/CPU0:router# show monitor-session status errors

RP/0/RP0/CPU0:router# show monitor-session status internal
```

Verifying UDF-based ACL

Use the **show monitor-session status detail** command to verify the configuration of UDF on ACL.

```
RP/0/RP0/CPU0:leaf1# show monitor-session 1 status detail

Fri May 12 19:40:39.429 UTC
Monitor-session 1
  Destination interface tunnel-ip3
  Source Interfaces
  -----------------
  TenGigE0/0/0/15
    Direction:  Rx-only
    Port level: True
    ACL match:  Enabled
    Portion:    Full packet
    Interval:   Mirror all packets
    Status:     Not operational (destination not active)
```

CHAPTER 8

Configuring Virtual Loopback and Null Interfaces

This module describes the configuration of loopback and null interfaces. Loopback and null interfaces are considered virtual interfaces.

A virtual interface represents a logical packet switching entity within the router. Virtual interfaces have a global scope and do not have an associated location. Virtual interfaces have instead a globally unique numerical ID after their names. Examples are Loopback 0, Loopback 1, and Loopback 99999. The ID is unique per virtual interface type to make the entire name string unique such that you can have both Loopback 0 and Null 0.

Loopback and null interfaces have their control plane presence on the active route switch processor (RSP). The configuration and control plane are mirrored onto the standby RSP and, in the event of a failover, the virtual interfaces move to the ex-standby, which then becomes the newly active RSP.

Information About Configuring Virtual Interfaces

To configure virtual interfaces, you must understand the following concepts:

Virtual Loopback Interface Overview

A virtual loopback interface is a virtual interface with a single endpoint that is always up. Any packet transmitted over a virtual loopback interface is immediately received by the same interface. Loopback interfaces emulate a physical interface.

In Cisco IOS XR Software, virtual loopback interfaces perform these functions:

- Loopback interfaces can act as a termination address for routing protocol sessions. This allows routing protocol sessions to stay up even if the outbound interface is down.
- You can ping the loopback interface to verify that the router IP stack is working properly.

In applications where other routers or access servers attempt to reach a virtual loopback interface, you must configure a routing protocol to distribute the subnet assigned to the loopback address.

Packets routed to the loopback interface are rerouted back to the router or access server and processed locally. IP packets routed out to the loopback interface but not destined to the loopback interface are dropped. Under these two conditions, the loopback interface can behave like a null interface.

Prerequisites for Configuring Virtual Interfaces

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

Configuring Virtual Loopback Interfaces

This task explains how to configure a basic loopback interface.

Restrictions

The IP address of a loopback interface must be unique across all routers on the network. It must not be used by another interface on the router, and it must not be used by an interface on any other router on the network.

SUMMARY STEPS

1. **configure**
2. **interface loopback** *instance*
3. **ipv4 address** *ip-address*
4. **end** or **commit**
5. **show interfaces***type instance*

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface loopback** *instance*

Example:

```
RP/0/RP0/CPU0:router#(config)# interface Loopback 3
```

Enters interface configuration mode and names the new loopback interface.

Step 3 **ipv4 address** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 100.100.100.69 255.255.255.255
```

Assigns an IP address and subnet mask to the virtual loopback interface using the **ipv4 address** configuration command.

Step 4 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# end
```

or

```
RP/0/RP0/CPU0:router(config-if)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5 **show interfaces***type instance*

Example:

```
RP/0/RP0/CPU0:router# show interfaces Loopback0
```

(Optional) Displays the configuration of the loopback interface.

Example

This example shows how to configure a loopback interface:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface Loopback0
RP/0/RP0/CPU0:router(config-if)# ipv4 address 100.100.100.69 255.255.255.255
RP/0/RP0/CPU0:router(config-if)# ipv6 address 100::69/128
RP/0/RP0/CPU0:router(config-if)# end
Uncommitted changes found, commit them? [yes]: yes
RP/0/RP0/CPU0:router# show interfaces Loopback0

Loopback0 is up, line protocol is up
  Interface state transitions: 1
  Hardware is Loopback interface(s)
  Internet address is 100.100.100.69/32
  MTU 1500 bytes, BW 0 Kbit
     reliability Unknown, txload Unknown, rxload Unknown
  Encapsulation Loopback,  loopback not set,
  Last link flapped 01:57:47
  Last input Unknown, output Unknown
  Last clearing of "show interface" counters Unknown
  Input/output data rate is disabled.
```

Null Interface Overview

A null interface functions similarly to the null devices available on most operating systems. This interface is always up and can never forward or receive traffic; encapsulation always fails. The null interface provides an alternative method of filtering traffic. You can avoid the overhead involved with using access lists by directing undesired network traffic to the null interface.

The only interface configuration command that you can specify for the null interface is the **ipv4 unreachables** command. With the **ipv4 unreachables** command, if the software receives a non-broadcast packet destined for itself that uses a protocol it does not recognize, it sends an Internet Control Message Protocol (ICMP) protocol unreachable message to the source. If the software receives a datagram that it cannot deliver to its ultimate destination because it knows of no route to the destination address, it replies to the originator of that datagram with an ICMP host unreachable message. By default **ipv4 unreachables** command is enabled. If we do not want ICMP to send protocol unreachable, then we need to configure using the **ipv4 icmp unreachable disable** command.

The Null 0 interface is created by default during boot process and cannot be removed. The **ipv4 unreachables** command can be configured for this interface, but most configuration is unnecessary because this interface just discards all the packets sent to it.

The Null 0 interface can be displayed with the **show interfaces null0** command.

Configuring Null Interfaces

This task explains how to configure a basic null interface.

SUMMARY STEPS

1. **configure**
2. **interface null 0**
3. **ipv4 icmp unreachables disable**
4. **end** or **commit**
5. **show interfaces null 0**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface null 0**

Example:

```
RP/0/RP0/CPU0:router(config)# interface null 0
```

Enters the null 0 interface configuration mode.

Step 3 **ipv4 icmp unreachables disable**

Example:

```
RP/0/RP0/CPU0:router(config-null0)# ipv4 icmp unreachables disable
```

This command disables the generation of IPv4 Internet Control Message Protocol (ICMP) unreachable messages.

Step 4 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-null0)# end
```

or

```
RP/0/RP0/CPU0:router(config-null0)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before
  exiting(yes/no/cancel)?
  [cancel]:
  ```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5 **show interfaces null 0**

Example:

```
RP/0/RP0/CPU0:router# show interfaces null 0
```

Verifies the configuration of the null interface.

