Interface and Hardware Component Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.0.x

First Published: 2020-03-01

Last Modified: 2020-07-31

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 4

Preface

Note This product has reached end-of-life status. For more information, see the End-of-Life and End-of-Sale Notices.

This guide describes the Netflow configuration details for Cisco IOS XR software. This chapter contains details on the changes made to this document.

Changes to This Document

This table lists the technical changes made to this document since it was first released.

Table 1: Changes to This Document

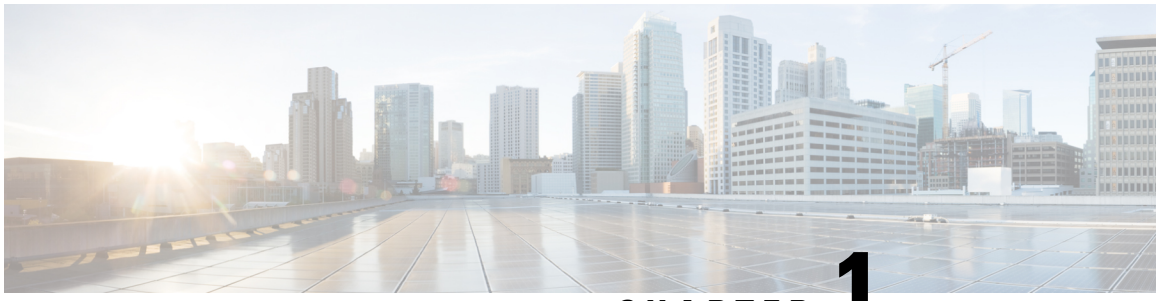
Date	Summary
August 2020	Republished for 7.0.14 release.
March 2020	Initial release of this document.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.



CHAPTER 1

New and Changed Feature Information

This table summarizes the new and changed feature information for the *Interfaces Configuration Guide for Cisco 8000 Series Routers* for Cisco 8000 Series Routers, and tells you where they are documented.

Interface and Hardware Component Features Added or Modified in IOS XR Release 7.0.x

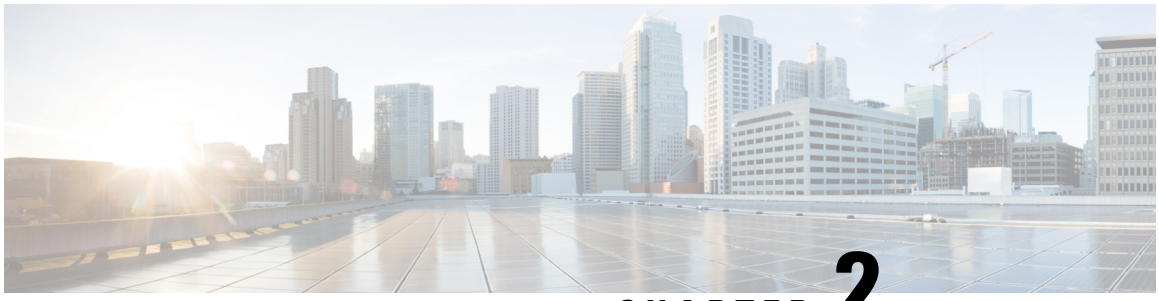
This table summarizes the new and changed feature information for the *Interfaces Configuration Guide for Cisco 8000 Series Routers* for Cisco 8000 Series Routers, and tells you where they are documented.

Table 2: New and Changed Features

Feature	Description	Introduced in Release	Where Documented
IP Event Dampening	The IP Event Dampening feature introduces a configurable exponential decay mechanism to suppress the effects of excessive interface flapping events on routing protocols and routing tables in the network. This feature allows the network operator to configure a router to automatically identify and selectively dampen a local interface that is flapping.	Release 7.0.12	For more information about the feature, see the chapter *IP Event Dampening*.
Extended ACL needs to match on the outer header for IP-in-IP De-capsulation	Extended ACL needs to match on the outer header for IP-in-IP De-capsulation.	Release 7.0.14	For more information about the feature, see the chapter *Configure Tunnels*.

Feature	Description	Introduced in Release	Where Documented
Configuration of IP DSCP in ERSPAN	Configuration of IP DSCP.	Release 7.0.14	For more information about the feature, see the section *Flexible CLI for ERSPAN* in chapter *Configuring Traffic Mirroring*.
Tunnel IP for ERSPAN	Tunnel IP support for ERSPAN	Release 7.0.14	For more information about the feature, see the section *Flexible CLI for ERSPAN* in chapter *Configuring Traffic Mirroring*.
Ability to source ranges of interfaces and SVIs in ERSPAN	Ability to source ranges of interfaces and SVIs in ERSPAN	Release 7.0.14	For more information about the feature, see the section *Flexible CLI for ERSPAN* in chapter *Configuring Traffic Mirroring*.
Sequence bit is set in the GRE header and the value of sequence number is always 0 for ERSPAN packets.	Sequence bit is set in the GRE header and the value of sequence number is always 0 for ERSPAN packets.	Release 7.0.14	For more information about the feature, see the section *ERSPAN* in chapter *Configuring Traffic Mirroring*.
ERSPAN and Security ACL should be separate.	ERSPAN and Security ACL should be separate.	Release 7.0.14	For more information about the feature, see the section *Configuring ACLs for Traffic Mirroring* in chapter *Configuring Traffic Mirroring*.
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.	Release 7.0.14	For more information about the feature, see the section *Introduction to File Mirroring* in chapter *Configure Traffic Mirroring*.

Feature	Description	Introduced in Release	Where Documented
Controlling the TTL Value of Inner Payload Header	This feature enables user to control the TTL value of the inner packet's header that is forwarded after de-capsulating the outer IP-in-IP header.	Release 7.0.14	For more information about the feature, see the section Controlling the TTL Value of Inner Payload Header, on page 107 in chapter *Configure Tunnels*.



CHAPTER 2

Preconfiguring Physical Interfaces

This module describes the preconfiguration of physical interfaces.

The system supports preconfiguration for the following interfaces:

- 10-Gigabit Ethernet
- 40-Gigabit Ethernet
- 100-Gigabit Ethernet
- 400-Gigabit Ethernet
- Management Ethernet

Preconfiguration allows you to configure line cards before you insert them into the router. When you insert the cards, they are instantly configured. The system creates the preconfiguration information in a different system database tree, rather than with the regularly configured interfaces. That database tree is known as the *preconfiguration directory* on the Route Processor.

There might be some preconfiguration data that you cannot verify unless the line card is present. This is because the verifiers themselves run only on the line card. You can verify such preconfiguration data when you insert the line card and initiate the verifiers. The system rejects a configuration if errors are found when you copy the configuration from the preconfiguration area to the active area.

Note Gigabit Ethernet interface is not supported. You can only preconfigure physical interfaces.

Prerequisites for Preconfiguring Physical Interfaces

Before preconfiguring physical interfaces, ensure that you meet the following condition(s):

- Preconfiguration drivers and files are installed. Although it might 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.

Information About Preconfiguring Physical Interfaces

To preconfigure interfaces, you must understand the following concepts:

Physical Interface Preconfiguration Overview

Preconfiguration is the process of configuring interfaces before they are present in the system. You cannot verify or apply preconfigured interfaces until you insert the actual interface into the router with the matching location. The location can be the rack, slot, or module. When you insert the anticipated line card and create the interface, the system verifies the precreated configuration information. If the verification is successful, the system immediately applies the running configuration of the router.

Note When you plug the anticipated line card in, ensure that you verify any preconfiguration by using the appropriate **show** commands.

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

Note We recommend filling out preconfiguration information in your site planning guide. This allows you to compare the anticipated configuration with the actual preconfigured interfaces when you install the card and the interfaces are up.

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 the valid configuration (best effort). Some configuration might fail due to semantic errors, but the valid configuration still comes up.

Benefits of Interface Preconfiguration

Preconfigurations reduce downtime when you add new cards to the system. With preconfiguration, you can instantly configure the new modular services card that actively runs during the line card bootup.

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

Use of the Interface Preconfigure Command

To preconfigure the interfaces that are not yet present in the system, use the **interface preconfigure** command in global configuration mode.

The **interface preconfigure** command places the router in interface configuration mode. You must be able to add any possible interface commands. The verifiers registered for the preconfigured interfaces verify the

configuration. The preconfiguration is complete when you enter the **end** command, or any matching exit or global configuration mode command.

Note It is possible that you are not able to verify some configurations until you insert 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.

You must provide names during preconfiguration that matches with the name of the interface that is created. If the interface names do not match, the system does not apply preconfiguration 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.

Active and Standby RPs and Virtual Interface Configuration

The standby RP is available and is in a state in which it can take the load from the an active RP, if required. Following are the conditions when a standby RP becomes an active RP:

- Failure detection by a watchdog.
- Standby RP is administratively commanded to take over.
- Removal of the active RP from the chassis.

If a second RP is not present in the chassis while the first is in operation, the system may insert a second RP. The second RP then automatically becomes the standby RP. The standby RP may also be removed from the chassis with no effect on the system other than loss of RP redundancy.

After failover, the virtual interfaces become available on the standby (now active) RP. Their state and configuration is unchanged, and there is no loss of forwarding (in the case of tunnels) over the interfaces during the failover. The routers use nonstop forwarding (NSF) over tunnels through the failover of the host RP.

Note You do not need to configure anything to guarantee that the standby interface configurations are maintained.

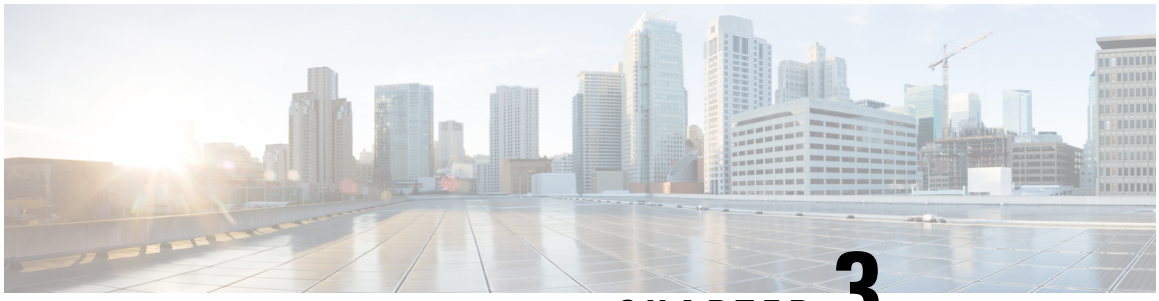
How to Preconfigure Physical Interfaces

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

```
/* Enter global configuration mode. */
 RP/0/RP0/CPU0:router:router:hostname# configure
```

```
/* 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. */

RP/0/RP0/CPU0:router:router(config)# interface preconfigure HundredGigE 0/3/0/2

/* Assign an IP address and mask to the interface. Use one of the following commands:
- ipv4 address ip-address subnet-mask
- ipv4 address ip-address/prefix */
RP/0/RP0/CPU0:router(config-if-pre)# ipv4 address 192.168.1.2/31
RP/0/RP0/CPU0:router(config-if-pre)# end
or
RP/0/RP0/CPU0:router(config-if-pre)# commit
RP/0/RP0/CPU0:router# show running-config
```

- 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.



CHAPTER 3

Advanced Configuration and Modification of the Management Ethernet Interface

This module describes the configuration of Management Ethernet interfaces.

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

Note By default, the Management Ethernet interfaces are present on the system. However, you must configure these interfaces to:

- Access the router.
- Use protocols and applications, such as Simple Network Management Protocol (SNMP), HTTP, eXtensible Markup Language (XML), TFTP, Telnet, and Command-Line Interface (CLI.)

Prerequisites for Configuring Management Ethernet Interfaces

Before you perform the Management Ethernet interface configuration procedures that are described in this chapter, ensure that you meet the following tasks and conditions:

- 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, ensure that both the active and standby Management Ethernet interfaces are physically connected to the same LAN or switch.

Information About Configuring Management Ethernet Interfaces

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

Default Interface Settings

This table describes the default Management Ethernet interface settings that you can change with manual configuration. The system does not display the default settings in the **show running-config** command output.

Table 3: Management Ethernet Interface Default Settings

Parameter	Default Value	Configuration File Entry
Speed in Mbps	Default speed is **1G** with autonegotiated.	Speed in non-configurable.
Duplex mode	Default duplex mode is **full-duplex** with autonegotiated.	Duplex mode is non-configurable.
MAC address	MAC address is read from the hardware burned-in address (BIA).	MAC address is non-configurable.

How to Perform Advanced Management Ethernet Interface Configuration

This section contains the following procedures:

Configure a Management Ethernet Interface

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

Note The maximum MTU value for the management interface MgmtEth0/RP0/CPU0/0 is 9678 bytes.

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

/* Enter interface configuration mode and specify the Ethernet interface name and notation
 rack/slot/module/port. */
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 (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. You can specify the network mask in either of the 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 system indicates the network mask 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 the network address.

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

Note (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 from 64 through 1514 bytes.

```
/* Remove the shutdown configuration, which removes the forced administrative down on the
interface, enabling it to move to an up or down state.  */
RP/0/RP0/CPU0:router(config-if)# no shutdown
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.

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

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)# ipv4 address 1.76.18.150/16
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 the 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
!
```

Verify Management Ethernet Interface Configuration

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

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

Configuration Examples for Management Ethernet Interfaces

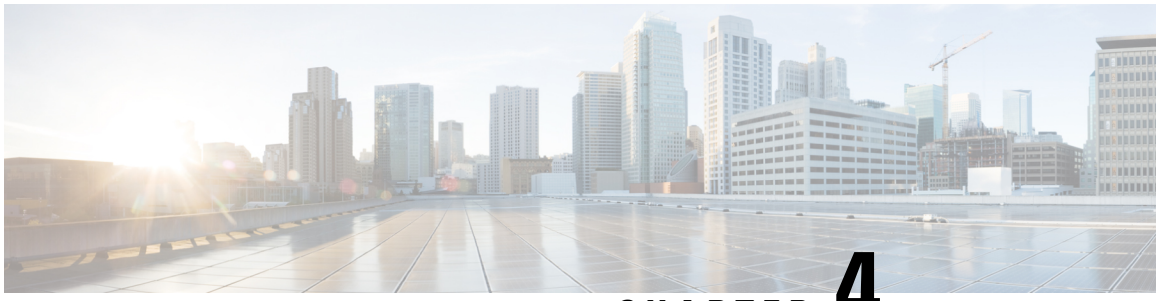
This section provides the following configuration examples:

Configuring a Management Ethernet Interface: Example

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

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# interface MgmtEth 0/RP0/CPU0/0
RP/0//CPU0:router(config)# ipv4 address 172.29.52.70 255.255.255.0
RP/0//CPU0:router(config-if)# no shutdown
RP/0//CPU0:router(config-if)# commit
RP/0//CPU0:Mar 26 01:09:28.685 :ifmgr[190]:%LINK-3-UPDOWN :Interface MgmtEth 0/RP0/CPU0/0,
 changed state to Up
RP/0//CPU0:router(config-if)# end

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

MMgmtEth0//CPU0/0 is up, line protocol is up
  Hardware is Management Ethernet, address is 0011.93ef.e8ea (bia 0011.93ef.e8ea
)
  Description: Connected to Lab LAN
  Internet address is 172.29.52.70/24
  MTU 1514 bytes, BW 100000 Kbit
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA,  loopback not set,
  ARP type ARPA, ARP timeout 04:00:00
  Last clearing of "show interface" counters never
  5 minute input rate 3000 bits/sec, 7 packets/sec
  5 minute output rate 0 bits/sec, 1 packets/sec
     30445 packets input, 1839328 bytes, 64 total input drops
     0 drops for unrecognized upper-level protocol
     Received 23564 broadcast packets, 0 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     171672 packets output, 8029024 bytes, 0 total output drops
     Output 16 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     1 carrier transitions