Example

This example shows how to configure a null interface:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface Null 0
RP/0/RP0/CPU0:router(config-null0)# ipv4 icmp unreachables disable
RP/0/RP0/CPU0:router(config-null0)# end
Uncommitted changes found, commit them? [yes]: yes
RP/0/RP0/CPU0:router# show interfaces Null 0

Null0 is up, line protocol is up
```

```
Interface state transitions: 1
Hardware is Null interface
Internet address is Unknown
MTU 1500 bytes, BW 0 Kbit
reliability 255/255, txload Unknown, rxload Unknown
Encapsulation Null, loopback not set,
Last link flapped 4d20h
Last input never, output never
Last clearing of "show interface" counters 05:42:04
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
0 packets input, 0 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 0 broadcast packets, 0 multicast packets
0 packets output, 0 bytes, 0 total output drops
Output 0 broadcast packets, 0 multicast packets
```

Configuring Virtual IPv4 Interfaces

This task explains how to configure an IPv4 virtual interface.

SUMMARY STEPS

1. **configure**
2. **ipv4 virtual address** *ipv4-*
3. **end** or **commit**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipv4 virtual address** *ipv4-*

Example:

```
RP/0/RP0/CPU0:router(config)# ipv4 virtual address 10.3.32.154/8
```

Defines an IPv4 virtual address for the management Ethernet interface.

Note While configuring the IPv4 virtual address, ensure that you match the IP address on the Management interface in the same network.

Step 3 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-null0)# end
```

or

```
RP/0/RP0/CPU0:router(config-null0)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before
exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Example

This is an example for configuring a virtual IPv4 interface:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# ipv4 virtual address 10.3.32.154/8
RP/0/RP0/CPU0:router(config-null0)# commit
```

CHAPTER 9

Configuring 802.1Q VLAN Interfaces

A VLAN is a group of devices on one or more LANs that are configured so that they can communicate as if they were attached to the same wire, when in fact they are located on a number of different LAN segments. VLANs are very flexible for user and host management, bandwidth allocation, and resource optimization because they are based on logical grouping instead of physical connections.

The IEEE 802.1Q protocol standard addresses the problem of dividing large networks into smaller parts so broadcast and multicast traffic does not consume more bandwidth than necessary. The standard also helps provide a higher level of security between segments of internal networks.

802.1Q Tagged Frames

The IEEE 802.1Q tag-based VLAN uses an extra tag in the MAC header to identify the VLAN membership of a frame across bridges. This tag is used for VLAN and quality of service (QoS) priority identification. The VLAN ID associates a frame with a specific VLAN and provides the information that switches must process the frame across the network. A tagged frame is four bytes longer than an untagged frame and contains two bytes of Tag Protocol Identifier (TPID) residing within the type and length field of the Ethernet frame and two bytes of Tag Control Information (TCI) which starts after the source address field of the Ethernet frame.

Configuring 802.1Q VLAN Interfaces

A VLAN is a group of devices on one or more LANs that are configured so that they can communicate as if they were attached to the same wire, when in fact they are located on a number of different LAN segments. VLANs are very flexible for user and host management, bandwidth allocation, and resource optimization because they are based on logical grouping instead of physical connections.

The IEEE 802.1Q protocol standard addresses the problem of dividing large networks into smaller parts so broadcast and multicast traffic does not consume more bandwidth than necessary. The standard also helps provide a higher level of security between segments of internal networks.

802.1Q Tagged Frames

The IEEE 802.1Q tag-based VLAN uses an extra tag in the MAC header to identify the VLAN membership of a frame across bridges. This tag is used for VLAN and quality of service (QoS) priority identification. The VLAN ID associates a frame with a specific VLAN and provides the information that switches must process the frame across the network. A tagged frame is four bytes longer than an untagged frame and contains two bytes of Tag Protocol Identifier (TPID) residing within the type and length field of the Ethernet frame and two bytes of Tag Control Information (TCI) which starts after the source address field of the Ethernet frame.

Information About Configuring 802.1Q VLAN Interfaces

To configure 802.1Q VLAN interfaces, you must understand these concepts:

Subinterfaces

Subinterfaces are logical interfaces created on a hardware interface. These software-defined interfaces allow for segregation of traffic into separate logical channels on a single hardware interface as well as allowing for better utilization of the available bandwidth on the physical interface.

Subinterfaces are distinguished from one another by adding an extension on the end of the interface name and designation. For instance, the Ethernet subinterface 23 on the physical interface designated TenGigE 0/0/0/0 would be indicated by TenGigE 0/0/0/0.23.

Before a subinterface is allowed to pass traffic it must have a valid tagging protocol encapsulation and VLAN identifier assigned. All Ethernet subinterfaces always default to the 802.1Q VLAN encapsulation. However, the VLAN identifier must be explicitly defined.

Subinterface MTU

The subinterface maximum transmission unit (MTU) is inherited from the physical interface with an additional four bytes allowed for the 802.1Q VLAN tag. By default subinterface inherits MTU of physical interface if the MTU is not configured. We can have maximum 3 different MTU for a subinterface per NPU. .

EFPs

An Ethernet Flow Point (EFP) is a Metro Ethernet Forum (MEF) term describing abstract router architecture. An EFP is implemented by an Layer 2 subinterface with a VLAN encapsulation. The term EFP is used synonymously with an VLAN tagged L2 subinterface. .

Layer 2 VPN on VLANs

The Layer 2 Virtual Private Network (L2VPN) feature enables Service Providers (SPs) to provide Layer 2 services to geographically disparate customer sites.

The configuration model for configuring VLAN attachment circuits (ACs) is similar to the model used for configuring basic VLANs, where the user first creates a VLAN subinterface, and then configures that VLAN in subinterface configuration mode. To create an AC, you need to include the **l2transport** keyword in the **interface** command string to specify that the interface is a Layer 2 interface.

VLAN ACs support these modes of L2VPN operation:

- Basic Dot1Q AC—The AC covers all frames that are received and sent with a specific VLAN tag.
- QinQ AC—The AC covers all frames received and sent with a specific outer VLAN tag and a specific inner VLAN tag. QinQ is an extension to Dot1Q that uses a stack of two tags.

Each VLAN on a CE-to-PE link can be configured as a separate L2VPN connection (using either VC type 4 or VC type 5).

How to Configure 802.1Q VLAN Interfaces

This section contains the following procedures:

Configuring 802.1Q VLAN Subinterfaces

This task explains how to configure 802.1Q VLAN subinterfaces. To remove these subinterfaces, see the "Removing an 802.1Q VLAN Subinterface" section.

SUMMARY STEPS

1. **configure**
2. **interface {TenGigE | FortyGigE | HundredGigE | Bundle-Ether}** *interface-path-id.subinterface*
3. **encapsulation dot1q**
4. **ipv4 address** *ip-address mask*
5. **exit**
6. Repeat Step 2 through Step 5 to define the rest of the VLAN subinterfaces.
7. **end** or **commit**
8. **show ethernet trunk bundle-ether** *instance*

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface {TenGigE | FortyGigE | HundredGigE | Bundle-Ether}** *interface-path-id.subinterface*

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/0/0/4.10
```

Enters subinterface configuration mode and specifies the interface type, location, and subinterface number.

- Replace the *interface-path-id* argument with one of the following instances:
- Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack*/*slot*/*module*/*port*, and a slash between values is required as part of the notation.
- Ethernet bundle instance. Range is from 1 through 65535.
- Replace the *subinterface* argument with the subinterface value. Range is from 0 through 2147483647.
- Naming notation is *interface-path-id.subinterface*, and a period between arguments is required as part of the notation.

Step 3 **encapsulation dot1q**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 100
```

Sets the Layer 2 encapsulation of an interface.

Step 4 **ipv4 address** *ip-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 178.18.169.23/24
```

Assigns an IP address and subnet mask to the subinterface.

- Replace *ip-address* with the primary IPv4 address for an interface.
- Replace *mask* with the mask for the associated IP subnet. The network mask can be specified in either of two ways:
- The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address.
- The network mask can be indicated as a slash (/) and number. For example, /8 indicates that the first 8 bits of the mask are ones, and the corresponding bits of the address are network address.

Step 5 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# exit
```

(Optional) Exits the subinterface configuration mode.

- The **exit** command is not explicitly required.

Step 6 Repeat Step 2 through Step 5 to define the rest of the VLAN subinterfaces.

—

Step 7 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config)# end
```

or

```
RP/0/RP0/CPU0:router(config)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8 **show ethernet trunk bundle-ether** *instance*

Example:

```
RP/0/RP0/CPU0:router# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Verification

This example shows how to verify the configuration of Ethernet interfaces :

```
# show ethernet trunk be 1020 Wed May 17 16:43:32.804 EDT

Trunk                                  Sub types        Sub states
Interface      St Ly    MTU    Subs       L2      L3       Up    Down  Ad-Down
BE1020        Up L3   9100       3         3       0        3      0        0

Summary                          3         3       0        3      0        0
```

Configuring an Attachment Circuit on a VLAN

Use the following procedure to configure an attachment circuit on a VLAN.

SUMMARY STEPS

1. **configure**
2. **interface [GigabitEthernet | TenGigE | Bundle-Ether | FortyGigE]** *interface-path*] *id.subinterface* **l2transport**
3. **encapsulation dot1q 100**
4. **end** or **commit**
5. **show interfaces** [**GigabitEthernet** |**FortyGigE**|**Bundle-Ether** | **TenGigE**] *interface-path-id.subinterface*

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **interface [GigabitEthernet | TenGigE | Bundle-Ether | FortyGigE]** *interface-path*] *id.subinterface* **l2transport**

Example:

```
RP/0//CPU0:router(config)# interface TenGigE 0/0/0/1.1 l2transport
```

Enters subinterface configuration and specifies the interface type, location, and subinterface number.

- Replace the *interface-path-id* argument with one of the following instances:
- Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack*/*slot*/*module*/*port*, and a slash between values is required as part of the notation.
- Ethernet bundle instance. Range is from 1 through 65535.
- Replace the *subinterface* argument with the subinterface value. Range is from 0 through 4095.
- Naming notation is *instance.subinterface*, and a period between arguments is required as part of the notation.
- You must include the **l2transport** keyword in the command string; otherwise, the configuration creates a Layer 3 subinterface rather that an AC.

Step 3 **encapsulation dot1q 100**

Example:

```
RP/0//CPU0:router (config-subif)# encapsulation dot1q 100
```

Sets the Layer 2 encapsulation of an interface.

Note The **dot1q vlan** command is replaced by the **encapsulation dot1q** command. It is still available for backward-compatibility, but only for Layer 3 interfaces.

Step 4 **end** or **commit**

Example:

```
RP/0//CPU0:router(config-if-l2)# end
```

or

```
RP/0//CPU0:router(config-if-l2)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?
  [cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5 **show interfaces** [**GigabitEthernet** |**FortyGigE**|**Bundle-Ether** | **TenGigE**] *interface-path-id.subinterface*

Example:

```
RP/0//CPU0:router# show interfaces TenGigE 0/0/0/3.1
```

(Optional) Displays statistics for interfaces on the router.

Removing an 802.1Q VLAN Subinterface

This task explains how to remove 802.1Q VLAN subinterfaces that have been previously configured using the Configuring 802.1Q VLAN subinterfaces section in this module.

SUMMARY STEPS

1. **configure**
2. **no interface** {**TenGigE** | **FortyGigE** | **HundredGigE** | **Bundle-Ether**] *interface-path-id.subinterface*
3. Repeat Step 2 to remove other VLAN subinterfaces.
4. **end** or **commit**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **no interface** {**TenGigE** | **FortyGigE** | **HundredGigE** | **Bundle-Ether**] *interface-path-id.subinterface*

Example:

```
RP/0/RP0/CPU0:router(config)# no interface TenGigE 0/0/0/4.10
```

Removes the subinterface, which also automatically deletes all the configuration applied to the subinterface.

- Replace the *instance* argument with one of the following instances:
- Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack*/*slot*/*module*/*port*, and a slash between values is required as part of the notation.
- Ethernet bundle instance. Range is from 1 through 65535.
- Replace the *subinterface* argument with the subinterface value. Range is from 0 through 2147483647.