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

interface MgmtEth0/RP0/CPU0/0
 description Connected to Lab LAN
```

```
 ipv4 address 172.29.52.70 255.255.255.0
!
```



CHAPTER 4

Configuring Ethernet Interfaces

This module describes the configuration of Ethernet interfaces.

The distributed 10-Gigabit, 40-Gigabit, 100-Gigabit Ethernet, 400-Gigabit Ethernet architecture and features deliver network scalability and performance, while enabling service providers to offer high-density, high-bandwidth networking solutions designed to interconnect the router with other systems in POPs, including core and edge routers, and Layer 3 switches.

Tip You can programmatically configure and manage the Ethernet interfaces using `openconfig-if-ethernet.yang` and `openconfig-interfaces.yang` OpenConfig data models. To get started with using data models, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

Prerequisites for Configuring Ethernet Interfaces

Before configuring Ethernet interfaces, ensure that you meet the following conditions:

- Access to Cisco 8200 series routers or Cisco 8800 series routers with at least one of the supported line cards installed.
- Know the interface IP address.
- Ensure to specify the generalized interface name with the standard notation of *rack/slot/module/port*.

Information About Configuring Ethernet

This section provides the following information:

Cisco 8000 Modular Line Cards

The current release of the Cisco 8800 Series Routers support the following line cards:

- 36-port QSFP56-DD 400 GbE Line Card - This line card provides 14.4 Tbps via 36 QSFP56-DD ports. It also supports 100G, 2x100G, and 400G modules. If 36 of 2x100G modules are used, the line card can have 72 HundredGigE interfaces.
- 48-port QSFP28 100 GbE Line Card - This line card provides 4.8 Tbps with MACsec support on all ports. It also supports QSFP+ optics for 40G compatibility.

The 8800 Series line cards utilize multiple #ChipName forwarding ASICs to achieve high performance and bandwidth with line rate forwarding.

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 36-port Line Card or a 48-port 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 4: 100-Gigabit Ethernet Line Card Default Configuration Values

Parameter	Configuration File Entry	Default Value
Flow control	**flow-control**	egress off ingress off
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)

Gigabit Ethernet Protocol Standards Overview

The Gigabit Ethernet interfaces support the following protocol standards:

These standards are further described in the sections that follow.

IEEE 802.3 Physical Ethernet Infrastructure

The IEEE 802.3 protocol standards define the physical layer and MAC sublayer of the data link layer of wired Ethernet. IEEE 802.3 uses Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access at various speeds over various physical media. The IEEE 802.3 standard covers 10 Mbps Ethernet. Extensions to the IEEE 802.3 standard specify implementations for 40-Gigabit Ethernet and 100-Gigabit Ethernet.

IEEE 802.3ae 10-Gbps Ethernet

Under the International Standards Organization's Open Systems Interconnection (OSI) model, Ethernet is fundamentally a Layer 2 protocol. 10-Gigabit Ethernet uses the IEEE 802.3 Ethernet MAC protocol, the IEEE 802.3 Ethernet frame format, and the minimum and maximum IEEE 802.3 frame size. 10-Gbps Ethernet conforms to the IEEE 802.3ae protocol standards.

Just as 1000BASE-X and 1000BASE-T (Gigabit Ethernet) remained true to the Ethernet model, 10-Gigabit Ethernet continues the natural evolution of Ethernet in speed and distance. Because it is a full-duplex only and fiber-only technology, it does not need the carrier-sensing multiple-access with the CSMA/CD protocol that defines slower, half-duplex Ethernet technologies. In every other respect, 10-Gigabit Ethernet remains true to the original Ethernet model.

IEEE 802.3ba 100 Gbps Ethernet

IEEE 802.3ba is supported on the Cisco 1-Port 100-Gigabit Ethernet PLIM beginning in Cisco IOS XR 7.0.11.

MAC Address

A MAC address is a unique 6-byte address that identifies the interface at Layer 2.

Ethernet MTU

The Ethernet maximum transmission unit (MTU) is the size of the largest frame, minus the 4-byte frame check sequence (FCS), that the system transmits 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 unfragmented packet that the system can send. 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, the system fragments that packet as necessary. This process ensures that the originating device does not send an IP packet that is too large.

The system automatically enables the jumbo frame support 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).

Flow Control on Ethernet Interfaces

The flow control that the system uses on 10-Gigabit Ethernet interfaces consists of periodically sending flow control pause frames. It is fundamentally different from the usual full and half-duplex flow control that is used on standard management interfaces. You can activate or deactivate flow control for ingress traffic only. The system automatically implements flow control for egress traffic.

802.1Q VLAN

A VLAN is a group of devices on one or more LANs that the system configures so that they can communicate as if they are attached to the same wire, when in fact they are located on several different LAN segments. Because VLANs are based on logical instead of physical connections, it is flexible for user and host management, bandwidth allocation, and resource optimization.

The IEEE's 802.1Q protocol standard addresses the problem of breaking large networks into smaller parts so broadcast and multicast traffic does not consume more bandwidth than necessary. The standard also helps to provide a higher level of security between segments of internal networks.

The 802.1Q specification establishes a standard method for inserting VLAN membership information into Ethernet frames.

Subinterfaces on the Router

In Cisco IOS XR, interfaces are, by default, main interfaces. A main interface is also known as a trunk interface, which you must not confuse with the word trunk in the context of VLAN trunking.

There are two types of trunk interfaces:

- Physical
- Bundle

On the router, the system automatically creates the physical interfaces when the router recognizes a card and its physical interfaces. However, the system does not automatically create bundle interfaces but you must create them at the time of cofiguration.

The following configuration samples are examples of the trunk interfaces that you can create:

- interface HundredGigE 0/5/0/0
- interface bundle-ether 1

A subinterface is a logical interface that the system create under a trunk interface.

To create a subinterface, you must first identify a trunk interface under which to place it. In case of bundle interfaces, if a trunk interface does not exist, you must create a bundle interface before creating any subinterfaces under it.

You can then assign a subinterface number to the subinterface that you want to create. The subinterface number must be a positive integer from zero to some high value. For a given trunk interface, each subinterface under it must have a unique value.

Subinterface numbers do not need to be contiguous or in numeric order. For example, the following subinterfaces numbers are valid under one trunk interface:

1001, 0, 97, 96, 100000

Subinterfaces can never have the same subinterface number under one trunk.

In the following example, the card in slot 5 has trunk interface, HundredGigE 0/5/0/0. A subinterface, HundredGigE 0/5/0/0.0, is created under it.

```
RP/0/RSP0/CPU0:router#  conf
Mon Sep 21 11:12:11.722 EDT
RP/0/RSP0/CPU0:router(config)#  interface HundredGigE0/5/0/0.0
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RSP0/CPU0:router(config-subif)# commit

RP/0/RSP0/CPU0:Sep 21 11:12:34.819 : config[65794]: %MGBL-CONFIG-6-DB_COMMIT : Configuration
 committed by user 'root'. Use 'show configuration commit changes 1000000152' to view the
changes.

RP/0/RSP0/CPU0:router(config-subif)# end

RP/0/RSP0/CPU0:Sep 21 11:12:35.633 : config[65794]: %MGBL-SYS-5-CONFIG_I : Configured from
 console by root
RP/0/RSP0/CPU0:router#
```

The **show run** command displays the trunk interface first, then the subinterfaces in ascending numerical order.

```
RP/0/RSP0/CPU0:router# show run | begin HundredGigE 0/5/0/0
Mon Sep 21 11:15:42.654 EDT
Building configuration...
interface HundredGigE 0/5/0/0
 shutdown
!
interface HundredGigE 0/5/0/0.0
 encapsulation dot1q 100
!
interface HundredGigE 0/5/0/1
 shutdown
!
```

When a subinterface is first created, the router recognizes it as an interface that, with few exceptions, is interchangeable with a trunk interface. After the new subinterface is configured further, the **show interface** command can display it along with its unique counters:

The following example shows the display output for the trunk interface, HundredGigE 0/5/0/0, followed by the display output for the subinterface HundredGigE 0/5/0/0.0.

```
RP/0/RSP0/CPU0:router# show interface HundredGigE 0/5/0/0
Mon Sep 21 11:12:51.068 EDT
HundredGigE0/5/0/0 is administratively down, line protocol is administratively down.
  Interface state transitions: 0
  Hardware is HundredGigE, address is 0024.f71b.0ca8 (bia 0024.f71b.0ca8)
  Internet address is Unknown
  MTU 1514 bytes, BW 1000000 Kbit
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation 802.1Q Virtual LAN,
```

```
  Full-duplex, 1000Mb/s, SXFD, link type is force-up
  output flow control is off, input flow control is off
  loopback not set,
  ARP type ARPA, ARP timeout 04:00:00
  Last input never, output never
  Last clearing of "show interface" counters never
  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 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions


RP/0/RSP0/CPU0:router# show interface HundredGigE0/5/0/0.0
Mon Sep 21 11:12:55.657 EDT
HundredGigE0/5/0/0.0 is administratively down, line protocol is administratively down.
  Interface state transitions: 0
  Hardware is VLAN sub-interface(s), address is 0024.f71b.0ca8
  Internet address is Unknown
  MTU 1518 bytes, BW 1000000 Kbit
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation 802.1Q Virtual LAN, VLAN Id 100,  loopback not set,
  ARP type ARPA, ARP timeout 04:00:00
  Last input never, output never
  Last clearing of "show interface" counters never
  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
```

This example shows two interfaces being created at the same time: first, the bundle trunk interface, then a subinterface attached to the trunk:

```
RP/0/RSP0/CPU0:router# conf
Mon Sep 21 10:57:31.736 EDT
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether1
RP/0/RSP0/CPU0:router(config-if)# no shut
RP/0/RSP0/CPU0:router(config-if)# interface bundle-Ether1.0
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RSP0/CPU0:router(config-subif)# commit
RP/0/RSP0/CPU0:Sep 21 10:58:15.305 : config[65794]: %MGBL-CONFIG-6-DB_COMMIT : C
onfiguration committed by user 'root'. Use 'show configuration commit changes 10
00000149' to view the changes.
RP/0/RSP0/CPU0:router# show run | begin Bundle-Ether1
Mon Sep 21 10:59:31.317 EDT
Building configuration..
interface Bundle-Ether1
!
interface Bundle-Ether1.0
 encapsulation dot1q 100
!
```

You delete a subinterface using the **no interface** command.

```
RP/0/RSP0/CPU0:router#
RP/0/RSP0/CPU0:router# show run | begin HundredGigE 0/5/0/0
Mon Sep 21 11:42:27.100 EDT
Building configuration...
interface HundredGigE 0/5/0/0
 negotiation auto
!
interface HundredGigE 0/5/0/0.0
 encapsulation dot1q 100
!
interface HundredGigE 0/5/0/1
 shutdown
!
RP/0/RSP0/CPU0:router# conf
Mon Sep 21 11:42:32.374 EDT
RP/0/RSP0/CPU0:router(config)# no interface HundredGigE 0/5/0/0.0
RP/0/RSP0/CPU0:router(config)# commit
RP/0/RSP0/CPU0:Sep 21 11:42:47.237 : config[65794]: %MGBL-CONFIG-6-DB_COMMIT : Configuration
 committed by user 'root'. Use 'show configuration commit changes 1000000159' to view the
changes.
RP/0/RSP0/CPU0:router(config)# end
RP/0/RSP0/CPU0:Sep 21 11:42:50.278 : config[65794]: %MGBL-SYS-5-CONFIG_I : Configured from
 console by root
RP/0/RSP0/CPU0:router# show run | begin HundredGigE 0/5/0/0
Mon Sep 21 11:42:57.262 EDT
Building configuration...
interface HundredGigE 0/5/0/0
 negotiation auto
!
interface HundredGigE 0/5/0/1
 shutdown
!
```

Other Performance Management Enhancements

The following additional performance management enhancements are included in Cisco IOS XR Release 7.0.11:

- You can retain performance management history statistics across a process restart or route processor (RP) failover using the new **history-persistent** keyword option for the **performance-mgmt statistics interface** command.
- You can save performance management statistics to a local file using the **performance-mgmt resources dump local** command.
- You can filter performance management instances by defining a regular expression group (**performance-mgmt regular-expression** command), which includes multiple regular expression indices that specify strings to match. You apply a defined regular expression group to one or more statistics or threshold templates in the **performance-mgmt statistics interface** or **performance-mgmt thresholds** interface commands.

Frequency Synchronization and SyncE

Cisco IOS XR Software supports SyncE-capable Ethernet on the router. Frequency Synchronization enables you to distribute the precision clock signals around the network. The system injects a highly accurate timing signals into the router in the network. The timing signals use an external timing technology, such as Cesium atomic clocks, or GPS, and then pass the signals to the physical interfaces of the router. Peer routers can then

recover this precision frequency from the line, and also transfer it around the network. This feature is traditionally applicable to SONET or SDH networks, but is now available on Ethernet for Cisco 8000 Series Router with Synchronous Ethernet capability. For more information, see *Cisco 8000 Series Router System Management Configuration Guide*.

LLDP

The Cisco Discovery Protocol (CDP) is a device discovery protocol that runs over Layer 2 (the Data Link layer) on all Cisco-manufactured devices (routers, bridges, access servers, and switches). CDP allows network management applications to automatically discover and learn about other Cisco devices connected to the network.

To support non-Cisco devices and to allow for interoperability between other devices, the router also supports the IEEE 802.1AB Link Layer Discovery Protocol (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.

LLDP supports a set of attributes that it uses to learn information about neighbor devices. These attributes have a defined format known as a Type-Length-Value (TLV). LLDP supported devices can use TLVs to receive and send information to their neighbors. Details such as configuration information, device capabilities, and device identity can be advertised using this protocol.

In addition to the mandatory TLVs (Chassis ID, Port ID, End of LLDPDU, and Time-to-Live), the router also supports the following basic management TLVs, which are optional:

- Port Description
- System Name
- System Description
- System Capabilities
- Management Address

These optional TLVs are automatically sent when LLDP is active, but you can disable them as needed using the **lldp tlv-select <Optional TLV> disable** command.

LLDP Frame Format

LLDP frames use the IEEE 802.3 format, which consists of the following fields:

- Destination address (6 bytes)—Uses a multicast address of 01-80-C2-00-00-0E.
- Source address (6 bytes)—MAC address of the sending device or port.
- LLDP Ethertype (2 bytes)—Uses 88-CC.
- LLDP PDU (1500 bytes)—LLDP payload consisting of TLVs.
- FCS (4 bytes)—Cyclic Redundancy Check (CRC) for error checking.

LLDP TLV Format

LLDP TLVs carry the information about neighboring devices within the LLDP PDU using the following basic format:

- TLV Header (16 bits), which includes the following fields:
  - TLV Type (7 bits)
  - TLV Information String Length (9 bits)
- TLV Information String (0 to 511 bytes)

LLDP Operation

LLDP is a one-way protocol. The basic operation of LLDP consists of a device enabled for transmit of LLDP information sending periodic advertisements of information in LLDP frames to a receiving device.

Devices are identified using a combination of the Chassis ID and Port ID TLVs to create an MSAP (MAC Service Access Point). The receiving device saves the information about a neighbor in a remote lldp cache for a certain amount of time as specified in the TTL TLV received from the neighbor, before aging and removing the information.

LLDP supports the following additional operational characteristics:

- LLDP can operate independently in transmit or receive modes. On globall lldp enablement, the default mode is to operate in both transmit and receive modes.
- LLDP operates as a slow protocol with transmission speeds not greater than one frame per five seconds.
- LLDP packets are sent when the following occurs:
  - The packet update frequency specified by the **lldp timer** command is reached. The default is 30 seconds.
  - When a change in the values of the managed objects occurs from the local system's LLDP MIB.
  - When LLDP is activated on an interface (3 frames are sent upon activation similar to CDP).
- When an LLDP frame is received, the LLDP remote services and PTOPO MIBs are updated with the information in the TLVs.
- LLDP supports the following actions on these TLV characteristics:
  - Interprets a neighbor TTL value of 0 as a request to automatically purge the information of the transmitting device. These shutdown LLDPDUs are typically sent prior to a port becoming inoperable.
  - An LLDP frame with a malformed mandatory TLV is dropped.
  - A TLV with an invalid value is ignored.
  - A copy of an unknown organizationally-specific TLV is maintained if the TTL is non-zero, for later access through network management.

Supported LLDP Functions

The router supports the following LLDP functions:

- IPv4 and IPv6 management addresses—In general, both IPv4 and IPv6 addresses will be advertised if they are available, and preference is given to the address that is configured on the transmitting interface.

  If the transmitting interface does not have a configured address, then the system populates the TLV with an address from another interface. The advertised LLDP IP address is implemented according to the following priority order of IP addresses for interfaces on the router:

  - Locally configured address on the transmitting interface
  - MgmtEth0/RSP0RP0/CPU0/0
  - MgmtEth0/RSP0RP0/CPU0/1
  - MgmtEth0/RSP1RP1/CPU0/0
  - MgmtEth0/RSP1RP1/CPU0/1
  - Loopback interfaces

    There are some differences between IPv4 and IPv6 address management in LLDP:

- For IPv4, as long as the IPv4 address is configured on an interface, it can be used as an LLDP management address.
- For IPv6, after the IPv6 address is configured on an interface, the interface status must be Up and pass the DAD (Duplicate Address Detection) process before it is can be used as an LLDP management address.
- LLDP is supported for the nearest physically attached, non-tunneled neighbors.
- LLDP is supported for Ethernet interfaces, L3 subinterfaces, bundle interfaces, and L3 bundle subinterfaces.

Unsupported LLDP Functions

The following LLDP functions are not supported on the router:

- LLDP-MED organizationally unique extension—However, interoperability still exists between other devices that do support this extension.
- Tunneled neighbors, or neighbors more than one hop away.
- LLDP TLVs cannot be disabled on a per-interface basis; However, certain optional TLVs can be disabled globally.
- LLDP SNMP trap lldpRemTablesChange.

Configuring Fabric Bandwidth

The following table allows you to configure the fabric bandwidth threshold level. This is a systemwide configuration that applies to all the line card NPUs in the chassis.

Table 5: Types of Fabric Bandwidth

Type	Definition	Default	Behavior
Total fabric bandwidth	Bandwidth between a line card and all 8 fabric cards in the chassis	Not Applicable	Constant
Total required bandwidth	Total fabric bandwidth x (bandwidth Threshold config)	5%	Configured value
Lower required bandwidth	Total fabric bandwidth x (bandwidth threshold config -10)	5%	Calculated value. When the available fabric bandwidth falls below this value, interfaces go down to avoid traffic black hole.
Available bandwidth	Total fabric bandwidth available for traffic. This is a function of number of links from un-isolated Fast Ethernet devices connecting to the NPU x 50G.	Not Applicable	—

The default bandwidth threshold value is 5%, you can change the value by using the following command in increments of 10.

Following is a systemwide configuration that applies to all the line card NPUs in the chassis.

```
Router# configure terminal
Router (config)# hw-module profile bw-threshold <value>
Router (config)# commit
```

If the *available bandwidth* drops to being equal to or below the *lower required bandwidth*, then all network interfaces of the line card get disabled to reroute the traffic.

When total *available bandwidth* reaches the *required bandwidth*, then the network interfaces of the line card get re-enabled.

For example, consider that the bandwidth threshold is set to 20%.

- If the available bandwidth goes below 10%, then network interfaces of the line cards are shut automatically.
- If the available bandwidth goes above 20%, then network interfaces of the line cards are unshut automatically.

How to Configure Ethernet

This section provides the following configuration procedures:

Configuring LLDP

Note LLDP is not supported on the FP-X line cards.

This section includes the following configuration topics for LLDP:

LLDP Default Configuration

This table shows the values of the LLDP default configuration on the router. To change the default settings, use the LLDP global configuration and LLDP interface configuration commands.

LLDP Function	Default
LLDP global state	Disabled
LLDP holdtime (before discarding)	120 seconds
LLDP timer (packet update frequency)	30 seconds
LLDP reinitialization delay	2 seconds
LLDP TLV selection	All TLVs are enabled for sending and receiving.
LLDP interface state	Enabled for both transmit and receive operation when LLDP is globally enabled.

Enabling LLDP Per Interface

When you enable LLDP globally, all interfaces that support LLDP are automatically enabled for both transmit and receive operations. However, if you want to enable LLDP per interface, perform the following configuration steps:

```
RP/0/RSP0/CPU0:ios(config)# int HundredGigE 0/2/0/0
RP/0/RSP0/CPU0:ios(config-if)# no sh
RP/0/RSP0/CPU0:ios(config-if)#commit
RP/0/RSP0/CPU0:ios(config-if)#lldp ?
RP/0/RSP0/CPU0:ios(config-if)#lldp enable
RP/0/RSP0/CPU0:ios(config-if)#commit
```

Running configuration

```
RP/0/RSP0/CPU0:ios#sh running-config
Wed Jun 27 12:40:21.274 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Wed Jun 27 00:59:29 2018 by UNKNOWN
!
interface HundredGigE0/1/0/0
 shutdown
!
interface HundredGigE0/1/0/1
 shutdown
!
interface HundredGigE0/1/0/2
 shutdown
!
```

```
interface HundredGigE0/2/0/0
 Shutdown
!
interface HundredGigE0/2/0/1
 shutdown
!
interface HundredGigE0/2/0/2
 shutdown
!
end
```

Verification

```
Verifying the config
==================
RP/0/RSP0/CPU0:ios#sh lldp interface <===== LLDP enabled only on GigEth0/2/0/0
Wed Jun 27 12:43:26.252 IST


HundredGigE0/2/0/0:
        Tx: enabled
        Rx: enabled
        Tx state: IDLE
        Rx state: WAIT FOR FRAME
RP/0/RSP0/CPU0:ios#

RP/0/RSP0/CPU0:ios# show lldp neighbors
Wed Jun 27 12:44:38.977 IST
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
ios             Gi0/2/0/0           120        R              Gi0/2/0/0       <====== LLDP
 enabled only on GigEth0/2/0/0 and neighborship seen for the same.