Naming notation is *instance.subinterface*, and a period between arguments is required as part of the notation.

Step 3 Repeat Step 2 to remove other VLAN subinterfaces.

—

Step 4 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config)# end
```

or

```
RP/0/RP0/CPU0:router(config)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?
  [cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

CHAPTER 10

Configuring GRE Tunnels

Generic Routing Encapsulation (GRE) is a tunneling protocol that provides a simple generic approach to transport packets of one protocol over another protocol by means of encapsulation. This module provides information about how to configure a GRE tunnel.

Configuring GRE Tunnels

Generic Routing Encapsulation (GRE) is a tunneling protocol that provides a simple generic approach to transport packets of one protocol over another protocol by means of encapsulation. This module provides information about how to configure a GRE tunnel.

Configuring GRE Tunnels

Tunneling provides a mechanism to transport packets of one protocol within another protocol. Generic Routing Encapsulation (GRE) is a tunneling protocol that provides a simple generic approach to transport packets of one protocol over another protocol with encapsulation. GRE encapsulates a payload, that is, an inner packet that needs to be delivered to a destination network inside an outer IP packet. The GRE tunnel behave as virtual point-to-point link that have two endpoints identified by the tunnel source and tunnel destination address. The tunnel endpoints send payloads through GRE tunnels by routing encapsulated packets through intervening IP networks. Other IP routers along the way do not parse the payload (the inner packet); they only parse the outer IP packet as they forward it towards the GRE tunnel endpoint. Upon reaching the tunnel endpoint, GRE encapsulation is removed and the payload is forwarded to the packet's ultimate destination.

Guidelines and Restrictions for Configuring GRE Tunnels

The following restrictions apply while configuring GRE tunnels:

- The router supports up to 500 GRE tunnels.
- Only up to 16 unique source IP addresses are supported for the tunnel source.

- 2-pass to Single-pass migration, which means converting the same GRE tunnel, is not possible in a single configuration step. You must first delete the 2-pass tunnel and then add the Single-pass tunnel.
- Configurable MTU is not supported on Single-pass GRE interface, but supported on 2-pass GRE interface.
-

Configuration Example

Configuring a GRE tunnel involves creating a tunnel interface and defining the tunnel source and destination. This example shows how to configure a GRE tunnel between Router1 and Router2. You need to configure tunnel interfaces on both the routers. Tunnel source IP address on Router1 will be configured as the tunnel destination IP address on Router2. Tunnel destination IP address on Router1 will be configured as the tunnel source IP address on Router2. In this example, OSPF is used as the routing protocol between the two routers. You can also configure BGP or IS-IS as the routing protocol.

```
RP/0/RP0/CPU0:Router1# configure
RP/0/RP0/CPU0:Router1(config)#  interface tunnel-ip 30
RP/0/RP0/CPU0:Router1(config-if)# tunnel mode gre ipv4
RP/0/RP0/CPU0:Router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
RP/0/RP0/CPU0:Router1(config-if)# tunnel source 192.168.1.1
RP/0/RP0/CPU0:Router1(config-if)# tunnel destination 192.168.2.1
RP/0/RP0/CPU0:Router1(config-if)# exit
RP/0/RP0/CPU0:Router1(config)# interface Loopback 0
RP/0/RP0/CPU0:Router1(config-if)# ipv4 address 10.10.10.1
RP/0/RP0/CPU0:Router1(config-if)# exit
RP/0/RP0/CPU0:Router1(config)# router ospf 1
RP/0/RP0/CPU0:Router1(config-ospf)# router-id 192.168.4.1
RP/0/RP0/CPU0:Router1(config-ospf)# area 0
RP/0/RP0/CPU0:Router1(config-ospf-ar)# interface tunnel-ip 30
RP/0/RP0/CPU0:Router1(config-ospf-ar)#  interface Loopback 0
RP/0/RP0/CPU0:Router1(config-ospf-ar)# commit

RP/0/RP0/CPU0:Router2# configure
RP/0/RP0/CPU0:Router2(config)#  interface tunnel-ip 30
RP/0/RP0/CPU0:Router2(config-if)# tunnel mode gre ipv4
RP/0/RP0/CPU0:Router2(config-if)# ipv4 address 10.1.1.2 255.255.255.0
RP/0/RP0/CPU0:Router2(config-if)# tunnel source 192.168.2.1
RP/0/RP0/CPU0:Router2(config-if)# tunnel destination 192.168.1.1
RP/0/RP0/CPU0:Router2(config-if)# exit
RP/0/RP0/CPU0:Router2(config)# interface Loopback 0
RP/0/RP0/CPU0:Router2(config-if)# ipv4 address 2.2.2.2
RP/0/RP0/CPU0:Router2(config)# router ospf 1
RP/0/RP0/CPU0:Router2(config-ospf)# router-id 192.168.3.1
RP/0/RP0/CPU0:Router2(config-ospf)# area 0
RP/0/RP0/CPU0:Router2(config-ospf-ar)# interface tunnel-ip 30
RP/0/RP0/CPU0:Router2(config-ospf-ar)# interface Loopback 0
RP/0/RP0/CPU0:Router2(config-if)# commit
```

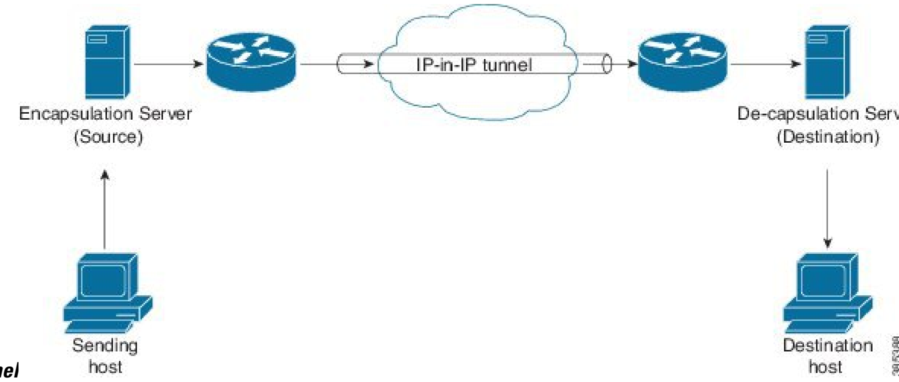
IP-in-IP Decapsulation

Encapsulation of datagrams in a network is done for multiple reasons, such as when a source server wants to influence the route that a packet takes to reach the destination host. The source server is also known as the encapsulation server.