Total entries displayed: 1

RP/0/RSP0/CPU0:ios#
```

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. For more information about how to selectively disable LLDP receive or transmit operations for an interface, see the *Disabling LLDP Receive and Transmit Operation for an Interface section*.

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

Attribute	Default	Range	Description
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//CPU0:router # configure`
2. `RP/0//CPU0:router(config) #lldp`
3. `end or commit`

Running configuration

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

RP/0/RP0/CPU0:turin-5#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:turin-5#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
```

Configuring Global LLDP Operational Characteristics

When you enable LLDP globally on the router using the **lldp** command, these defaults are used for the protocol.

To modify the global LLDP operational characteristics such as the LLDP neighbor information holdtime, initialization delay, or packet rate, complete the following steps:

Step 1 **Example:**

```
/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **lldp holdtime** *seconds*

Example:

```
RP/0//CPU0:router(config)#lldp holdtime 60
```

(Optional) Specifies the length of time that information from an LLDP packet should be held by the receiving device before aging and removing it.

Step 3 **lldp reinit** *seconds*

Example:

```
RP/0//CPU0:router(config)# lldp reinit 4
```

(Optional) Specifies the length of time to delay initialization of LLDP on an interface.

Step 4 **lldp timer** *seconds*

Example:

```
RP/0//CPU0:router(config)#lldp reinit 60
```

(Optional) Specifies the LLDP packet rate.

Step 5 **end** or **commit**

Example:

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

or

```
RP/0//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.

Disabling Transmission of Optional LLDP TLVs

Certain TLVs are classified as mandatory in LLDP packets, such as the Chassis ID, Port ID, and Time to Live (TTL) TLVs. These TLVs must be present in every LLDP packet. You can suppress transmission of certain other optional TLVs in LLDP packets.

To disable transmission of optional LLDP TLVs, complete the following steps:

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **lldp tlv-select** *tlv-name* **disable**

Example:

```
RP/0/RSP0/CPU0:router(config)# lldp tlv-select system-capabilities disable
```

(Optional) Specifies that transmission of the selected TLV in LLDP packets is disabled. The *tlv-name* can be one of the following LLDP TLV types:

- **management-address**
- **port-description**
- **system-capabilities**
- **system-description**
- **system-name**

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

Example:

```
RP/0/RSP0/CPU0:router(config)# end
```

or

```
RP/0/RSP0RP0/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.

Disabling LLDP Receive and Transmit Operation for an Interface

When you enable LLDP globally on the router, all supported interfaces are automatically enabled for LLDP receive and transmit operation. You can override this default by disabling these operations for a particular interface.

To disable LLDP receive and transmit operations for an interface, complete the following steps:

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **interface HundredGigE 0/2/0/0**

Example:

```
RP/0/RSP0RP0/CPU0:router(config)#interface HundredGigE 0/2/0/0
```

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

- HundredGigE
- TenGigE

Step 3 **lldp**

Example:

```
RP/0/RSP0/CPU0:router(config-if)#lldp
```

(Optional) Enters LLDP configuration mode for the specified interface.

Step 4 **receive disable**

Example:

```
RP/0/RSP0/CPU0:router(config-lldp)#receive disable
```

(Optional) Disables LLDP receive operations on the interface.

Step 5 **transmit disable**

Example:

```
RP/0/RSP0/CPU0:router(config-lldp)#transmit disable
```

(Optional) Disables LLDP transmit operations on the interface.

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

Example:

```
RP/0/RSP0/CPU0:router(config)# end
```

or

```
RP/0/RSP0/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.

Verifying the LLDP Configuration

This section describes how you can verify the LLDP configuration both globally and for a particular interface.

Verifying the LLDP Global Configuration

To verify the LLDP global configuration status and operational characteristics, use the **show lldp** command as shown in the following example:

```
RP/0/RSP0/CPU0:router# show lldp
Wed Apr 13 06:16:45.510 DST
Global LLDP information:
        Status: ACTIVE
        LLDP advertisements are sent every 30 seconds
        LLDP hold time advertised is 120 seconds
        LLDP interface reinitialisation delay is 2 seconds
```

If LLDP is not enabled globally, the following output appears when you run the **show lldp** command:

```
RP/0/RSP0/CPU0:router# show lldp
Wed Apr 13 06:42:48.221 DST
% LLDP is not enabled
```

Verifying the LLDP Interface Configuration

To verify the LLDP interface status and configuration, use the **show lldp interface** command as shown in the following example:

```
RP/0/RSP0/CPU0:router# show lldp interface HundredGigE 0/1/0/7
Wed Apr 13 13:22:30.501 DST


HundredGigE0/1/0/7:
        Tx: enabled
        Rx: enabled
        Tx state: IDLE
        Rx state: WAIT FOR FRAME
```

To monitor and maintain LLDP on the system or get information about LLDP neighbors, use one of the following commands:

Command	Description
clear lldp counters	Resets LLDP traffic counters or LLDP neighbor information.
show lldp entry	Displays detailed information about LLDP neighbors.
show lldp errors	Displays LLDP error and overflow statistics.
show lldp neighbors	Displays information about LLDP neighbors.
show lldp traffic	Displays statistics for LLDP traffic.

Configuration Examples for Ethernet

This section provides the following configuration examples:

Configuring an Ethernet Interface: Example

The following example shows how to configure an interface for a 10-Gigabit Ethernet modular services card:

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# interface TenGigE 0/0/0/1
RP/0//CPU0:router(config-if)# ipv4 address 172.18.189.38 255.255.255.224
RP/0//CPU0:router(config-if)# flow-control ingress
RP/0//CPU0:router(config-if)# mtu 1448
RP/0//CPU0:router(config-if)# mac-address 0001.2468.ABCD
RP/0//CPU0:router(config-if)# no shutdown
RP/0//CPU0:router(config-if)# end
Uncommitted changes found, commit them? [yes]: yes


RP/0//CPU0:router# show interfaces TenGigE 0/0/0/1


TenGigE0/0/0/1 is down, line protocol is down
  Hardware is TenGigE, address is 0001.2468.abcd (bia 0001.81a1.6b23)
  Internet address is 172.18.189.38/27
  MTU 1448 bytes, BW 10000000 Kbit
```

```
     reliability 0/255, txload Unknown, rxload Unknown
Encapsulation ARPA,
  Full-duplex, 10000Mb/s, LR
  output flow control is on, input flow control is on
  Encapsulation ARPA,
ARP type ARPA, ARP timeout 01:00:00
  Last clearing of "show interface" counters never
  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 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

Configuring LLDP: Examples

The following example shows how to enable LLDP globally on the router and modify the default LLDP operational characteristics:

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# lldp
RP/0//CPU0:router(config)# lldp holdtime 60
RP/0//CPU0:router(config)# lldp reinit 4
RP/0//CPU0:router(config)# lldp timer 60
RP/0//CPU0:router(config)# commit
```

The following example shows how to disable a specific Gigabit Ethernet interface for LLDP transmission:

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# interface HundredGigE 0/2/0/0
RP/0//CPU0:router(config-if)# lldp
RP/0//CPU0:router(config-lldp)# transmit disable
```

Where to Go Next

When you have configured an Ethernet interface, you can configure individual VLAN subinterfaces on that Ethernet interface.

For information about modifying Ethernet management interfaces for the shelf controller (SC), route processor (RP), and distributed RP, see the Advanced Configuration and Modification of the Management Ethernet Interface later in this document.

For information about IPv6 see the Implementing Access Lists and Prefix Lists on Cisco IOS XR Software module in the Cisco IOS XR IP Addresses and Services Configuration Guide.

Configuring Physical Ethernet Interfaces

Use this procedure to create a basic Ethernet interface configuration.

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** [**TenGigE FortyGigE HundredGigE FourHundredGigE**] *interface-path-id*

Example:

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

(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 **show interfaces** [**TenGigE FortyGigE HundredGigE FourHundredGigE**] *interface-path-id*

Example:

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

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

- 10GigE
- 40GigE
- 100GigE

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

- 400GigE

The examples of *interface-path-id* ranges are:

- **TenGigE** — 0/0/0/0 - 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 **flow-control** {**bidirectional**| **egress** | **ingress**}

Example:

```
RP/0/RP0/CPU0:router(config-if)# flow control ingress
```

(Optional) Enables the sending and processing of flow control pause frames.

- **egress**—Enables the sending of flow control pause frames in egress.
- **ingress**—Enables the processing of received pause frames on ingress.
- **bidirectional**—Enables the sending of flow control pause frames in egress and the processing of received pause frames on ingress.

Step 7 **mtu** *bytes*

Example:

```
RP/0/RP0/CPU0:router(config-if)# mtu 1448
```

(Optional) Sets the MTU value for the interface.

- The default is 1514 bytes for normal frames and 1518 bytes for 802.1Q tagged frames.
- The range for 100-Gigabit Ethernet mtu values is 64 bytes to 65535 bytes.

Step 8 **no shutdown**

Example:

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

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

Step 9 **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 10 **show interfaces** [**TenGigE FortyGigE HundredGigE FourHundredGigE**] *interface-path-id*

Example:

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

(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 0/1/0/1
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 1448

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 0/5/0/24
HundredGigE0/5/0/24 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 0/5/0/24

interface HundredGigE 0/5/0/24
 mtu 9216
 service-policy input linerate
 service-policy output elinerate
 ipv4 address 3.24.1.1 255.255.255.0
 ipv6 address 3:24:1::1/64
 flow ipv4 monitor perfv4 sampler fsm ingress
!
```

How to Configure Interfaces in Breakout Mode

Information About Breakout

The router supports transmission of traffic in the breakout mode. The breakout mode enables a 40 Gigabit Ethernet port to be split into four independent and logical 10 Gigabit Ethernet ports. The 4x10 breakout mode is supported on the following types of 40G modules:

- QSFP-4x10-LR-S
- QSFP-40G-SR4

Guidelines and Restrictions for Breakout Mode

- The native 40G mode on QSFP-40G-SR4 is not supported.
- The 36-port QSFP56-DD 400 GbE Line Card does not support the 4x10G breakout.
- If port 24~35 on Cisco 8200 Series Routers are used for 4x10G breakout configuration, you must use even port numbers (24/26/28/30/32/34) for breakout configurations.
- The system automatically disables the ports 25/27/29/31/33/35 if the ports with even numbers (24/26/28/30/32/34) are used for breakout configurations.
- Use the *hw-module port-range* command to set the port range for the breakout configuration in the global configuration.
- To remove the global *hw-module port-range* configuration, you must first remove the ‘breakout 4x10’ configuration under the controller.

- For 4x10G breakout on 48-port Line Card, only QSFP-4x10-LR-S module is supported.

Configure Breakout in a Port

Configuring breakout in a port:

```
RP/0/RP0/CPU0:uut# configure
Fri Oct 11 23:58:47.165 UTC
RP/0/RP0/CPU0:uut(config)# controller optics 0/1/0/28
RP/0/RP0/CPU0:uut(config-Optics)# breakout 4x10
RP/0/RP0/CPU0:uut(config-Optics)# commit
Fri Oct 11 23:59:51.261 UTC
RP/0/RP0/CPU0:uut(config-Optics)# end
RP/0/RP0/CPU0:uut#
```

Remove the Breakout Configuration

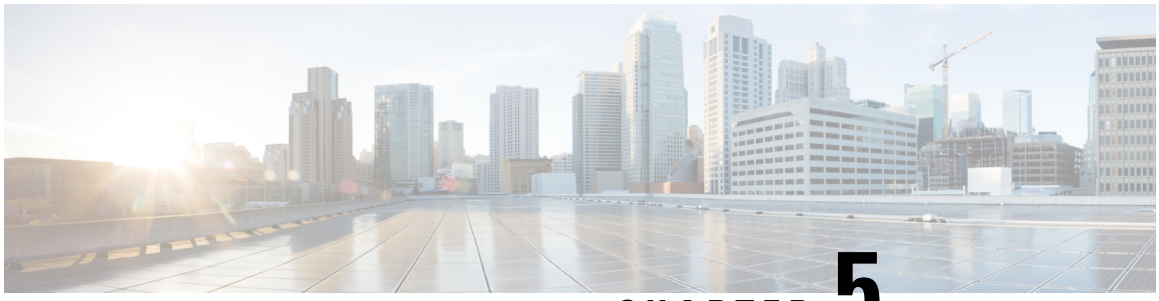
Removing the breakout configuration:

```
RP/0/RP0/CPU0:uut# configure
Sat Oct 12 00:01:38.673 UTC
RP/0/RP0/CPU0:uut(config)# controller optics 0/1/0/28
RP/0/RP0/CPU0:uut(config-Optics)# no breakout 4x10
RP/0/RP0/CPU0:uut(config-Optics)# commit
Sat Oct 12 00:01:55.864 UTC
RP/0/RP0/CPU0:uut(config-Optics)# end
```

Verify a Breakout Configuration

Verifying a breakout configuration:

```
RP/0/RP0/CPU0:uut# show running-config controller optics 0/1/0/28
Sat Oct 12 00:11:33.962 UTC
controller Optics0/1/0/28
breakout 4x10
!

RP/0/RP0/CPU0:uut# show int br location 0/1/CPU0 | i Te
Sat Oct 12 00:11:38.609 UTC
       Te0/1/0/27/0          up          up               ARPA 10000   10000000
       Te0/1/0/27/1          up          up               ARPA 10000   10000000
       Te0/1/0/27/2          up          up               ARPA 10000   10000000
       Te0/1/0/27/3          up          up               ARPA 10000   10000000
       Te0/1/0/28/0          up          up               ARPA 10000   10000000
       Te0/1/0/28/1          up          up               ARPA 10000   10000000
       Te0/1/0/28/2          up          up               ARPA 10000   10000000
       Te0/1/0/28/3          up          up               ARPA 10000   10000000

.
```



CHAPTER 5

IP Event Dampening

The IP Event Dampening feature introduces a configurable exponential decay mechanism to suppress the effects of excessive interface flapping events on routing protocols and routing tables in the network. This feature allows the network operator to configure a router to automatically identify and selectively dampen a local interface that is flapping.

Guidelines and Limitations

See the following guidelines and limitations before configuring IP Event Dampening feature:

- Due to changes in the netstack-IP component, all the IP clients observe the impact of dampening or interface.
- For each flap of the interface, a certain penalty is added. The penalty decays exponentially whose parameters are configured.
- When penalty exceeds a certain high level, the interface is dampened. It is unsuppressed when the penalty decays below a low level.
- When an interface is dampened, the IP address and the static routes are removed from the interface. All the clients of IP get an IP delete notification.
- When an interface is unsuppressed, the IP address and the relevant routes are added back. All the clients of IP get an IP address add notification for all the IP addresses of the interface.
- All Layer 3 interfaces that are configured on the Ethernet interface, port changes, and SVI support this feature.

IP Event Dampening Overview

Interface state changes occur when interfaces are administratively brought up or down or if an interface changes state. When an interface changes state or flaps, routing protocols are notified of the status of the routes that are affected by the change in state. Every interface state change requires all affected devices in the network to recalculate best paths, install or remove routes from the routing tables, and then advertise valid routes to peer routers. An unstable interface that flaps excessively can cause other devices in the network to

consume substantial amounts of system processing resources and cause routing protocols to lose synchronisation with the state of the flapping interface.

The IP Event Dampening feature introduces a configurable exponential decay mechanism to suppress the effects of excessive interface flapping events on routing protocols and routing tables in the network. This feature allows the network operator to configure a router to automatically identify and selectively dampen a local interface that is flapping. Dampening an interface removes the interface from the network until the interface stops flapping and becomes stable. Configuring the IP Event Dampening feature improves convergence times and stability throughout the network by isolating failures so that disturbances are not propagated. This, in turn, reduces the utilisation of system processing resources by other devices in the network and improves overall network stability.

Interface State Change Events

This section describes the interface state change events of the IP Event Dampening feature. This feature employs a configurable exponential decay mechanism that is used to suppress the effects of excessive interface flapping or state changes. When the IP Event Dampening feature is enabled, flapping interfaces are dampened from the perspective of the routing protocol by filtering excessive route updates. Flapping interfaces are identified, assigned penalties, suppressed if necessary, and made available to the network when the interface stabilizes.

Suppress Threshold

The suppress threshold is the value of the accumulated penalty that triggers the router to dampen a flapping interface. The flapping interface is identified by the router and assigned a penalty for each up and down state change, but the interface is not automatically dampened. The router tracks the penalties that a flapping interface accumulates. When the accumulated penalty reaches the default or preconfigured suppress threshold, the interface is placed in a dampened state.

Half-Life Period

The half-life period determines how fast the accumulated penalty can decay exponentially. When an interface is placed in a dampened state, the router monitors the interface for additional up and down state changes. If the interface continues to accumulate penalties and the interface remains in the suppress threshold range, the interface will remain dampened. If the interface stabilises and stops flapping, the penalty is reduced by half after each half-life period expires. The accumulated penalty will be reduced until the penalty drops to the reuse threshold. The configurable range of the half-life period timer is from 1 to 45 minutes. The default half-life period timer is 1 minute.

Reuse Threshold

When the accumulated penalty decreases until the penalty drops to the reuse threshold, the route is unsuppressed and made available to other devices in the network. The range of the reuse value is from 1 to 20000 penalties. The default value is 750 penalties.

Maximum Suppress Time

The maximum suppress time represents the maximum time an interface can remain dampened when a penalty is assigned to an interface. The maximum suppress time can be configured from 1 to 255 seconds. The maximum penalty is truncated to maximum 20000 unit. The maximum value of the accumulated penalty is calculated based on the maximum suppress time, reuse threshold, and half-life period.

Affected Components

When an interface is not configured with dampening, or when an interface is configured with dampening but is not suppressed, the routing protocol behavior as a result of interface state transitions is not changed by the IP Event Dampening feature. However, if an interface is suppressed, the routing protocols and routing tables are immune to any further state transitions of the interface until it is unsuppressed.

Route Types

The following interfaces are affected by the configuration of this feature:

- Connected routes:
  - The connected routes of dampened interfaces are not installed into the routing table.
  - When a dampened interface is unsuppressed, the connected routes will be installed into the routing table if the interface is up.
- Static routes:
  - Static routes assigned to a dampened interface are not installed into the routing table.
  - When a dampened interface is unsuppressed, the static route will be installed into the routing table if the interface is up.

Note Only the primary interface can be configured with this feature, and all subinterfaces are subject to the same dampening configuration as the primary interface. IP Event Dampening does not track the flapping of individual subinterfaces on an interface.

Supported Protocols

All the protocols that are used are impacted by the IP Event Dampening feature. The IP Event Dampening feature supports Border Gateway Protocol (BGP), Enhanced Interior Gateway Routing Protocol (EIGRP), Hot Standby Routing Protocol (HSRP), Open Shortest Path First (OSPF), Routing Information Protocol (RIP), and VRRP. Ping and SSH to the concerned interface IP address does not work.

Note The IP Event Dampening feature has no effect on any routing protocols if it is not enabled or an interface is not dampened.

How to Configure IP Event Dampening

Enabling IP Event Dampening

The `dampening` command is entered in interface configuration mode to enable the IP Event Dampening feature. If this command is applied to an interface that already has dampening configured, all dampening states are reset and the accumulated penalty will be set to 0. If the interface has been dampened, the accumulated penalty will fall into the reuse threshold range, and the dampened interface will be made available to the network. The flap counts, however, are retained.

Table 6: Procedure

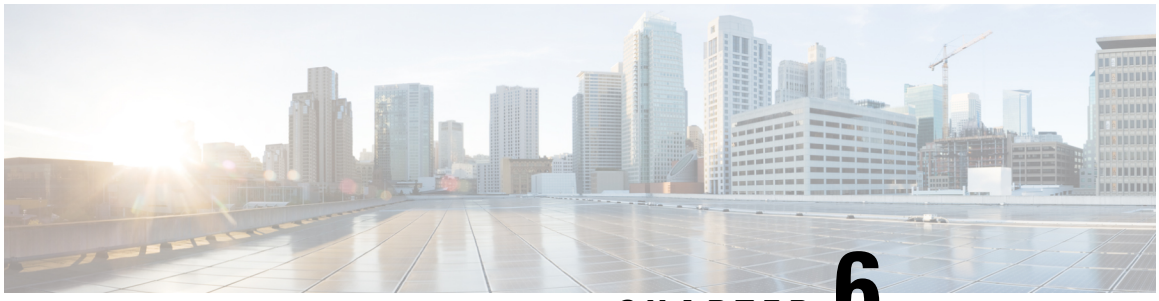
Steps	Command or Action	Purpose
Step 1	**configure terminal**	Enters global configuration mode.
Step 2	**interface** *type number*	Enters interface configuration mode and configures the specified interface.
Step 3	**dampening** [*half-life-period reuse-threshold*] [*suppress-threshold max-suppress* [*restart-penalty*]]	Enables interface dampening. • Entering the `dampening` command without any arguments enables interface dampening with default configuration parameters. • When manually configuring the timer for the *restart-penalty* argument, the values must be manually entered for all arguments.
Step 4	**end**	Exits interface configuration mode.

Verifying IP Event Dampening

Use the `show dampening interface` or `show interface dampening` commands to verify the configuration of the IP Event Dampening feature.

Table 7: Procedure

Steps	Command or Action	Purpose
Step 1	**show dampening interface**	Displays dampened interfaces.
Step 2	**show interface dampening**	Displays dampened interfaces on the local router.



CHAPTER 6

Configuring Link Bundling

This module describes the configuration of link bundle interfaces on the Cisco 8000 Series Router.

A link bundle is a group of one or more ports that are aggregated together and treated as a single link.

Each bundle has a single MAC, a single IP address, and a single configuration set (such as ACLs).

Note The router supports both Layer 2 and Layer 3 Link Bundles. If the Link Bundle is a Layer 3 interface, the system requires an IP address. If the Link Bundle is a Layer 2 interface, the system does not require an IP address. A Link Bundle on the router may contain Layer 2 and Layer 3 subinterfaces within it. In which case, the Layer 3 subinterfaces require IP addresses, but the Link Bundle interface does not require an IP address.

The router supports bundling for these types of interfaces:

- Ethernet interfaces

Feature History for Configuring Link Bundling

Release	Modification
Release 7.0.11	Support for this feature added on the router.

Limitations and Compatible Characteristics of Ethernet Link Bundles

This list describes the properties and limitations 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 are 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 a maximum of 512 bundle interfaces and a default of 64 member links per bundle.
- A single router supports up to a maximum of 1024 bundle subinterfaces and up to a maximum of 64 member links per bundle.
- 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.
- The MAC address that is set on the bundle becomes the MAC address of the links within that bundle.
- 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.
- QoS is supported and is applied proportionally on each bundle member.
- 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.

Prerequisites for Configuring Link Bundling on a Router

Before configuring Link Bundling, ensure that you meet the following tasks and conditions:

- You know the interface IP address (Layer 3 only).
- You know the links that you must include in the bundle that you are configuring.
- If you are configuring an Ethernet link bundle, you must install Ethernet line cards on the router.

Note For more information about physical interfaces, PLIMs, and modular services cards, refer to the *Cisco 8000 Series Router Hardware Installation Guide*.

Information About Configuring Link Bundling

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

Link Bundling Overview

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. The system assigns a virtual interface to the bundled link. You can dynamically add and delete component links from the virtual interface.

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

A link bundle is a group of ports that the system bundles together and the group then acts as a single link. Following are the advantages of link bundles:

Note A single router supports 512 bundles per system.

- 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. You can add bandwidth without interrupting the packet flow.

All the individual links within a single bundle must be of the same type.

Cisco IOS XR software supports the following methods 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. The system automatically removes the links from a bundle that are incompatible or have failed.

Link Aggregation Through LACP

The optional Link Aggregation Control Protocol (LACP) is defined in the IEEE 802 standard. LACP communicates between two directly connected systems (or peers) to verify the compatibility of bundle members. For the router, the peer can be either another router or a switch. LACP monitors the operational state of link bundles to ensure the following:

- All links terminate on the same two systems.

- Both systems consider the links to be part of the same bundle.
- All links have the appropriate settings on the peer.

LACP transmits frames containing the local port state and the local view of the partner system's state. The system analyzes these frames to ensure that both the systems are in agreement.

IEEE 802.3ad Standard

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

For each link configured as a 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 by using 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. This determines the compatibility of the links that are configured to be members of a bundle.

Bundle MAC addresses in the router come from a set of reserved MAC addresses in the backplane. This MAC address stays with the bundle as long as the bundle interface exists. The bundle uses this MAC address until you configure a different MAC address. The member links use the bundle MAC address when passing the 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.

Configuring LACP Fallback

This section describes how to configure the LACP Fallback feature.

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.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface-specific configuration commands. Use the **exit** command to exit from the interface configuration submode back to the normal global configuration mode.

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.

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 64 member links
- Up to 1280 packets per second (pps)

After 6 missed 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.

Load Balancing

Load balancing is a forwarding mechanism that distributes traffic over multiple links that are based on certain parameters. The router support load balancing for all links in a bundle using Layer 2, Layer 3, and Layer 4 routing information.

This section describes the load balancing support on link bundles.

For more information about other forms of load balancing on the router, see the following:

- Per-flow load balancing on non-bundle interfaces using Layer 3 and 4 routing information.
- Pseudowire (PW) Load Balancing beginning in Cisco IOS XR 4.0.1.

Layer 3 Egress Load Balancing on Link Bundles

Layer 3 load balancing support began on the router in Cisco IOS XR 7.0.11 release.

Layer 3 load balancing for link bundles is enabled globally by default.

The ingress linecard does bundle member selection and forwards the packet to the linecard and network processor (NP) corresponding to the selected bundle member. The same hash value is used for both ingress and egress linecards. Therefore, even though the egress linecard also does bundle member selection, it selects the same bundle member that was selected by the ingress linecard.

Multicast IPv4 and IPv6 Traffic

For outbound multicast IPv4 or IPv6 traffic, a set of egress linecards is predetermined by the system. If a bundle interface or bundle subinterface is an outgoing interface, the system selects the bundle member for

each outgoing interface in a route based on the multicast group address. This helps with load distribution of multicast routed traffic to different bundle members, while providing traffic sequencing within a specific route.

The egress linecard does NP selection using the same approach, when bundle members are spread across multiple NPs within the egress linecard.

When the packet arrives on an egress NP, it uses the 5-tuple hash to select a bundle member within an NP for each packet. This provides better resiliency for bundle member state changes within an NP.

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.

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 0/1/0/1
```

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.

Step 5 **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.

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 0/1/0/1
   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 0/1/0/1
   bundle id 1 mode active
   lacp period short
   commit


config
interface HundredGigE 0/1/0/1
    lacp period short transmit 100
   commit


config
interface HundredGigE 0/1/0/1
   lacp period short receive 100
   commit
```

Configuring Custom LACP Short Period Time Intervals

Step 1 **interface Bundle-Ether** *bundle-id*

Example:

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

Creates and names a new Ethernet link bundle.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface-specific configuration commands. Use the **exit** command to exit from the interface configuration submode back to the normal global configuration mode.

Step 2 **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 3 **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 4 **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 5 **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 6 **exit**

Example:

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

Exits the interface configuration submode.

Step 7 **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).

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

Step 8 **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 4094 inclusive (0 and 4095 are reserved).

Step 9 **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 10 **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 11 **exit**

Example:

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

Exits subinterface configuration mode for the VLAN subinterface.

Step 12 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 13 **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)?`
- 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 14 **exit**

Example:

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

Exits interface configuration mode.

Step 15 **exit**

Example:

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

Enters global configuration mode.

Step 18 **interface** {**GigabitEthernet** | **TenGigE**} *interface-path-id*

Example:

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

Enters the 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.

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.

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

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 **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)?`
- 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 23 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 24 **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 25 **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.

QoS and Link Bundling

On the router, when the system applies QoS on the bundle for either the ingress or egress direction, QoS is applied at each member interface. For complete information on configuring QoS on link bundles on the router, refer to the *Cisco 8000 Series Aggregation Services Router Modular Quality of Service Configuration Guide* and the *Cisco 8000 Series Aggregation Services Router Modular Quality of Service Command Reference*.

Link Bundle Configuration Overview

The following steps provide a general overview of the link bundle configuration process. Ensure that you clear all previous network layer configuration before adding it 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 that you created in Step 1 with the **bundle id** command in the interface configuration submode.

You can add up to 64 links to a single bundle.

Note The system configures a link as a member of a bundle from the interface configuration submode for that link.

Nonstop Forwarding During Card Failover

Cisco IOS XR software supports nonstop forwarding during a failover between active and standby paired RP cards. Nonstop forwarding ensures that there is no change in the state of the link bundles when a failover occurs.

For example, if an active RP fails, the standby RP becomes operational. The system replicates the configuration, node state, and checkpoint data of the failed RP to the standby RP. The bundled interfaces are present when the standby RP becomes the active RP.

Note Failover is always onto the standby RP.

You do not need to configure anything to guarantee that the system maintains the standby interface configurations.

Link Failover

When one member link in a bundle fails, the system redirects the to the remaining operational member links and traffic flow remains uninterrupted.

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.

To do so, you can use the **lacp switchover suppress-flaps** command.

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.

How to Configure Link Bundling

This section contains the following procedures:

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-lacp.yang`, `openconfig-if-aggregate.yang` OpenConfig data models, `Cisco-IOS-XR-bundlemgr-oper.yang` Cisco IOS XR native data model or `Cisco-IOS-XR-um-lacp-cfg.yang` Unified data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

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.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface specific configuration commands are entered. Use the **exit** command to exit from the interface configuration submode back to the normal global configuration mode.

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 • On the router, 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.

Note • The priority of the active and standby links is based on the value of the **bundle port-priority** command.

Step 7 **lacp fast-switchover**

Example:

```
RP/0/RP0/CPU0:router(config-if)# lacp fast-switchover
```

(Optional) If you enabled 1:1 link protection (you set the value of the **bundle maximum-active links** command to 1) on a bundle with member links running LACP, you can optionally disable the wait-while timer in the LACP state machine. Disabling this timer causes a bundle member link in standby mode to expedite its normal state negotiations, thereby enabling a faster switchover from a failed active link to the standby link.

Step 8 **exit**

Example:

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

Exits interface configuration submode for the Ethernet link bundle.

Step 9 **interface** {**GigabitEthernet** | **TenGigE**} interface-path-id

Example:

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

Enters interface configuration mode for the specified interface.

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.

Step 10 **bundle id** *bundle-id* [**mode** {**active** | **on** | **passive**}]

Example:

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

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 11 **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 12 **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 13 **exit**

Example:

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

Exits interface configuration submode for the Ethernet interface.

Step 14 **bundle id** *bundle-id* [**mode** {**active** | **passive** | **on**}] **no shutdown exit**

Example:

```
RP/0/RP0/CPU0:router(config)# interface GigabitEthernet 1/0/2/1

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 GigabitEthernet 1/0/2/3

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 15 **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 16 **exit**

Example:

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

Exits interface configuration mode.

Step 17 **exit**

Example:

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

Exits global configuration mode.

Step 18 Perform Step 1 through Step 15 on the remote end of the connection.

Brings up the other end of the link bundle.

Step 19 **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 20 **show lacp bundle Bundle-Ether** *bundle-id*

Example:

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

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

Configuring VLAN Bundles

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

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

These tasks are described 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.

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.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface-specific configuration commands. Use the **exit** command to exit from the interface configuration submode back to the normal global configuration mode.

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 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 5 **exit**

Example:

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

Exits the interface configuration submode.

Step 6 **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).

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

Step 7 **encapsulation dot1q**

Example:

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

Sets the Layer 2 encapsulation of an interface.

Step 8 **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 9 **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 10 **exit**

Example:

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

Exits subinterface configuration mode for the VLAN subinterface.

Step 11 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 12 **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 13 **exit**

Example:

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

Exits interface configuration mode.

Step 14 **exit**

Example:

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

Exits global configuration mode.

Step 15 **configure**

Example:

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

Enters global configuration mode.

Step 16 **interface** {**GigabitEthernet** | **TenGigE**}*interface-path-id*

Example:

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

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.

Step 17 **lacp fast-switchover**

Example:

```
RP/0/RP0/CPU0:router(config-if)# lacp fast-switchover
```

(Optional) If you enabled 1:1 link protection (you set the value of the **bundle maximum-active links** command to 1) on a bundle with member links running LACP, you can optionally disable the wait-while timer in the LACP state machine. Disabling this timer causes a bundle member link in standby mode to expedite its normal state negotiations, thereby enabling a faster switchover from a failed active link to the standby link.

VLANs on an Ethernet Link Bundle

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

- The maximum number of VLANs allowed per bundle is 2048.
- The maximum number of bundled VLANs allowed per router is 2048.

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, the system supports all VLAN subinterface configuration on that link bundle.

VLAN subinterfaces can support Ethernet Flow Points (EFPs) and Layer 3 services.

You can configure Layer 3 VLAN subinterfaces as follows:

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

Configuration Examples for Link Bundling

This section contains the following examples:

Example: Configuring an Ethernet Link Bundle

The following example shows how to join two ports to form an EtherChannel bundle that runs LACP:

```
RP/0/RP0/CPU0:Router(config)# config

RP/0/RP0/CPU0:Router(config-if)# interface Bundle-Ether 3
RP/0/RP0/CPU0:Router(config-if)# ipv4 address 1.2.3.4/24
RP/0/RP0/CPU0:Router(config-if)# bundle minimum-active bandwidth 620000
RP/0/RP0/CPU0:Router(config-if)# bundle minimum-active links 1
RP/0/RP0/CPU0:Router(config-if)# exit
RP/0/RP0/CPU0:Router(config-if)# interface TenGigE 0/3/0/0
RP/0/RP0/CPU0:Router(config-if)# bundle id 3 mode active
RP/0/RP0/CPU0:Router(config-if)# no shutdown
RP/0/RP0/CPU0:Router(config)# exit
RP/0/RP0/CPU0:Router(config-if)# interface TenGigE 0/3/0/1
RP/0/RP0/CPU0:Router(config-if)# bundle id 3 mode active
RP/0/RP0/CPU0:Router(config-if)# no shutdown
RP/0/RP0/CPU0:Router(config-if)# exit
```

This example shows the configuration in the case of a mixed speed bundle:

```
RP/0/RP0/CPU0:Router(config)# config

RP/0/RP0/CPU0:Router(config-if)# interface bundle-ether 50
RP/0/RP0/CPU0:Router(config-if)# root
RP/0/RP0/CPU0:Router(config-if)# interface TenGigE 0/0/0/11
RP/0/RP0/CPU0:Router(config-if)# bundle id 50 mode active
RP/0/RP0/CPU0:Router(config-if)# no shutdown
RP/0/RP0/CPU0:Router(config-if)# interface TenGigE 0/0/0/16
RP/0/RP0/CPU0:Router(config-if)# bundle id 50 mode active
RP/0/RP0/CPU0:Router(config-if)# no shutdown
RP/0/RP0/CPU0:Router(config-if)# interface TenGigE 0/0/0/27
RP/0/RP0/CPU0:Router(config-if)# bundleid 50 mode active
RP/0/RP0/CPU0:Router(config-if)# no shutdown
RP/0/RP0/CPU0:Router(config-if)# interface HundredGigE 0/6/0/1
RP/0/RP0/CPU0:Router(config-if)# bundleid 50 mode active
RP/0/RP0/CPU0:Router(config-if)# no shutdown
RP/0/RP0/CPU0:Router(config-if)# root
RP/0/RP0/CPU0:Router(config)# commit
```

```
RP/0/RP0/CPU0:Router(config)# end
```

The following output is shown for the **show bundle bundle-ether** command:

```
show bundle bundle-ether50
```

```
Bundle-Ether50
 Status:                  Up
 Local links <active/standby/configured>:  4 / 0 / 4
 Local bandwidth <effective/available>:   130000000 (130000000) kbps
 MAC address (source):           0011.2233.4458 (Chassis pool)
 Inter-chassis link:            No
 Minimum active links / bandwidth:     1 / 1 kbps
 Maximum active links:           64
 Wait while timer:             2000 ms
 Load balancing:              Default
 LACP:                    Operational
  Flap suppression timer:          Off
  Cisco extensions:             Disabled
 mLACP:                   Not configured
 IPv4 BFD:                 Not configured

 Port          Device      State    Port ID     B/W, kbps
 --------------------- --------------- ----------- -------------- ----------
 Te0/0/0/11      Local      Active    0x8000, 0x0002  10000000
   Link is Active
 Te0/0/0/16      Local      Active    0x8000, 0x0003  10000000
   Link is Active
 Te0/0/0/27      Local      Active    0x8000, 0x0004  10000000
   Link is Active
 Hu0/6/0/1       Local      Active    0x8000, 0x0001  100000000
   Link is Active
```

In order to view the weight of a mixed speed bundle, run the **show bundle load-balancing** command. The following is the truncated output of this command.