IP-in-IP encapsulation involves the insertion of an outer IP header over the existing IP header. The source and destination address in the outer IP header point to the endpoints of the IP-in-IP tunnel. The stack of IP headers is used to direct the packet over a predetermined path to the destination, provided the network administrator knows the loopback addresses of the routers transporting the packet. This tunneling mechanism can be used for determining availability and latency for most network architectures. It is to be noted that the entire path from source to the destination does not have to be included in the headers, but a segment of the network can be chosen for directing the packets.

The following illustration describes the basic IP-in-IP encapsulation and decapsulation model.

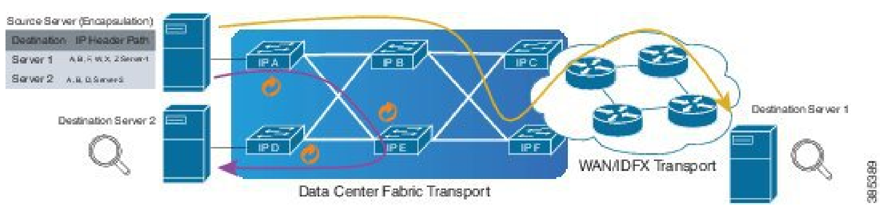
Figure 11: Basic Encapsulation and Decapsulation with an IP-in-IP Tunnel



Use Case: Configure IP-in-IP Decapsulation

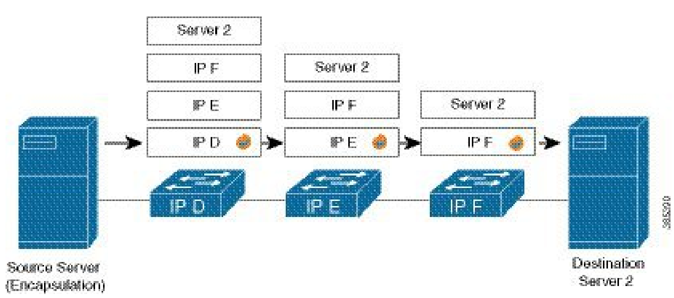
The following topology describes a use case where IP-in-IP encapsulation and decapsulation are used for different segments of the network from source to destination. The IP-in-IP tunnel consists of multiple routers that are used to decapsulate and direct the packet through the data center fabric network.

Figure 12: IP-in-IP Decapsulation Through a Data Center Network



The following illustration shows how the stacked IPv4 headers are de-capsulated as they traverse through the de-capsulating routers.

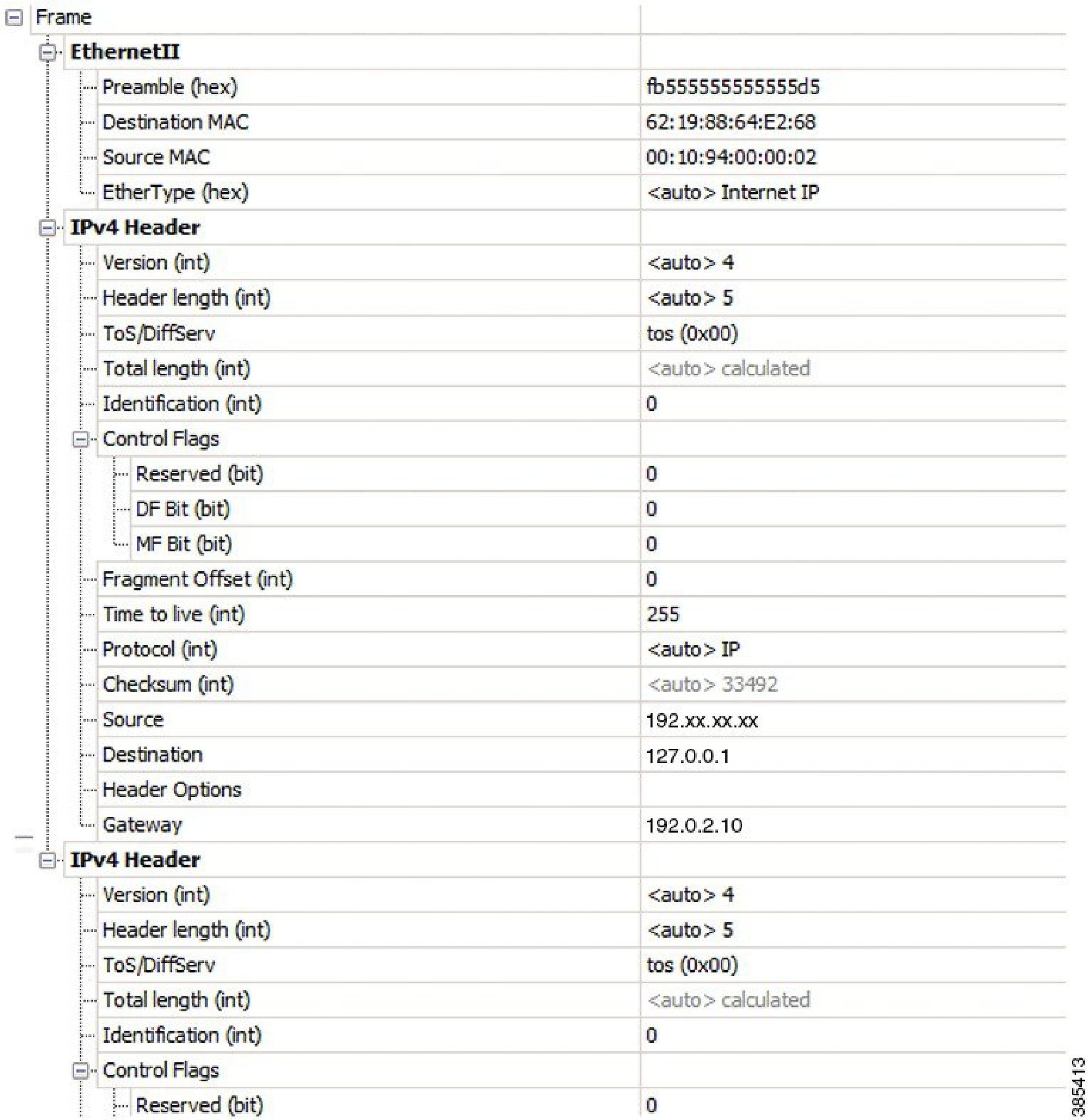
Figure 13: IP Header Decapsulation



Stacked IP Header in an Encapsulated Packet

The encapsulated packet has an outer IPv4 header that is stacked over the original IPv4 header, as shown in the following illustration.

Encapsulated Packet



Configuration

You can use the following sample configuration on the routers to decapsulate the packet as it traverses the IP-in-IP tunnel:

```
RP/0/RP0/CPU0:router(config)# interface tunnel-ip 10
RP/0/RP0/CPU0:router(config-if)# tunnel mode ipv4 decap
RP/0/RP0/CPU0:router(config-if)# tunnel source loopback 0
RP/0/RP0/CPU0:router(config-if)# tunnel destination 10.10.1.2/32
```

- **tunnel-ip**: configures an IP-in-IP tunnel interface.

- **ipv4 unnumbered loopback address**: enables ipv4 packet processing without an explicit address, except for loopback address.
- **tunnel mode ipv4 decap**: enables IP-in-IP decapsulation.
- **tunnel source**: indicates the source address for the IP-in-IP decap tunnel w.r.t the router interface.
- **tunnel destination**: indicates the destination address for the IP-in-IP decap tunnel w.r.t the router interface.

Running Configuration

```
RP/0/RP0/CPU0:router# show running-config interface tunnel-ip 10
...
interface tunnel-ip 10
tunnel mode ipv4 decap
tunnel source Loopback 0
tunnel destination 10.10.1.2/32
```

This completes the configuration of IP-in-IP decapsulation.

Single Pass GRE Encapsulation Allowing Line Rate Encapsulation

Single Pass GRE Encapsulation Allowing Line Rate Encapsulation feature, also known as Prefix-based GRE Tunnel Destination for Load Balancing feature, enables line rate GRE encapsulation traffic and enables flow entropy. Data-plane forwarding performance supports full line rate, which is adjusted to consider added encapsulation. GRE tunnel goes down if the destination is not available in RIB. Routing over GRE Single-pass tunnel is not supported in Release 6.3.2, so the traffic that is eligible for GRE encapsulation is identified using an ACL filter that is based on GRE encapsulation. GRE tunnel destination address is an anycast address. All of the GRE encapsulation must be assigned based upon either an ACL or a policy-map, or both. Destinations may be individual addresses or /28 prefixes.

Configuration

Perform the following tasks to configure the GRE Single-Pass Entropy feature:

- GRE Single-pass
- GRE Entropy(ECMP/UCMP)

```
/* GRE Single-Pass */

Router# configure
Router(config)# interface tunnel-ip30016
Router(config-if)# ipv4 address 216.1.1.1 255.255.255.0
Router(config-if)# ipv6 address 216:1:1::1/64
Router(config-if)# ipv6 enable
Router(config-if)# tunnel mode gre ipv4 encap
Router(config-if)# tunnel source Loopback22
Router(config-if)# tunnel destination 170.170.170.22
Router(config-if)# commit
Router(config-if)# exit
```

```
/* GRE Entropy(ECMP/UCMP)*/

ECMP (ISIS)

Router# configure
Router(config)# router isis core
Router(config)# apply-group ISIS-INTERFACE
Router(config-isis)# is-type level-2-only
Router(config-isis)# net 49.1111.0000.0000.002.00
Router(config-isis)# nsr
Router(config-isis)# log adjacency changes
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# metric 2
Router(config-isis-af)# mpls traffic-eng level-2-only
Router(config-isis-af)# mpls traffic-eng router-id Loopback0
Router(config-isis-af)# maximum-paths 5
Router(config-isis-af)# commit
!

/* UCMP(ISIS) */

Router# configure
Router(config)# router isis core
Router(config)# apply-group ISIS-INTERFACE
Router(config-isis)# is-type level-2-only
Router(config-isis)# net 49.1111.0000.0000.002.00
Router(config-isis)# nsr
Router(config-isis)# log adjacency changes
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# ucmp
Router(config-isis-af)# metric 2
Router(config-isis-af)# mpls traffic-eng level-2-only
Router(config-isis-af)# mpls traffic-eng router-id Loopback0
Router(config-isis-af)# maximum-paths 5
Router(config-isis-af)# redistribute connected
Router(config-isis-af)# commit
Router(config-isis-af)# exit
!

Router# configure
Router(config)# interface Bundle-Ether3
Router(config-if)# apply-group ISIS-INTERFACE
Router(config-if)# address-family ipv4 unicast
Router(config-af)# metric 20
Router(config-af)# commit
Router(config-af)# exit
!

Router# configure
Router(config)# interface Bundle-Ether111
Router(config-if)# apply-group ISIS-INTERFACE
Router(config-if)# address-family ipv4 unicast
Router(config-af)# metric 15
Router(config-af)# commit
Router(config-af)# exit
!

/* ECMP(OSPF) */

Router# configure
Router(config)# router ospf 3
```

```
Router(config-ospf)# nsr
Router(config-ospf)# maximum paths 5
Router(config-ospf)# address-family ipv4 unicast
Router(config-ospf-af)# area 0
Router(config-ospf-af-ar)# interface Bundle-Ether3
Router(config-ospf-af-ar-if)# exit
!
Router(config-ospf-af-ar)# interface Bundle-Ether4
Router(config-ospf-af-ar-if)# exit
!
Router(config-ospf-af-ar)# interface Bundle-Ether111
Router(config-ospf-af-ar-if)# exit
!
Router(config-ospf-af-ar)# interface Bundle-Ether112
Router(config-ospf-af-ar-if)# exit
!
Router(config-ospf-af-ar)# interface Loopback23
Router(config-ospf-af-ar-if)# exit
!
Router(config-ospf-af-ar)# interface HundredGigE0/7/0/23interface HundredGigE0/0/1/0
Router(config-ospf-af-ar-if)# commit
Router(config-ospf-af-ar-if)# exit


/* UCMP(OSPF) */

Router# configure
Router(config)# router ospf 3
Router(config-ospf)# nsr
Router(config-ospf)# maximum paths 5
Router(config-ospf)# ucmp
Router(config-ospf)# address-family ipv4 unicast
Router(config-ospf-af)# area 0
Router(config-ospf-af-ar)# interface Bundle-Ether3 cost 2
Router(config-ospf-af-ar-if)# exit
!
Router(config-ospf-af-ar)# interface Bundle-Ether4
Router(config-ospf-af-ar-if)# exit
!
Router(config-ospf-af-ar)# interface Bundle-Ether111
Router(config-ospf-af-ar-if)# exit
!
Router(config-ospf-af-ar)# interface Bundle-Ether112 cost 2
Router(config-ospf-af-ar-if)# exit
!
Router(config-ospf-af-ar)# interface Loopback23
Router(config-ospf-af-ar-if)# exit
!
Router(config-ospf-af-ar)# interface HundredGigE0/7/0/23interface HundredGigE0/0/1/0
Router(config-ospf-af-ar-if)# commit
Router(config-ospf-af-ar-if)# exit


/* ECMP(BGP) */
Router# configure
Router(config)# router bgp 800
Router(config-bgp)# bgp bestpath as-path multipath-relax
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# network 170.170.170.3/32
Router(config-bgp-af)# network 170.170.170.10/32
Router(config-bgp-af)# network 170.170.170.11/32
Router(config-bgp-af)# network 170.170.172.3/32
Router(config-bgp-af)# network 180.180.180.9/32
```

```
Router(config-bgp-af)# network 180.180.180.20/32
Router(config-bgp-af)# network 180.180.180.21/32
Router(config-bgp-af)# network 180.180.180.24/32
Router(config-bgp-af)# network 180.180.180.25/32
Router(config-bgp-af)# commit
!
Router# configure
Router(config)# router bgp 800
Router(config-bgp)# neighbor 4.1.1.2
Router(config-bgp-nbr)# remote-as 300
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# commit
!

/* UCMP(BGP) */

Router# configure
Router(config)# router bgp 800
Router(config-bgp)# bgp bestpath as-path multipath-relax
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# maximum-paths ebgp 5
Router(config-bgp-af)# network 180.180.180.9/32
Router(config-bgp-af)# network 180.180.180.20/32
Router(config-bgp-af)# network 180.180.180.21/32
Router(config-bgp-af)# network 180.180.180.24/32
Router(config-bgp-af)# network 180.180.180.25/32
Router(config-bgp-af)# commit
!
Router# configure
Router(config)# router bgp 800
Router(config-bgp)# neighbor 7.1.5.2
Router(config-bgp-nbr)# remote-as 4000
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy TRANSIT0_IN in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# commit
!
Router# configure
Router(config)# router bgp 800
Router(config-bgp)# 4.1.111.2
Router(config-bgp-nbr)# remote-as 4000
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy TRANSIT0_IN in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# commit
!

/* Configure roupte policy */

Router# configure
Router(config)# route-policy TRANSIT0_IN
Router(config-rpl)# if destination in (170.170.170.24/32) then
Router(config-rpl-if)# set extcommunity bandwidth (2906:1250000)
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl-else)# endif
```

```
Router(config-rpl)# end-policy
!

Router# configure
Router(config)# route-policy TRANSIT1_IN
Router(config-rpl)# if destination in (170.170.170.24/32) then
Router(config-rpl-if)# set extcommunity bandwidth (2906:37500000
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
```