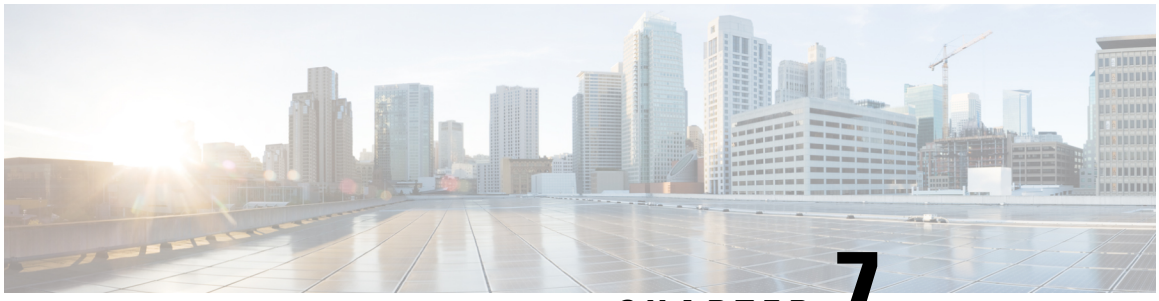
```
show bundle load-balancing bundle-ether50 location 0/0/cpu0

<snip>

Bundle-Ether50
 Type:         Ether (L3)
 Members <current/max>: 4/64
 Total Weighting:    13
 Load balance:     Default
 Locality threshold:  65
 Avoid rebalancing?   False
 Sub-interfaces:    1

 Member Information:
  Port:         LON ULID BW
  --------------------- --- ---- --
  Hu0/6/0/1        0   0 10
  Te0/0/0/11       1   1  1
  Te0/0/0/16       2   2  1
  Te0/0/0/27       3   3  1

 Platform Information:
 =====================

         * Bundle Summary Information *
          --------------------------
```

```
Interface       : Bundle-Ether50    Ifhandle      : 0x00000ce0
Lag ID         : 1           Virtual Port    : 255
Number of Members   : 4           Local to LC    : Yes
Hash Modulo Index   : 13
MGSCP Operational Mode : No

Member Information:
LON    Interface    ifhandle   SFP  port  slot remote/rack_id
-----  ---------------  ----------  ---  ----  ---- --------------
0    Hu0/6/0/1     0x100001c0  648  116   8    0/0
1    Te0/0/0/11    0x04000380   65   9   2    0/0
2    Te0/0/0/16    0x040004c0   67   8   2    0/0
3    Te0/0/0/27    0x04000780   72   4   2    0/0

</snip>
```

Example: Configuring a VLAN Link Bundle

The following example shows how to create and bring up two VLANs on an Ethernet bundle:

```
RP/0/RP0/CPU0:Router(config-subif)# config
RP/0/RP0/CPU0:Router(config-subif)# interface Bundle-Ether 1
RP/0/RP0/CPU0:Router(config-ifsubif)# ipv4 address 1.2.3.4/24
RP/0/RP0/CPU0:Router(config-ifsubif)# bundle minimum-active bandwidth 620000
RP/0/RP0/CPU0:Router(config-if)# bundle minimum-active links
RP/0/RP0/CPU0:Router(config-ifsubif)# exit
RP/0/RP0/CPU0:Router(config-subif)# ip addr 20.2.3.4/24
RP/0/RP0/CPU0:Router(config-subif)# interface Bundle-Ether 1.1
RP/0/RP0/CPU0:Router(config-subif)# encapsulation dot1q 100
RP/0/RP0/CPU0:Router(config-subif) # ip addr 10.2.3.4/24
RP/0/RP0/CPU0:Router(config-subifif)# no shutdown
RP/0/RP0/CPU0:Router(config-subifif)# exit
RP/0/RP0/CPU0:Router(config-if)# interface Bundle-Ether 1.2
RP/0/RP0/CPU0:Router(config-subif)# dot1q vlan 10
RP/0/RP0/CPU0:Router(config-subif)Router # ip addr20.2.3.4/24

RP/0/RP0/CPU0:Router(config-subifif)# no shutdown
RP/0/RP0/CPU0:Router(config-subifif)# exit
RP/0/RP0/CPU0:Router(config)# interface gig 0/1/5/7
RP/0/RP0/CPU0:Router(config-if)# bundle-id 1 mode act
RP/0/RP0/CPU0:Router(config-if)# commit
RP/0/RP0/CPU0:Router(config-if)# exit
```



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).

Feature History for Traffic Mirroring

Release 7.0.14	Support for the following features was introduced in ERSPAN: • Configuration of IP DSCP. • Tunnel IP. • Ability to source ranges of interfaces and SVIs. • Sequence bit is set in the GRE header and the value of sequence number is always 0 for ERSPAN packets. • ERSPAN and Security ACL should be separate. Support for File Mirroring was introduced.
Release 7.0.11	This feature was introduced.

Introduction to Traffic Mirroring

Traffic mirroring, which is sometimes called port mirroring, or Switched Port Analyzer (SPAN) is a Cisco proprietary feature. Traffic mirroring enables you to monitor Layer 3 network traffic passing in, or out of, a set of Ethernet interfaces. You can then pass this traffic to a network analyzer for analysis.

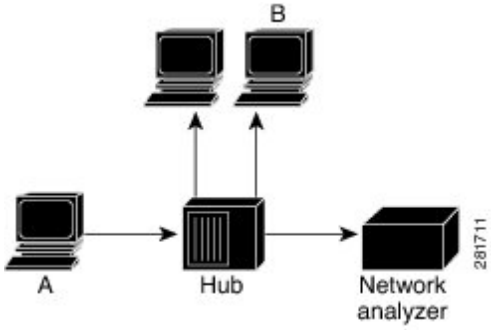
Traffic mirroring copies traffic from one or more Layer 3 interfaces or sub-interfaces. Traffic mirroring then sends the copied traffic to one or more destinations for analysis by a network analyzer or other monitoring device. Traffic mirroring does not affect the switching of traffic on the source interfaces or sub-interfaces. It allows the system to send mirrored traffic to a destination interface or sub-interface.

Traffic mirroring is introduced on switches because of a fundamental difference between switches and hubs. When a hub receives a packet on one port, the hub sends out a copy of that packet from all ports except from the one at which the hub received the packet. In case of switches, after a switch boots, it starts to build up a Layer 2 forwarding table on the basis of the source MAC address of the different packets that the switch receives. After the system builds this forwarding table, the switch forwards traffic that is destined for a MAC address directly to the corresponding port.

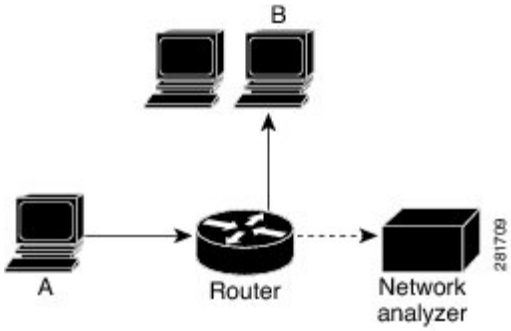
Layer 2 SPAN is not supported on the router.

For example, if you want to capture Ethernet traffic that is sent by host A to host B, and both are connected to a hub, attach a traffic analyzer to this hub. All other ports see the traffic between hosts A and B.

Figure 1: Traffic Mirroring Operation on a Hub



On a switch or router, after the system learns host B MAC address, the system forwards the unicast traffic from A to B to the B port. Therefore, the traffic analyzer does not see this traffic.



In this configuration, the traffic analyzer captures only traffic that is flooded to all ports, such as:

- Broadcast traffic
- Multicast traffic with CGMP or Internet Group Management Protocol (IGMP) snooping disabled
- Unknown unicast traffic on a switch

An extra feature is necessary that artificially copies unicast packets that host A sends. This extra feature is traffic mirroring. When traffic mirroring is enabled, the traffic analyzer is attached to a port that is configured

to receive a copy of every packet that host A sends. This port is called a traffic mirroring port. The other sections of this document describe how you can fine tune this feature.

Implementing Traffic Mirroring on the Cisco 8000 Series Routers

ERSPAN

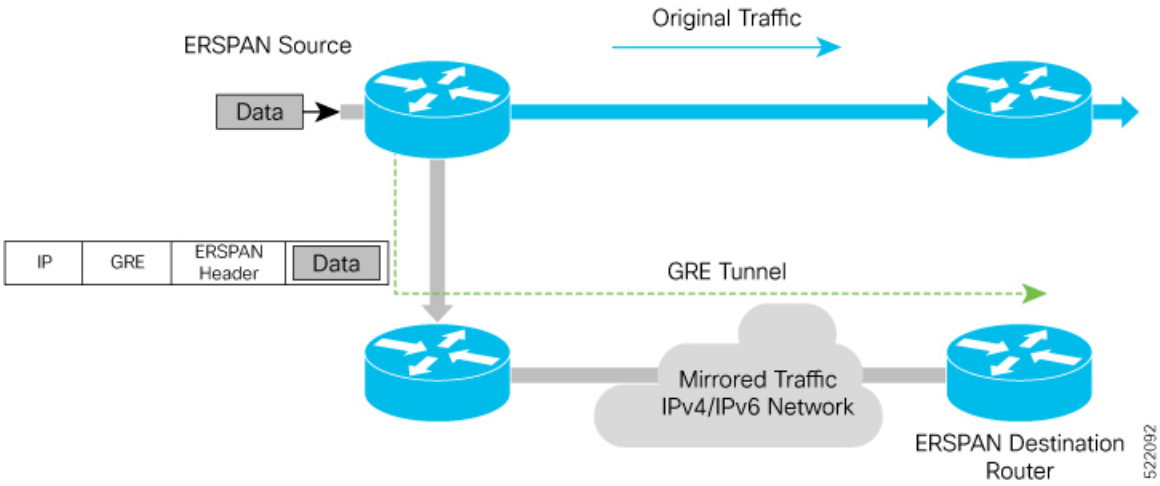
Table 8: Feature History Table

Encapsulated Remote Switched Port Analyzer (ERSPAN) mirrors traffic on one or more source ports and delivers the mirrored traffic to destination port on another switch or management server. ERSPAN enables network operators to troubleshoot issues in the network in real-time using automated tools that auto-configures ERSPAN parameters on the network devices to send specific flows to management servers for in-depth analysis.

ERSPAN transports mirrored traffic over an IP network. The traffic is encapsulated at the source router and is transferred across the network.

Starting with Cisco IOS XR Software Release 7.0.14, sequence bit is set in the GRE header and the value of sequence number is always 0 for ERSPAN packets.

Figure 2: ERSPAN over GRE



Supported Capabilities

The following capabilities are supported:

- The source interfaces are layer 3 interfaces, such as physical, and bundle interfaces or subinterface.
- The routers mirror IPv4 and IPv6 traffic.
- ERSPAN with GRE IPv4 or IPv6 has tunnel destinations.
- ERSPAN supports only RX direction.
- ERSPAN over GRE IPv4 and IPv6 supports SPAN ACL.
- Each monitor session allows only one destination interface.

- ACL permit or deny entries with capture action are part of mirroring features.
- The next hop interface must be a main interface. It can be a Physical or Bundle interface.
- Supports full packet capture.
- In ERSPAN over GRE IPv6, the **HopLimit** and **TrafficClass** fields in outer IPv6 header are editable under the tunnel configuration.

Restrictions

The following are the ERSPAN and SPAN ACL restrictions:

- The router mirrors only unicast traffic.

  However, from Cisco IOS XR Software Release 7.5.3 onwards, the router can mirror multicast traffic.
- Remove and re-apply monitor-sessions on all interfaces after modifying the access control list (ACL).
- GRE tunnel is only dedicated to ERSPAN mirrored packets.
- Only ERSPAN TYPE II header is supported. The value of the index field is always 0. The value of the session-ID field is an internal number that is used by the data path to distinguish between sessions.
- ERSPAN decapsulation is unsupported.
- In Cisco IOS XR Release 7.5.2 and earlier, ERSPAN over GRE IPv6 is supported only when the router does not have any configuration related to MPLS or LDP.

  However, from Cisco IOS XR Software Release 7.5.3 onwards, the ERSPAN will be functional regardless of any configuration related to MPLS or LDP present on the router.
- MPLS packet mirroring is supported only from Cisco IOS XR Software Release 7.5.3 onwards.
- Due to data path limitation, the source IPv6 addresses of the outer IPv6 header of the ERSPAN packet have only higher 64 bits as valid. The lower 64-bits value is changed to zero. The destination GREv6 IPv6 address should contain all the 128 bits.

Restrictions for ERSPAN Packet Truncation support

- From Cisco IOS XR Software Release 7.5.3 onwards, you can truncate the packet size only to a fixed value of 343 bytes.

Traffic Mirroring Terminology

- Ingress traffic—Traffic that enters the switch.
- Egress traffic—Traffic that leaves the switch.
- Source port—A port that the systen monitors 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 traffic mirroring configurations consisting of a single destination and, potentially, many source interfaces.

Characteristics of the Source Port

A source port, also called a monitored port, is a switched or routed port that you monitor for network traffic analysis. In a single local or remote traffic mirroring session, you can monitor source port traffic, such as received (Rx) for ingress traffic. Your router can support any number of source ports (up to a maximum number of 800).

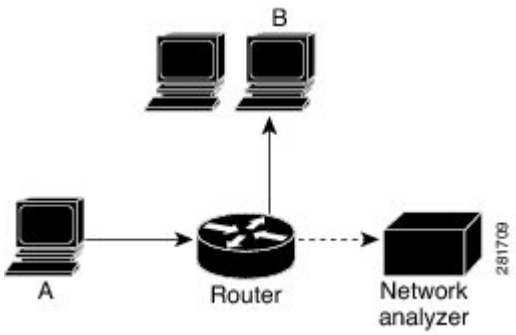
A source port has these characteristics:

- It can be any port type, such as Bundle Interface, sub-interface, 100-Gigabit Ethernet, or 400-Gigabit Ethernet.

Note Bridge group virtual interfaces (BVIs) are not supported.

- Each source port can be monitored in only one traffic mirroring session.
- It cannot be a destination port.
- Each source port can be configured with a direction (ingress) to monitor. For bundles, the monitored direction applies to all physical ports in the group.

Figure 3: Network Analysis on a Router with Traffic Mirroring



In the figure above, the network analyzer is attached to a port that is configured to receive a copy of every packet that host A sends. This port is called a traffic mirroring port.

Characteristics of the 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 is 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 monitor session can have only one destination port.
- A single destination port can belong to only one monitor session.

Note The destination of ERSPAN monitoring session is a GRE IPv4 or IPv6 tunnel.

Supported Traffic Mirroring Types

The system supports the following traffic mirroring types:

- ACL-based traffic mirroring. The system mirrors traffic that is based on the configuration of the global interface ACL.
- Layer 3 traffic mirroring is supported. The system can mirror Layer 3 source ports.

ACL-Based Traffic Mirroring

You can mirror traffic that is based on the definition of a global interface access list (ACL). When you are mirroring Layer 3 traffic, the ACL is configured using the **ipv4 access-list** or **ipv6 access-list** command with the **capture** keyword. The **permit** and **deny** commands determine the behavior of regular traffic. The **capture** keyword designates that the packet is to be mirrored to the destination port.

Starting with Cisco IOS XR Software Release 7.0.14, configuration of ERSPAN and security ACL will be separate. Neither of these will have an impact or dependency on the other, but both can be applied simultaneously.

Restrictions for Traffic Mirroring

The system supports the following forms of traffic mirroring:

- Mirroring traffic to a GRE IPv4 or IPv6 tunnel (also known as Encapsulated Remote Switched Port Analyzer [ER-SPAN] in Cisco IOS Software). The system allows 8 monitor sessions for ERSPAN, 4 monitor sessions for Local SPAN, and 4 monitor sessions for SPAN to File. The total number of monitor sessions for all SPAN features is 8.
- The system does not support traffic mirroring counters per interface.
- The system does not support bundle member interfaces as sources for mirroring sessions.
- ERSPAN tunnel statistics is not supported.

The following general restrictions apply to traffic mirroring using ACLs:

- Configure ACLs on the source interface to avoid default mirroring of traffic. If a Bundle interface is a source interface, configure the ACLs on the bundle interface (not bundle members).

The following restrictions apply to ERSPAN ACL:

- ERSPAN next-hop must have ARP resolved.
  - Any other traffic or protocol triggers ARP.
- ERSPAN decapsulation is not supported.
- ERSPAN does not work if the GRE next hop is reachable over subinterface. For ERSPAN to work, the next hop must be reachable over the main interface.

However, from Cisco IOS XR Software Release 7.5.3 onwards, GRE next hop can be resolved over subinterface or the main interface.

Configuring Traffic Mirroring

These tasks describe how to configure traffic mirroring:

Configuring ACLs for Traffic Mirroring

This section describes the configuration for creating ACLs for traffic mirroring. You must configure the global interface ACLs by using one of the following commands with the **capture** keyword:

- **ipv4 access-list**
- **ipv6 access-list**

Note Starting with Cisco IOS XR Software Release 7.0.14, ACL feature will provide a support of separate ACL configuration for SPAN.

Configuration

- **Security ACL**

  Use the following configuration to configure ACLs for traffic mirroring.

```
/* Create an IPv4 ACL (TM-ACL) for traffic mirroring */
Router(config)# ipv4 access-list TM-ACL
Router(config-ipv4-acl)# 10 permit udp 10.10.10.0 0.0.0.255 eq 10 any capture
Router(config-ipv4-acl)# 20 permit udp 10.10.10.0 0.0.0.255 eq 20 any
Router(config-ipv4-acl)# exit
Router(config)# commit

/* Apply the traffic monitoring to SPAN source interface */
Router(config)# interface HundredGigE0/0/0/12
Router(config-if)# monitor-session mon1 ethernet direction rx-only port-level acl
Router(config-if)# ipv4 access-group TM-ACL ingress
!
```

Use the following configuration as an example to deny data forwarding for an ACE entry, but still mirror the traffic:

```
ipv4 access-list acl1
10 deny ipv4 any 2.1.0.0/16 capture
20 permit ipv4 any any
!
```

If `acl1` is attached to the interface as shown below:

```
RP/0/RP0/CPU0(config-if)# ipv4 access-group acl1 ingress
```

Data Traffic to 2.1.0.0/16 is dropped. Mirroring happens only if `icmp-off` keyword is added to the ACE as shown below. If this keyword is not added, mirroring does not take place. Furthermore, the `icmp-off` workaround is applicable only to security ACL.

```
ipv4 access-list acl1
10 deny ipv4 any 2.1.0.0/16 capture icmp-off
20 permit ipv4 any any
!
```

- **SPAN ACL**
  - SPAN ACL does not support User Defined Fields (UDF).
  - Deny action in SPAN ACL is ignored, and no packet drops from SPAN ACL. Deny ACEs will be internally converted to permit ACEs. Packets will also be mirrored.
  - There is no implicit deny-all entry in SPAN ACL.
  - IPV6 ACL is required for mirroring IPV6 packet, if IPV4 ACL is configured, and vice versa. This follows the same structure as Security ACL with IPv4 and IPv6 mirror options.