Running Configuration

```
/* GRE Single-Pass configuration */

interface tunnel-ip30016
ipv4 address 216.1.1.1 255.255.255.0
ipv6 address 216:1:1::1/64
ipv6 enable
tunnel mode gre ipv4 encap
tunnel source Loopback22
tunnel destination 170.170.170.22
!

/* GRE Entropy(ECMP/UCMP) */

ECMP (ISIS)

router isis core
apply-group ISIS-INTERFACE
is-type level-2-only
net 49.1111.0000.0000.002.00
nsr
log adjacency changes
address-family ipv4 unicast
metric-style wide
metric 2
mpls traffic-eng level-2-only
mpls traffic-eng router-id Loopback0
maximum-paths 5
!

/* UCMP(ISIS) */

router isis core
apply-group ISIS-INTERFACE
is-type level-2-only
net 49.1111.0000.0000.002.00
nsr
log adjacency changes
address-family ipv4 unicast
metric-style wide
ucmp
metric 2
mpls traffic-eng level-2-only
mpls traffic-eng router-id Loopback0
maximum-paths 5
redistribute connected
!
```

```
interface Bundle-Ether3
apply-group ISIS-INTERFACE
address-family ipv4 unicast
metric 20
!

interface Bundle-Ether111
apply-group ISIS-INTERFACE
address-family ipv4 unicast
metric 15
!

!

/* ECMP(OSPF) */

router ospf 3
nsr
maximum paths 5
address-family ipv4 unicast
area 0
interface Bundle-Ether3
!
interface Bundle-Ether4
!
interface Bundle-Ether111
!
interface Bundle-Ether112
!
interface Loopback23
!
interface HundredGigE0/7/0/23HundredGigE0/0/1/0
!
!
!
/* UCMP (OSPF) */

router ospf 3
nsr
maximum paths 5
ucmp
address-family ipv4 unicast
area 0
interface Bundle-Ether3
cost 2
!
interface Bundle-Ether4
!
interface Bundle-Ether111
!
interface Bundle-Ether112
cost 2
!
interface Loopback23
!
interface HundredGigE0/7/0/23HundredGigE0/0/1/0
!
!
!

/* ECMP(BGP)*/

router bgp 800
bgp bestpath as-path multipath-relax
```

```
address-family ipv4 unicast
maximum-paths ebgp 5
network 170.170.170.3/32
network 170.170.170.10/32
network 170.170.170.11/32
network 170.170.172.3/32
network 180.180.180.9/32
network 180.180.180.20/32
network 180.180.180.21/32
network 180.180.180.24/32
network 180.180.180.25/32
!
neighbor 4.1.1.2
remote-as 300
address-family ipv4 unicast
route-policy PASS-ALL in
route-policy PASS-ALL out
next-hop-self
!
!

/* UCMP(BGP) */

router bgp 800
bgp bestpath as-path multipath-relax
address-family ipv4 unicast
maximum-paths ebgp 5
network 180.180.180.9/32
network 180.180.180.20/32
network 180.180.180.21/32
network 180.180.180.24/32
network 180.180.180.25/32
!

neighbor 7.1.5.2
remote-as 4000
address-family ipv4 unicast
route-policy TRANSIT0_IN in
route-policy PASS-ALL out
next-hop-self
!
!
neighbor 4.1.111.2
remote-as 4000
address-family ipv4 unicast
route-policy TRANSIT1_IN in
route-policy PASS-ALL out
next-hop-self
!
!

/* Configure roupte policy */

route-policy TRANSIT0_IN
if destination in (170.170.170.24/32) then
set extcommunity bandwidth (2906:1250000)
else
pass
endif
end-policy
!
route-policy TRANSIT1_IN
if destination in (170.170.170.24/32) then
set extcommunity bandwidth (2906:37500000)
```

```
else
pass
endif
end-policy
!
```

Verification

Verify if the tunnel mode GRE encapsulation is enabled.

```
Router# show int tunnel-ip2

interface tunnel-ip2
 ipv4 address 80.80.82.1 255.255.255.0
 ipv6 address 2000:80:80:82::1/64
 load-interval 30
 tunnel mode gre ipv4 encap
 tunnel source Loopback4
 tunnel destination 11.4.2.2
!

RP/0/RP0/CPU0:PE1_5516#show int tunnel-ip2
tunnel-ip2 is up, line protocol is up
  Interface state transitions: 1
  Hardware is Tunnel
  Internet address is 80.80.82.1/24
  MTU 1500 bytes, BW 100 Kbit (Max: 100 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation TUNNEL_IP,  loopback not set,
  Last link flapped 1d18h
  Tunnel TOS 0
  Tunnel mode GRE IPV4, encap
  Keepalive is disabled.
  Tunnel source 11.11.12.1 (Loopback4), destination 11.4.2.2/32
  Tunnel TTL 255
  Last input never, output never
  Last clearing of "show interface" counters 14:53:37
  30 second input rate 0 bits/sec, 0 packets/sec
  30 second output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
     0 packets output, 0 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
```

Verify if the recycle of the packets are not done under Recycle VoQ: 48:

```
 Router# show tunnel ip ea summary location 0/7/CPU0

Number of tunnel updates to retry: 0
Number of tunnel updates retried: 0
Number of tunnel retries failed: 0
Platform:
Recycle VoQ: 48
          ReceivedBytes   ReceivedPackets    ReceivedKbps
          DroppedBytes    DroppedPackets     DroppedKbps

NPU 0:0      0                0                    0
             0                0                    0
1            0                0                    0
             0                0                    0
2            0                0                    0
             0                0                    0
```

```
3           0               0                    0
            0               0                    0
...
NPU 1:0     0               0                    0
            0               0                    0
1           0               0                    0
            0               0                    0
2           0               0                    0
            0               0                    0
3           0               0                    0
            0               0                    0

NPU 2:0     0               0                    0
            0               0                    0
1           0               0                    0
            0               0                    0
2           0               0                    0
            0               0                    0
3           0               0                    0
            0               0                    0
```

Verify if the tunnel mode GRE encapsulation is enabled.

```
Router# show interfaces tunnel-ip * brief

Thu Sep 7 00:04:39.125 PDT
Intf Intf LineP Encap MTU BW
Name     State    State   Type       (byte)  (Kbps)
--------------------------------------------------------------------------------
ti30001  down     down    TUNNEL_IP   1500   100
ti30002  up       up      TUNNEL_IP   1500   100
```

Verify the tunnel endpoint route in RIB.

```
Router# show route 10.1.1.1

Routing entry for 10.0.0.0/8
Known via "static", distance 1, metric 0 (connected)
Installed Oct 2 15:50:56.755 for 00:39:24
Routing Descriptor Blocks
directly connected, via tunnel-ip109
Route metric is 0, Wt is 1
No advertising protos.
```

Verify if the tunnel mode GRE encapsulation is enabled.

```
Router# show tunnel ip ea database tunnel-ip 109 location 0/7/CPU0

----- node0_0_CPU0 -----
tunnel ifhandle 0x80022cc
tunnel source 161.115.1.2
tunnel destination 162.1.1.1/32
tunnel transport vrf table id 0xe0000000
tunnel mode gre ipv4, encap
tunnel bandwidth 100 kbps
tunnel platform id 0x0
tunnel flags 0x40003400
IntfStateUp
BcStateUp
Ipv4Caps
Encap
tunnel mtu 1500
tunnel tos 0
tunnel ttl 255
tunnel adjacency flags 0x1
tunnel o/p interface handle 0x0
```

```
tunnel key 0x0, entropy length 0 (mask 0xffffffff)
tunnel QT next 0x0
tunnel platform data (nil)
Platform:
Handle: (nil)
Decap ID: 0
Decap RIF: 0
Decap Recycle Encap ID: 0x00000000
Encap RIF: 0
Encap Recycle Encap ID: 0x00000000
Encap IPv4 Encap ID: 0x4001381b
Encap IPv6 Encap ID: 0x00000000
Encap MPLS Encap ID: 0x00000000
DecFEC DecRcyLIF DecStatsId EncRcyLIF
```

Verify if the QoS table is updated properly.

```
Router# show controllers npu stats voq base 48 instance all location
0/0/CPU0
Asic Instance = 0
VOQ Base = 48
       ReceivedPkts    ReceivedBytes    DroppedPkts    DroppedBytes
-------------------------------------------------------------------
COS0 =   0              0                0              0
COS1 =   0              0                0              0
COS2 =   0              0                0              0
COS3 =   0              0                0              0

Asic Instance = 1
VOQ Base = 48
       ReceivedPkts    ReceivedBytes    DroppedPkts    DroppedBytes
-------------------------------------------------------------------
COS0 =   0              0                0              0
COS1 =   0              0                0              0
COS2 =   0              0                0              0
COS3 =   0              0                0              0

Asic Instance = 2
VOQ Base = 48
       ReceivedPkts    ReceivedBytes    DroppedPkts    DroppedBytes
-------------------------------------------------------------------
COS0 =   0              0                0              0
COS1 =   0              0                0              0
COS2 =   0              0                0              0
COS3 =   0              0                0              0
```