The maximum scale for SPAN ACL ID for fixed and centralized chassis is 3/slice pair and 9/NP. For distributed chassis, the maximum scale for SPAN ACL ID is 6/NP.

Use the following configuration to enable traffic mirroring with ACLs.

```
/* Create a SPAN IPv4 ACL (v4-monitor-acl) for traffic mirroring */
Router(config)# ipv4 access-list v4-monitor-acl
Router(config-ipv4-acl)# 10 permit udp 20.1.1.0 0.0.0.255 eq 10 any
Router(config-ipv4-acl)# 20 permit udp 30.1.1.0 0.0.0.255 eq 20 any
Router(config-ipv4-acl)# exit
Router(config)# commit

/*Create a SPAN IPv6 ACL (v6-monitor-acl) for traffic mirroring */
Router(config)# ipv6 access-list v6-monitor-acl
Router(config-ipv6-acl)# 10 permit ipv6 host 120:1:1::1 host 130:1:1::1
Router(config-ipv6-acl)# exit

/* Apply the traffic monitoring to SPAN source interface */
Router(config)# interface HundredGigE0/0/0/12
Router(config-if)# monitor-session mon1 ethernet direction rx-only
Router(config-if)# acl ipv4 v4-monitor-acl
Router(config-if)# acl ipv4 v6-monitor-acl!
```

Note The `capture` keyword which is required for Security ACL for SPAN to work, is optional for SPAN ACL.

Use the `show access-lists [ipv4 | ipv6] acl-name hardware ingress span [detail | interface | location | sequence | verify] location x` command to display ACL information:

```
Router# show access-lists ipv4 v4span1 hardware ingress span interface bundle-Ether 100
location 0/3/cpu0
ipv4 access-list v4span1
10 permit ipv4 host 51.0.0.0 host 101.0.0.0
20 permit ipv4 host 51.0.0.1 host 101.0.0.1
30 permit ipv4 host 51.0.0.2 any
40 permit ipv4 any host 101.0.0.3
50 permit ipv4 51.0.1.0 0.0.0.255 101.0.1.0 0.0.0.255
60 permit ipv4 51.0.2.0 0.0.0.255 101.0.2.0 0.0.0.255 precedence critical
```

Troubleshooting ACL-Based Traffic Mirroring

Take note of these configuration issues:

- Even when the system configures the **acl** command on the source mirroring port, if the ACL configuration command does not use the **capture** keyword, the system does not mirror traffic.
- If the ACL configuration uses the **capture** keyword, but you have not configured the **acl** command on the source port, the system mirrors the traffic, but does not apply access list configuration.

This example shows both the **capture** keyword in the ACL definition and the **acl** command that is configured on the interface:

```
/* Create an IPv4 ACL (TM-ACL) for traffic mirroring */
Router(config)# ipv4 access-list TM-ACL
Router(config-ipv4-acl)# 10 permit udp 10.1.1.0 0.0.0.255 eq 10 any capture
Router(config-ipv4-acl)# 20 permit udp 10.1.1.0 0.0.0.255 eq 20 any

Apply the traffic monitoring to interface
Router(config)#interface HundredGigE0/0/0/12
Router(config-if)# monitor-session mon1 ethernet direction rx-only port-only acl
Router(config-if)# ipv4 access-group TM-ACL ingress
```

Flexible CLI for ERSPAN

Starting with Cisco IOS XR Software Release 7.0.14, ERSPAN can be configured using flexible CLI. This CLI is a single configuration object containing all the properties of an ERSPAN session, tunnel properties, and the list of source interfaces, which can be easily removed and re-added. Flexible CLI minimises risk of user error and promotes operational simplicity.

Configure a flexible CLI group in ERSPAN containing:

- Global ERSPAN session configuration
- Tunnel interface configuration
- ERSPAN source attachment configuration, applied to a regexp of interface names

Note The flexible CLI group contains only the session and interface properties. The session and interface objects themselves must be created in the configuration as usual.

The following example shows a global flexible CLI configuration:

```
group erspan-group-foo
  monitor-session 'foo' ethernet   /* Global configuration */
    destination interface tunnel-ip0
  !
  interface 'tunnel-ip0'   /* Tunnel interface configuration */
    tunnel tos 10
    tunnel mode gre ipv4
    tunnel source 10.10.10.1
    tunnel destination 20.20.20.2
  !
  interface 'GigabitEthernet0/0/0/[0-3]'   /* Interface configuration */
    monitor-session foo ethernet
```

```
  !
end-group
```

To enable all ERSPAN configurations, execute `apply-group erspan-group-foo` command. To disable ERSPAN configuration, delete this command.

Note The following three keywords are regular expressions and must be quoted:

- Definition of session name (example: `foo`)
- Definition of tunnel name (example: `tunnel-ip0`)
- Set of source interface names (example: `GigabitEthernet0/0/0/[0-3]`)

Use the `show running-config inheritance` command to view the final configuration after the group is expanded, and the `show monitor-session status` to check the operational state of ERSPAN session.

Attaching the Configurable Source Interface

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **interface** *type number*

Example:

```
RP/0/RP0/CPU0(config)# interface HundredGigE 0/1/0/10/0/1/0
```

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(config-if)# ipv4 access-group acl1 ingress
```

Controls access to an interface.

Step 4 **monitor-session** *session-name* **ethernet direction rx-onlyport-level**

Example:

```
RP/0/RP0/CPU0(config-if)# monitor-session mon1 ethernet direction rx-only port-level acl
RP/0/RP0/CPU0(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(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 overrides any ACL that may be configured on the interface.

Step 6 **exit**

Example:

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

Exits monitor session configuration mode and returns to interface configuration mode.

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

Example:

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

or

```
RP/0/RP0/CPU0(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# show monitor-session status
```

Displays information about the monitor session.

Introduction to ERSPAN Rate Limit

With ERSPAN rate limit feature, you can monitor traffic flow through any IP network. This includes third-party switches and routers.

ERSPAN operates in the following modes:

- ERSPAN Source Session – box where the traffic originates (is SPANned).
- ERSPAN Termination Session or Destination Session – box where the traffic is analyzed.

This feature provides rate limiting of the mirroring traffic. With rate limiting, you can limit the amount of traffic to a specific rate, which prevents the network and remote ERSPAN destination traffic overloading. Be informed, if the rate-limit exceeds then the system may cap or drop the monitored traffic.

You can configure the QoS parameters on the traffic monitor session.

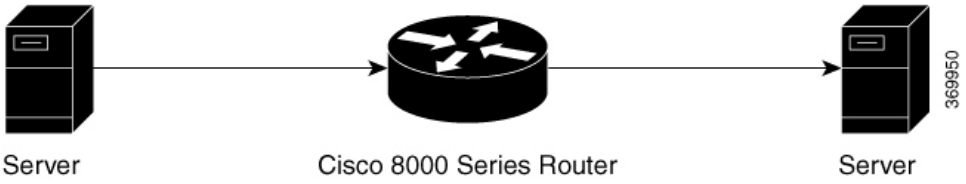
- Traffic Class (0 through 7)
  - Traffic class 0 has the lowest priority and 7 the highest.
  - The default traffic class is the same as that of the original traffic class.

Benefits

With ERSPAN rate limit feature, you can limit the mirrored traffic and use the mirrored traffic for data analysis.

Topology

Figure 4: Topology for ERSPAN Rate Limit



The encapsulated packet for ERSPAN is in ARPA/IP format with GRE encapsulation. The system sends the GRE tunneled packet to the destination box identified by an IP address. At the destination box, SPAN-ASIC decodes this packet and sends out the packets through a port. ERSPAN rate limit feature is applied on the router interface to rate limit the monitored traffic.

The intermediate switches carrying ERSPAN traffic from source session to termination session can belong to any L3 network.

Configure ERSPAN Rate Limit

Use the following steps to configure ERSPAN rate limit:

```
monitor-session ERSPAN ethernet
destination interface tunnel-ip1
!
```

```
RP/0/RP0/CPU0:pyke-008#sh run int tunnel-ip 1

interface tunnel-ip1
ipv4 address 4.4.4.1 255.255.255.0
tunnel mode gre ipv4
tunnel source 20.1.1.1
tunnel destination 20.1.1.2
!

RP/0/RP0/CPU0:pyke-008#sh run int hundredGigE 0/0/0/16

interface HundredGigE0/0/0/16
ipv4 address 215.1.1.1 255.255.255.0
ipv6 address 3001::2/64
monitor-session ERSPAN ethernet direction rx-only port-level
  acl
!
ipv4 access-group ACL6 ingress
```

Running Configuration

```
!!A traffic class needs to be configured under the monitor session.
monitor-session mon2 ethernet
destination interface tunnel-ip30
traffic class 5

A shaper needs to be configured for this traffic class:

policy-map m8
class TC1
  bandwidth percent 11
 !
 class TC2
  bandwidth percent 12
 !
 class TC3
  bandwidth percent 13
 !
 class TC4
  bandwidth percent 14
 !
 class TC5
  shape average percent 15
 !
 class TC6
  bandwidth percent 16
 !
 class TC7
  bandwidth percent 17

This policy-map has to be installed on the interface over which the mirrored traffic is
sent in the egress direction:
interface TenGigE0/6/0/9/0
service-policy output m8
```

Verification

```
RP/0/RP0/CPU0:ios#show monitor-session FOO status detail
Wed May 2 15:14:05.762 UTC
Monitor-session FOO
Destination interface tunnel-ip100
Source Interfaces
-----------------
TenGigE0/6/0/4/0
```

```
Direction: Both
Port level: True
ACL match: Disabled
```

Introduction to File Mirroring

Prior to Cisco IOS XR Software Release 7.0.14, the router did not support file mirroring from active RP to standby RP. Administrators had to manually perform the task or use EEM scripts to sync files across active RP and standby RP. Starting with Cisco IOS XR Software Release 7.0.14, 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.

Two new CLIs have been introduced for the file mirroring feature:

- **`mirror enable`**

  The `/harddisk:/mirror` directory is created by default, but file mirroring functionality is only enabled by executing the `mirror enable` command from configuration terminal. Status of the mirrored files can be viewed with `show mirror status` command.

- **`mirror enable checksum`**

  The `mirror enable checksum` command enables MD5 checksum across active to standby RP to check integrity of the files. This command is optional.

Limitations

The following limitations apply to file mirroring:

- Supported only on Dual RP systems.
- Supports syncing only from active to standby RP. If files are copied into standby `/harddisk:/mirror` location, it won't be synced to active RP.
- A slight delay is observed in `show mirror` command output when mirror checksum configuration is enabled.
- Not supported on multichassis systems.

Configure File Mirroring

File mirroring has to be enabled explicitly on the router. It is not enabled by default.

```
RP/0/RSP0/CPU0:router#show run mirror

Thu Jun 25 10:12:17.303 UTC
mirror enable
mirror checksum
```

Following is an example of copying running configuration to `harddisk:/mirror` location:

```
RP/0/RSP0/CPU0:router#copy running-config harddisk:/mirror/run_config
Wed Jul  8 10:25:51.064 PDT
Destination file name (control-c to abort): [/mirror/run_config]?
Building configuration..
```

```
32691 lines built in 2 seconds (16345)lines/sec
[OK]
```

Verification

To verify the syncing of file copied to mirror directory, use the `show mirror` command.

```
RP/0/RSP0/CPU0:router#show mirror
Wed Jul  8 10:31:21.644 PDT
% Mirror rsync is using checksum, this show command may take several minutes if you have
many files. Use Ctrl+C to abort
MIRROR DIR: /harddisk:/mirror/
% Last sync of this dir ended at Wed Jul  8 10:31:11 2020
Location   |Mirrored |MD5 Checksum                     |Modification Time
-------------------------------------------------------------------------
run_config |yes      |76fc1b906bec4fe08ecda0c93f6c7815 |Wed Jul  8 10:25:56 2020
```

If checksum is disabled, `show mirror` command displays the following output:

```
RP/0/RSP0/CPU0:router#show mirror
Wed Jul 8 10:39:09.646 PDT
MIRROR DIR: /harddisk:/mirror/
% Last sync of this dir ended at Wed Jul  8 10:31:11 2020
Location   |Mirrored |Modification Time
----------------------------------------
run_config |yes         |Wed Jul  8 10:25:56 2020
```

If there is a mismatch during the syncing process, use `show mirror mismatch` command to verify.

```
RP/0/RP0/CPU0:router# show mirror mismatch
Wed Jul  8 10:31:21.644 PDT
 MIRROR DIR: /harddisk:/mirror/
 % Last sync of this dir ended at Wed Jul  8 10:31:11 2020
 Location  |Mismatch Reason     |Action Needed
-----------------------------------------------
 test.txt  |newly created item. |send to standby
```

Monitor Multiple ERSPAN Sessions with SPAN and Security ACL

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
Monitor Multiple ERSPAN Sessions with SPAN and Security ACL	Release 7.5.4	

Starting Cisco IOS XR Software Release 7.5.4 you can monitor multiple ERSPAN sessions using GREv4 and GREv6 under the same source interface. Multiple ERSPAN monitor sessions configured on an interface allow you to choose the destination interface for the mirrored traffic. For the configuration of monitor sessions, you can use SPAN and security ACLs together. The SPAN and security ACLs are applicable only in the ingress traffic.

Configure Multiple Monitor ERSPAN Sessions with SPAN and Security ACL

This example shows how to configure SPAN and Security ACL for SPAN with GREv4 and GREv6 Monitor Sessions.

Running configuration

```
Router(config)#show running-config interface
monitor-session always-on-v4 ethernet direction rx-only port-level
  acl ipv4 v4-monitor-acl2
  acl ipv6 v6-monitor-acl2
!
monitor-session on-demand-v4 ethernet direction rx-only port-level
  acl ipv4 v4-monitor-acl2
  acl ipv6 v6-monitor-acl2
!
ipv4 access-group sec_aclv4 ingress
ipv6 access-group sec_aclv6 ingress
!
!
```

Traffic Mirroring Configuration Examples

This section contains examples of how to configure traffic mirroring:

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 HundredGigE0/5/0/38
================================================================================
Source Interface   Dir  Status
---------------------- ---- -------------------------------------------------
Gi0/5/0/4         Rx Operational
Gi0/5/0/17        Rx Operational

RP/0/RP0/CPU0:router# show monitor-session status detail

Monitor-session sess1
 Destination interface is not configured
 Source Interfaces
 -----------------
 HundredGigE0/0/0/0
  Direction: Rx
  ACL match: Enabled
  Portion:  Full packet
  Status:  Not operational (destination interface not known).
 HundredGigE0/0/0/2
  Direction: Rx
  ACL match: Disabled
  Portion:  First 100 bytes

RP/0/RP0/CPU0:router# show monitor-session status error
```

```
Monitor-session ms1
Destination interface HundredGigE0/2/0/15 is not configured
================================================================================
Source Interface   Dir  Status
-------------------- ---- ------------------------------------------------------

Monitor-session ms2
Destination interface is not configured
================================================================================
Source Interface   Dir  Status
-------------------- ---- ------------------------------------------------------
```

Monitor Session Statistics: Example

The monitor session statistics is provided in the form of packets and bytes. Use the following command to get the status:

Note

- Currently, the system does not allow you to clear these counters.
- The counters are present on the line-card that contains the interface over which the mirrored packets are sent to the ERSPAN session destination.

  If required, to clear the counters, delete and recreate the monitor session. Also, clear the counters by performing a Shut/No Shut of the tunnel interface, which triggers a Delete+Create action.

Layer 3 ACL-Based Traffic Mirroring: Example

This example shows how to configure Layer 3 ACL-based traffic mirroring:

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:

```
Monitor-session sess1
<Session status>
================================================================================
Source Interface   Dir  Status
-------------------- ---- ------------------------------------------------------
Gi0/0/0/0       Both <Source interface status>
Gi0/0/0/2       Both <Source interface status>
```

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. Check **show run** command output to ensure that a session with a correct name has been configured.
Destination interface <intf> is not configured	The interface that has been configured as the destination does not exist. For example, the destination interface may be configured to be a VLAN subinterface, but the VLAN subinterface may not have been yet created.
Destination interface <intf> (<down-state>)	The destination interface is not in Up state in the Interface Manager. You can verify the state using the **show interfaces** command. Check the configuration to see what might be keeping the interface from coming up (for example, a sub-interface needs to have an appropriate encapsulation configured).

The <Source interface status> can report these messages:

Source Interface Status	Explanation
Operational	Everything appears to be working correctly in traffic mirroring PI. Please follow up with the platform teams in the first instance, if mirroring is not operating as expected.
Not operational (Session is not configured globally)	The session does not exist in global configuration. Check the **show run** command output to ensure that a session with the right name has been configured.
Not operational (destination interface not known)	The session exists, but it either does not have a destination interface specified, or the destination interface named for the session does not exist (for example, 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 interface, so traffic mirroring does not work.
Not operational (destination not active)	The destination interface or pseudowire is not in the Up state. See the corresponding *Session status* error messages for suggested resolution.
Not operational (source state <down-state>)	The source interface is not in the Up state. You can verify the state using the **show interfaces** command. Check the configuration to see what might be keeping the interface from coming up (for example, a sub-interface needs to have an appropriate encapsulation configured).

Source Interface Status	Explanation
Error: see detailed output for explanation	Traffic mirroring has encountered an error. Run the **show monitor-session status detail** command to display more information.

The **show monitor-session status detail** command displays full details of the configuration parameters, and of any errors encountered. For example:

```
RP/0/RP0/CPU: router#show monitor-session status detail

Monitor-session sess1
 Destination interface is not configured
 Source Interfaces
 -----------------
 HundredGigE0/0/0/0
  Direction: Both
  ACL match: Enabled
  Portion:  Full packet
  Status:  Not operational (destination interface not known)
 HundredGigE0/0/0/2
  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 HundredGigE 0/0/0/0
 Source Interfaces
 -----------------
 HundredGigE 0/1/0/0.100:
  Direction: Both
  Status:  Operating
 HundredGigE 0/2/0/0.200:
  Direction: Tx
  Status:  Error: <blah>

Monitor session bar
 No destination configured
 Source Interfaces
 -----------------
 HundredGigE 0/3/0/0.100:
  Direction: Rx
  Status:  Not operational(no destination)
```

Additional Debugging Commands

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 platform ?

 all   Turn on all the debugs
 errors VKG SPAN EA errors
 event  VKG SPAN EA event
 info  VKG SPAN EA info

RP/0/RP0/CPU0:router# debug monitor-session platform all

RP/0/RP0/CPU0:router# debug monitor-session platform event

RP/0/RP0/CPU0:router# debug monitor-session platform info

RP/0/RP0/CPU0:router# show monitor-session status ?

 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
```

If there is no route to the destination IPv4 address, the status displayed for the monitor session looks like this:

```
RP/0/RP0/CPU0:Router1#show monitor-session mon2 status internal
Wed Oct  9 19:24:06.084 UTC
Information from SPAN Manager and MA on all nodes:
Monitor-session mon2 (ID 0x00000001) (Ethernet)
SPAN Mgr: Destination interface tunnel-ip2 (0x0f000034) (down)
          Last error: Success
          Tunnel data:
            Mode: GREoIPv4
            Source IP: 2.2.2.2
            Dest IP: 130.10.10.2
            VRF:
            ToS: 0 (copied)
            TTL: 255
            DFbit: Not set
0/1/CPU0: Destination interface  is not configured
          Tunnel data:
            Mode: GREoIPv4
            Source IP: 2.2.2.2
            Dest IP: 130.10.10.2
            VRF:
            ToS: 0 (copied)
            TTL: 255
            DFbit: Not set
```

To verify if there is a route to the destination IPv4 address, use the following command:

```
RP/0/RP0/CPU0:Router1#show cef ipv4 130.10.10.2
Wed Oct  9 19:25:12.282 UTC
0.0.0.0/0, version 0, proxy default, default route handler, drop adjacency, internal 0x1001011
 0x0 (ptr 0x8e88d2b8) [1], 0x0 (0x8ea4d0a8), 0x0 (0x0)
Updated Oct  9 19:03:36.068
Prefix Len 0, traffic index 0, precedence n/a, priority 15
   via 0.0.0.0/32, 3 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8e2db240 0x0]
    next hop 0.0.0.0/32
     drop adjacency
```

When a route is present, the command used in the previous example displays the following:

```
RP/0/RP0/CPU0:Router1#show cef ipv4 130.10.10.2
Wed Oct  9 19:26:06.141 UTC
130.1.1.0/24, version 20, internal 0x1000001 0x0 (ptr 0x8e88aa18) [1], 0x0 (0x8ea4dc68),
0x0 (0x0)
Updated Oct  9 19:26:02.139
Prefix Len 24, traffic index 0, precedence n/a, priority 3
   via 131.1.1.1/32, HundredGigE0/1/0/2, 2 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8f8e2260 0x0]
    next hop 131.10.10.1/32
    local adjacency
```

The show monitor command displays the following:

```
show monitor-session mon2 status internal
Wed Oct  9 19:26:12.405 UTC
Information from SPAN Manager and MA on all nodes:
Monitor-session mon2 (ID 0x00000001) (Ethernet)
SPAN Mgr: Destination interface tunnel-ip2 (0x0f000034)
          Last error: Success
          Tunnel data:
            Mode: GREoIPv4
            Source IP: 2.2.2.2
            Dest IP: 130.10.10.2
            VRF:
            ToS: 0 (copied)
            TTL: 255
            DFbit: Not set
0/1/CPU0: Destination interface tunnel-ip2 (0x0f000034)
          Tunnel data:
            Mode: GREoIPv4
            Source IP: 2.2.2.2
            Dest IP: 130.10.10.2
            VRF:
            ToS: 0 (copied)
            TTL: 255
            DFbit: Not set

Information from SPAN EA on all nodes:
Monitor-session 0x00000001 (Ethernet)
0/1/CPU0: Name 'mon2', destination interface tunnel-ip2 (0x0f000034)
Platform, 0/1/CPU0:

  Monitor Session ID: 1

  Monitor Session Packets: 0
  Monitor Session Bytes: 0

0/2/CPU0: Name 'mon2', destination interface tunnel-ip2 (0x0f000034)
Platform, 0/2/CPU0:

  Monitor Session ID: 1

  Monitor Session Packets: 0
  Monitor Session Bytes: 0

Missing ARP to the next hop to the destination
This condition is detected via this show command:
show monitor-session mon2 status internal
```

After resolving ARP for the next hop, which is done by invoking a ping command to the destination, the show command output displays the following:

```
RP/0/RP0/CPU0:Router1#show monitor-session mon2 status internal
Wed Oct  9 19:32:24.856 UTC
Information from SPAN Manager and MA on all nodes:
Monitor-session mon2 (ID 0x00000001) (Ethernet)
SPAN Mgr: Destination interface tunnel-ip2 (0x0f000034)
          Last error: Success
          Tunnel data:
            Mode: GREoIPv4
            Source IP: 2.2.2.2
            Dest IP: 130.10.10.2
            VRF:
            ToS: 0 (copied)
            TTL: 255
            DFbit: Not set
0/1/CPU0: Destination interface tunnel-ip2 (0x0f000034)
          Tunnel data:
            Mode: GREoIPv4
            Source IP: 2.2.2.2
            Dest IP: 130.10.10.2
            VRF:
            ToS: 0 (copied)
            TTL: 255
            DFbit: Not set

Information from SPAN EA on all nodes:
Monitor-session 0x00000001 (Ethernet)
0/1/CPU0: Name 'mon2', destination interface tunnel-ip2 (0x0f000034)
Platform, 0/1/CPU0:

  Monitor Session ID: 1
    Monitor Session Packets: 0
  Monitor Session Bytes: 0

0/2/CPU0: Name 'mon2', destination interface tunnel-ip2 (0x0f000034)
Platform, 0/2/CPU0:

  Monitor Session ID: 1
    Monitor Session Packets: 0
  Monitor Session Bytes: 0
```

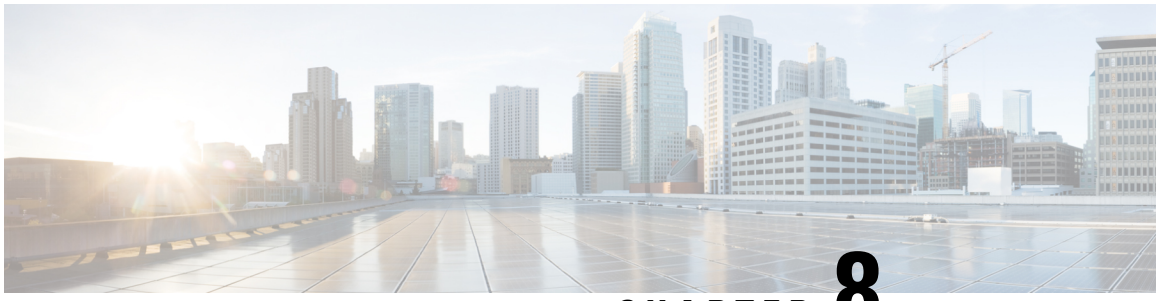
Where to Go Next

When you have configured an Ethernet interface, you can configure individual VLAN subinterfaces on that Ethernet interface.

For information about modifying Ethernet management interfaces for the shelf controller (SC), route processor (RP), and distributed RP, see the Advanced Configuration and Modification of the Management Ethernet Interface later in this document.

For information about IPv6 see the Implementing Access Lists and Prefix Lists on

Cisco IOS XR Software module in the Cisco IOS XR IP Addresses and Services Configuration Guide.



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 (RP). The configuration and control plane are mirrored onto the standby RP and, in the event of a failover, the virtual interfaces move to the ex-standby, which then becomes the newly active RP.

Feature History for Configuring Loopback and Null Interfaces on Cisco IOS XR Software

Release	Modification
Release 7.0.11	This feature was introduced.

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 that you need for each command. If you suspect a user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

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 or active. Any packet that the system transmits 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.

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 that is 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 nonbroadcast 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, the system enables the **ipv4 unreachables** command. If we do not want ICMP to send protocol unreachable, then you need to configure using the **ipv4 icmp unreachable disable** command.

By default, the system creates the Null 0 interface during boot process and you cannot remove it. You can configure the **ipv4 unreachables** command for this interface, but most configuration is unnecessary because this interface just discards all the packets thats the system sends.

Use the **show interfaces null0** command to display the Null 0 interface.

Virtual Management Interface Overview

Configuring an IPv4 virtual address enables you to access the router from a single virtual address with a management network without prior knowledge of which RP is active. An IPv4 virtual address persists across route switch processor (RP) failover situations. For this to happen, the virtual IPv4 address must share a common IPv4 subnet with a management Ethernet interface on both the RPs.

On a router where each RP has multiple management Ethernet interfaces, the virtual IPv4 address maps to the management Ethernet interface on the active RP that shares the same IP subnet.

Active and Standby RPs and Virtual Interface Configuration

The standby RP is available and in a state in which it can take over the work from the active RPs should that prove necessary. Conditions that necessitate the standby RP to become the active RP and assume the active RP's duties include:

- Failure detection by a watchdog
- Administrative command to take over
- Removal of the active RP from the chassis

If a second RP is not present in the chassis while the first is in operation, a second RP may be inserted and automatically becomes the standby RP. The standby RP may also be removed from the chassis with no effect on the system other than loss of RP redundancy.

After failover, the virtual interfaces all are present on the standby (now active) RP. Their state and configuration are unchanged and there has been no loss of forwarding (in the case of tunnels) over the interfaces during the failover. The routers use nonstop forwarding (NSF) over bundles and tunnels through the failover of the host RP.

Note The user need not configure anything to guarantee that the standby interface configurations are maintained.

Protocol configuration such as tacacs source-interface, snmp-server trap-source, ntp source, logging source-interface do not use the virtual management IP address as their source by default. Use the **ipv4 virtual address use-as-src-addr** command to ensure that the protocol uses the virtual IPv4 address as its source address. Alternatively, you can also configure a loopback address with the designated or desired IPv4 address and set that as the source for protocols such as TACACS+ using the **tacacs source-interface** command.

How to Configure Virtual Interfaces

This section contains the following procedures:

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.
- That IP address must not be used by another interface on the router.
- The IP address must not be used by an interface on any other router on the network.

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

/* Enters interface configuration mode and names the new loopback interface */

```
RP/0/RP0/CPU0:router#(config)# interface Loopback 3
```

/* Assigns an IP address and subnet mask to the virtual loopback interface */

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 172.18.189.38/32

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

- 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.

/* Display the configuration of the loopback interface */

```
RP/0/RP0/CPU0:router# show interfaces Loopback 3
```

Configuring Null Interfaces

This task explains how to configure a basic null interface.

/* Enters global configuration mode. */

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

/* Enter the null 0 interface configuration mode. */

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

/* Save configuration changes. */

```
RP/0/RP0/CPU0:router(config-null0)# end
```

/* Verif the configuration of the null interface. */

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

Configuring Virtual IPv4 Interfaces

This task explains how to configure an IPv4 virtual interface.

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

/* Define an IPv4 virtual address for the management Ethernet interface. */

```
RP/0/RSP0/CPU0:router(config)# ipv4 virtual address 10.3.32.154/8
RP/0/RSP0/CPU0:router(config-null0)# end
or
RP/0/RSP0/CPU0:router(config-null0)# commit
```

- 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.

This is an example for configuring a virtual IPv4 interface:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# ipv4 virtual address 10.3.32.154/8
RP/0/RSP0/CPU0:router(config-null0)# commit
```

Configuration Examples for Virtual Interfaces

This section provides the following configuration examples:

Configuring a Loopback Interface: Example

The following example indicates how to configure a loopback interface:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface Loopback 3
RP/0/RP0/CPU0:router(config-if)# ipv4 address 172.18.189.38/32
RP/0/RP0/CPU0:router(config-if)# end
Uncommitted changes found, commit them? [yes]: yes
RP/0/RP0/CPU0:router# show interfaces Loopback 3

Loopback3 is up, line protocol is up
 Hardware is Loopback interface(s)
 Internet address is 172.18.189.38/32
 MTU 1514 bytes, BW Unknown
   reliability 0/255, txload Unknown, rxload Unknown
 Encapsulation Loopback, loopback not set
 Last clearing of "show interface" counters never
 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
```

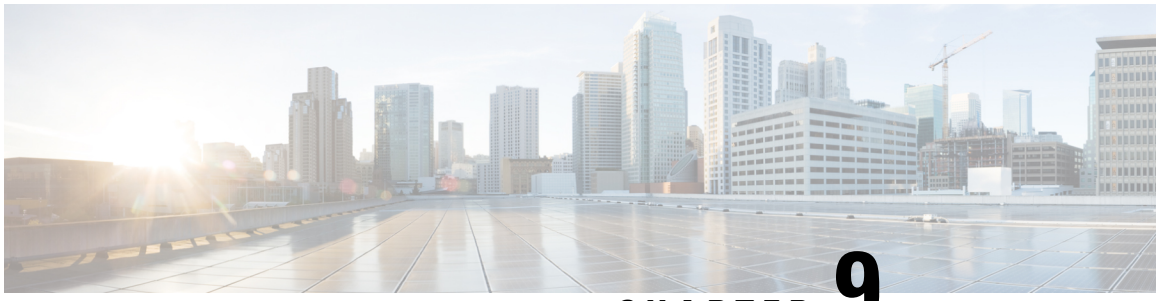
Configuring a Null Interface: Example

The following example indicates 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 unreachables
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
 Hardware is Null interface
 Internet address is Unknown
 MTU 1500 bytes, BW Unknown
   reliability 0/255, txload Unknown, rxload Unknown
 Encapsulation Null, loopback not set
 Last clearing of "show interface" counters never
 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 a Virtual IPv4 Interface: Example

```
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

This module describes the configuration and management of 802.1Q VLAN interfaces.

The IEEE 802.1Q specification establishes a standard method for tagging Ethernet frames with VLAN membership information. It defines the operation of VLAN bridges that permit the definition, operation, and administration of VLAN topologies within a bridged LAN infrastructure.

The 802.1Q standard is intended to address the problem of how to divide large networks into smaller parts so broadcast and multicast traffic does not use more bandwidth than necessary. The standard also helps provide a higher level of security between segments of internal networks.

Feature History for Configuring 802.1Q VLAN Interfaces

Release	Modification
Release 7.0.11	This feature was introduced.

Prerequisites for Configuring 802.1Q VLAN 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.

Before configuring 802.1Q VLAN interfaces, ensure that you meet the following conditions:

- You must have configured a HundredGigE interface, a FourHundredGigE interface, or an Ethernet bundle interface.

Information About Configuring 802.1Q VLAN Interfaces

To configure 802.1Q VLAN interfaces, you must understand the following concepts:

802.1Q VLAN Overview

A VLAN is a group of devices on one or more LANs that you can configure so that the devices can communicate as if they were attached to the same wire. When in fact, they are located on several different LAN segments. Because VLANs are based on logical instead of physical connections, they are flexible for user and host management, bandwidth allocation, and resource optimization.

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.

The 802.1Q specification establishes a standard method for inserting VLAN membership information into Ethernet frames.

Cisco IOS XR software supports VLAN subinterface configuration on 40Gigabit, HundredGig, FourHundredGig, and bundle interfaces.

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 VLANs can be created statically by manual entry or dynamically through Generic Attribute Registration Protocol (GARP) VLAN Registration Protocol (GVRP). 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.

Subinterfaces

Subinterfaces are logical interfaces that you can create on a hardware interface. These software-defined interfaces allow the segregation of traffic into separate logical channels on a single hardware interface. It also allows for the better utilization of the available bandwidth on the physical interface.

You can distinguish subinterfaces from each other by adding an extension at the end of the interface name and designation. For instance, the system indicates Ethernet subinterface 23 on the physical interface designated TenGigE 0/1/0/0, by TenGigE 0/1/0/0.23.

Before the system allows a subinterface 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, you must explicitly define the VLAN identifier.

Subinterface MTU

The system inherits the subinterface maximum transmission unit (MTU)from the physical interface with an additional four bytes allowed for the 802.1Q VLAN tag.

Native VLAN

The router does not support a native VLAN. However, the equivalent functionality is accomplished using an **encapsulation** command as follows:

```
encapsulation dot1q TAG-ID
```

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.

Tip You can programmatically configure and retrieve the VLAN interfaces and subinterfaces parameters using `openconfig-vlan.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

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

/* Enter subinterface configuration mode and specifies the interface type, location, and subinterface number. */

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/2/0/4.10
```

- 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 *interface-path-id.subinterface*, and a period between arguments is required as part of the notation.

/* Set the Layer 2 encapsulation of an interface. */

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

Note
- The **dot1q vlan** command is replaced by the **encapsulation dot1q** command on the Cisco 8000 Series Router. It is still available for backward-compatibility, but only for Layer 3 interfaces.

/* Assign an IP address and subnet mask to the subinterface. */

```
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 178.18.169.23/24
```

- 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.

/* The **exit** command is not explicitly required. */

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

- 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

```
"RP/0/RP0/CPU0:S3(config)#interface fourHundredGigE 0/5/0/1.100
RP/0/RP0/CPU0:S3(config-subif)#ipv4 address 100.100.100.100/31
RP/0/RP0/CPU0:S3(config-subif)#encapsulation dot1q 100
RP/0/RP0/CPU0:S3(config-subif)#no shutdown
RP/0/RP0/CPU0:S3(config-subif)#commit
Mon Jul  8 23:05:01.979 PDT
RP/0/RP0/CPU0:S3(config-subif)#end
RP/0/RP0/CPU0:S3#show interfaces fourHundredGigE 0/5/0/1.100 brief
Mon Jul  8 23:05:08.784 PDT

               Intf       Intf        LineP              Encap  MTU        BW
               Name       State       State               Type (byte)    (Kbps)
--------------------------------------------------------------------------------
      FH0/5/0/1.100        up        up             802.1Q  1518  400000000

RP/0/RP0/CPU0:S3#show interfaces brief location 0/5/CPU0 | include 802.1Q
Mon Jul  8 23:07:43.929 PDT
      FH0/5/0/1.100        up        up             802.1Q  1518  400000000
RP/0/RP0/CPU0:S3#

RP/0/RP0/CPU0:S3#"
```

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.

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

/* Remove the subinterface, which also automatically deletes all the configuration applied to the subinterface. */

```
RP/0/RP0/CPU0:router(config)# no interface TenGigE 0/2/0/4.10
```

- 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 4095.

Naming notation is *instance.subinterface*, and a period between arguments is required as part of the notation.

Note Repeat to remove other VLAN subinterfaces.

```
RP/0/RP0/CPU0:router(config)# end
or
RP/0/RP0/CPU0:router(config)# commit
```

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

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

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

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

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

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

Configuration Examples for VLAN Interfaces

This section contains the following example:

VLAN Subinterfaces: Example

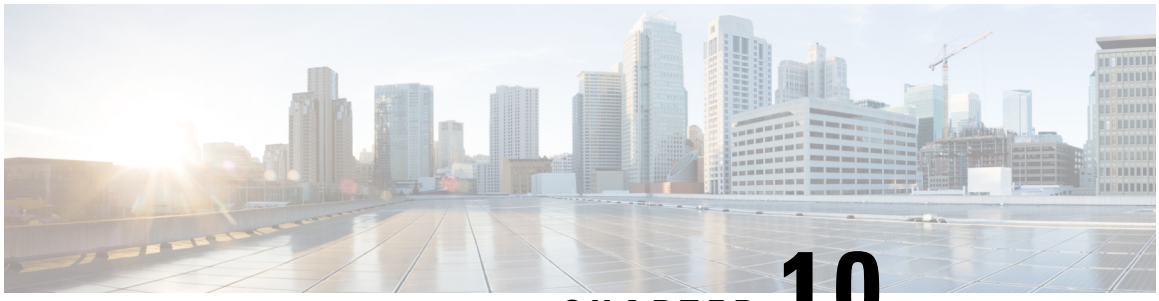
The following example shows how to create three VLAN subinterfaces at one time:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/2/0/4.1

RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 10.0.10.1/24
RP/0/RP0/CPU0:router(config-subif)# interface TenGigE0/2/0/4.2

RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 101
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 10.0.20.1/24
RP/0/RP0/CPU0:router(config-subif)# interface TenGigE0/2/0/4.3

RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 102
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 10.0.30.1/24
RP/0/RP0/CPU0:router(config-subif)# commit
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# exit

RP/0/RP0/CPU0:router# show ethernet trunk bundle-Ether 1
Trunk                              Sub types        Sub states
VLAN trunks: 1,
  1 are 802.1Q (Ether)
Sub-interfaces: 3,
  3 are up.
802.1Q VLANs: 3,
  3 have VLAN Ids,

RP/0/RP0/CPU0:router# show vlan interfaceInterface     St Ly    MTU    Subs    L3
   Up    Down  Ad-Down
Te0/2/0/4.1         802.1Q             10  up
Te0/2/0/4.2         802.1Q             20  up
Te0/2/0/4.3         802.1Q             30  up
RP/0/RP0/CPU0:router# show vlan trunks briefBE1           Up L3   1514    1000
  0    1000     1000       0       0

Summary                       1000        0    1000     1000       0       0

Te0/2/0/4           802.1Q (Ether)        up
```

The following example shows how to create two VLAN subinterfaces on an Ethernet bundle:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface bundle-ether 2
RP/0/RP0/CPU0:router(config-if)# ipv4 address 192.168.2.1/24
RP/0/RP0/CPU0:router(config-if)# exit
RP/0/RP0/CPU0:router(config)# interface bundle-ether 2.1

RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 192.168.100.1/24
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# interface bundle-ether 2.2
```

```
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 200
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 192.168.200.1/24
RP/0/RP0/CPU0:router(config-subif)# exit
RP/0/RP0/CPU0:router(config)# commit
```



CHAPTER 10

Configure IP-in-IP Tunnels

This chapter provides conceptual and configuration information for IP-in-IP tunnels.

Table 10: Feature History for Configure Tunnels

Release 7.0.11	This feature was introduced.
Release 7.0.14	Support for the following feature was introduced in Configure Tunnels: • Extended ACL must match on the outer header for IP-in-IP Decapsulation.

Controlling the TTL Value of Inner Payload Header

Cisco 8000 Routers allow you to control the TTL value of inner payload header of IP-in-IP tunnel packets before it gets forwarded to the next-hop router. This feature enables a router to forward custom formed IP-in-IP stacked packets even if the inner packet TTL is 1. Therefore, this feature enables you to measure the link-state and path reachability from end to end in a network.

Note After you enable or disable the decrement of the TTL value of the inner payload header of a packet, you do not need to reload the line card.

Configuration

To disable the decrement of the TTL value of inner payload header of an IP-in-IP packet, use the following steps:

1. Enter the global configuration mode.
2. Disable the decrement of TTL value of inner payload header of an IP-in-IP packet.

Configuration Example

```
/* Enter the Global Configuration mode. */
Router# configure

/* Disable the decrement of TTL value of inner payload header of an IP-in-IP packet. */
Router(config)# hw-module profile cef ttl tunnel-ip decrement disable
Router(config)# commit
```

Associated Commands

- hw-module profile cef ttl tunnel-ip decrement disable

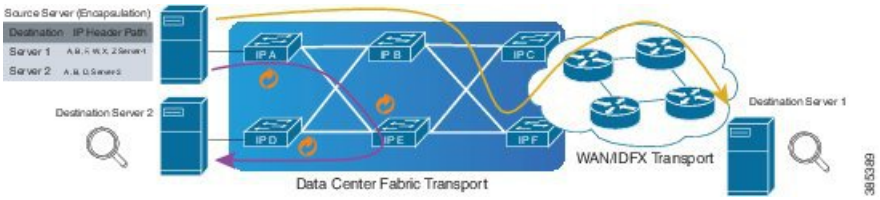
IP-in-IP Decapsulation

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.

Note The router only supports decapsulation and no encapsulation. Encapsulation is done by remote routers.

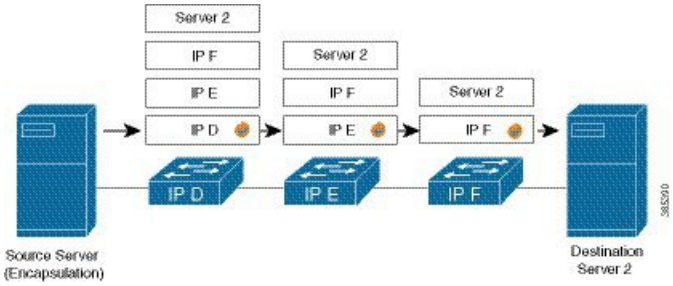
The following topology describes a use case where IP-in-IP encapsulation and decapsulation are used for different segments of the network from source to destination. The IP-in-IP tunnel consists of multiple routers that are used to decapsulate and direct the packet through the data center fabric network.

Figure 5: IP-in-IP Decapsulation Through a Data Center Network



The following illustration shows how the stacked IPv4 headers are decapsulated as they traverse through the decapsulating routers.

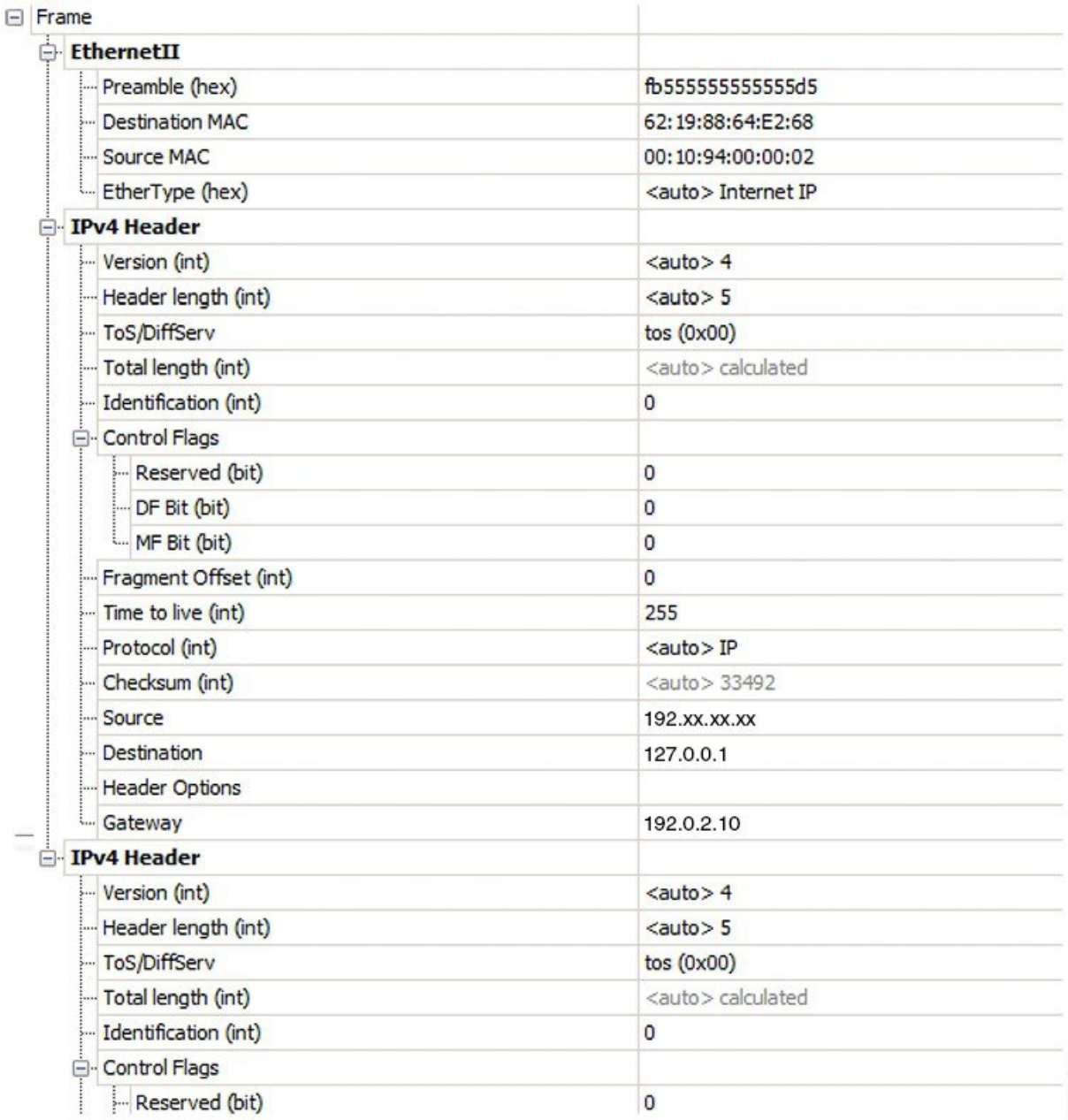
Figure 6: IP Header Decapsulation



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.

Figure 7: Encapsulated Packet



Configuration

You can use the following sample configuration in the routers to decapsulate the packet as it traverses the IP-in-IP tunnel:

```
Router(config)# interface loopback 0
Router(config-if)# ipv4 address 127.0.0.1/32
Router(config-if)# no shutdown
Router(config-if)# interface tunnel-ip 10
```

```
Router(config-if)# ipv4 unnumbered loopback 1
Router(config-if)# tunnel mode ipv4 decap
Router(config-if)# tunnel source loopback 0
```

- **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 with respect to the router interface.

Note You can configure the tunnel destination only if you want to decapsulate packets from a particular destination. If no tunnel destination is configured, then all the ip-in-ip ingress packets on the configured interface are decapsulated.

Running Configuration

```
Router# show running-config interface tunnel-ip 10
...
interface tunnel-ip 10
ipv4 unnumbered loopback 1
tunnel mode ipv4 decap
```

Extended ACL to Match the Outer Header for IP-in-IP Decapsulation

Starting with Cisco IOS XR Software Release 7.0.14, extended ACL has to match on the outer header for IP-in-IP Decapsulation. Extended ACL support reduces mirrored traffic throughput. This match is based only on the IPv4 protocol, and extended ACL is applied to the received outermost IP header, even if the outer header is locally terminated.

Sample configuration:

```
Router#show running-config interface bundle-Ether 50.5
Tue May 26 12:11:49.017 UTC
interface Bundle-Ether50.5
ipv4 address 101.1.5.1 255.255.255.0
encapsulation dot1q 5
ipv4 access-group ExtACL_IPinIP ingress
ipv4 access-group any_dscpegg egress
!


Router#show access-lists ipv4 ExtACL_IPinIP hardware ingress location$
Tue May 26 12:11:55.940 UTC
ipv4 access-list ExtACL_IPinIP
10 permit ipv4 192.168.0.0 0.0.255.255 any ttl gt 150
11 deny ipv4 172.16.0.0 0.0.255.255 any fragments
12 permit ipv4 any any
```

ECMP Hashing Support for Load Balancing

The system inherently supports the n-tuple hash algorithm. The first inner header in the n-tuple hashing includes the source port and the destination port of UDP / TCP protocol headers.

The load balancing performs these functions:

- Incoming data traffic is distributed over multiple equal-cost connections.
- Incoming data traffic is distributed over multiple equal-cost connections member links within a bundle interface.
- Layer 2 bundle and Layer 3 (network layer) load-balancing decisions are taken on IPv4, and IPv6. If it is an IPv4 or an IPv6 payload, then an n-tuple hashing is done.
- An n-tuple hash algorithm provides more granular load balancing and used for load balancing over multiple equal-cost Layer 3 (network layer) paths. The Layer 3 (network layer) path is on a physical interface or on a bundle interface.
- The n-tuple load-balance hash calculation contains:
    - Source IP address
    - Destination IP address
    - IP Protocol type
    - Router ID
    - Source port
    - Destination port
    - Input interface

Configure Tunnel Destination with an Object Group

Table 11: Feature History Table

Feature Name	Release Information	Description
Configure Tunnel Destination with an Object Group	Release 7.5.4	You can now assign an object group as the destination for an IP-in-IP decapsulation tunnel. With this functionality, you could configure an IPv4 or IPv6 object group consisting of multiple IPv4 or IPv6 addresses as the destination for the tunnel instead of a single IPv4 or IPv6 address. Using an object group instead of a singular IP address. This helps reduce the configuration complexity in the router by replacing the multiple tunnels with one destination with a single decapsulation tunnel that supports a diverse range of destinations The feature introduces these changes: • **CLI**: New **tunnel destination** command. • **YANG Data Model**: New **object-group** option supported in Cisco-IOS-XR-um-if-tunnel-cfg.yang Cisco native model (see GitHub).

In IP-in-IP Decapsulation, the router accepts a packet on a tunneled interface only when the tunnel IP address matches the source IP address of the incoming packets. With this implementation, the user needs to configure separate interface tunnels for each IP address that the router needs to receive the traffic packets. This limitation often leads to configuration overload on the router.

You can eliminate the configuration overload on the router by assigning an object group as the tunnel destination for IPv4 and IPv6 traffic types. That is, the router matches the source IP address of the incoming packet against the object group available as the tunnel destination. The decapsulation tunnel accepts the incoming traffic packets when there's a match between the packet source and the object group. Otherwise, the router drops the packets.

Restrictions

The following restrictions are applicable to the tunnel destination with an object group feature:

- GRE tunnels don't support configuring object groups as the tunnel destination.

- The router supports configuring tunnel destination with an object group only when the tunnel source is tunnel source direct.
- You can configure the object group as tunnel destination only on default VRF.
- Configuring object groups as the tunnel destination isn't applicable to tunnel encapsulation.
- Subinterfaces don't support configuring object groups as the tunnel destination.
- Configuring object groups as the tunnel destination feature is mutually exclusive with ACL and QoS features.
- The tunnel destination feature supports only IPv4 and IPv6 object groups.
- The router does not support changing tunnel configuration after its creation. Configure the tunnel source direct and tunnel destination with an object group while creating the tunnel only.

Prerequisites

- Define an object group including the network elements for the tunnel destination.
- Enable the tunnel source direct feature. For more information, see decapsulation using tunnel source direct.

Configuration Example

This section provides an example for configuring the tunnel destination with an object group:

Configuration

IPv4:

```
Router# configure
/* Configure the IPv4 object group */
Router(config)# object-group network ipv4 Test_IPv4
Router(config-object-group-ipv4)# 192.0.2.0/24
Router(config-object-group-ipv4)# 198.51.100.0/24
Router(config-object-group-ipv4)# 203.0.113.0/24
Router(config-object-group-ipv4)# commit
Router(config-object-group-ipv4)# exit

/* Enters the tunnel configuration mode */
Router(config)# interface tunnel TestIPv4


/* Configures the tunnel mode */
Router(config-if)# tunnel mode ipv4 decap

/* Configures the tunnel to accept all packets with destination address matching the IP
addresses on the router */
Router(config-if)# tunnel source direct

/* Configures the tunnel to accept all packets with destination address that are in the
specified object group */
Router(config-if)# tunnel destination object-group ipv4 Test_IPv4

Router(config-if)# no shutdown
Router(config-if)# commit
Router(config-if)# exit
```

IPv6:

```
Router# configure
/* Configure the IPv6 object group */
Router(config)# object-group network ipv6 Test_IPv6
Router(config-object-group-ipv6)# 2001:DB8::/32
Router(config-object-group-ipv6)# 2001:DB8::/48
Router(config-object-group-ipv6)# commit
Router(config-object-group-ipv6)# exit

/* Enters the tunnel configuration mode */
Router(config)# interface tunnel TestIPv6


/* Configures the tunnel mode */
Router(config-if)# tunnel mode ipv6 decap

/* Configures the tunnel to accept all packets with destination address matching the IP
addresses on the router */
Router(config-if)# tunnel source direct

/* Configures the tunnel to accept all packets with destination address that are in the
specified object group */
Router(config-if)# tunnel destination object-group ipv6 Test_IPv6

Router(config-if)# no shutdown
Router(config-if)# commit
Router(config-if)# exit
```

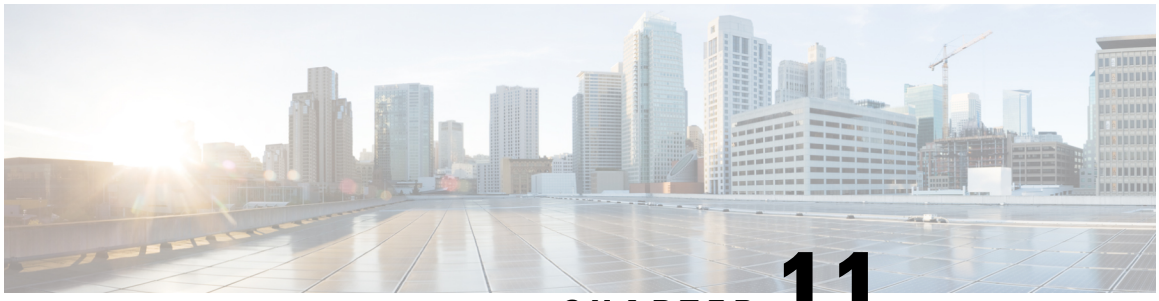
Running Configuration

```
Router# show running config object-group
object-group network ipv4 Test_IPv4
192.0.2.0/24
198.51.100.0/24
203.0.113.0/24
!
object-group network ipv6 Test_IPv6
2001:DB8::/32
2001:DB8::/48
!

Router# show interface tunnel TestIPv4
interface TunnelTestIPv4
  tunnel mode ipv4 decap
   tunnel source direct
  tunnel destination object-group ipv4 Test_IPv4
  no shutdown
!

Router# show interface tunnel TestIPv6
interface TunnelTestIPv6
 tunnel mode ipv6 decap
   tunnel source direct
  tunnel destination object-group ipv6 Test_IPv6
 no shutdown
!
```

Verification

```
Router# show tunnel ip ea database

----- node0_0_CPU0 -----
tunnel ifhandle 0x80022cc
tunnel source 161.115.1.2
tunnel destination address group Test_IPv4
```

```
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
```



CHAPTER 11

Configuring Controllers

This chapter describes the Optics Controller for the 36-port QSFP56-DD 400 GbE and 48-port QSFP28 100 GbE Line Cards. This chapter also describes the procedures used to configure the controllers.

Note When two MACsec enabled Cisco 8000 Series Routers with Coherent Line Cards are connected, there is no compatibility between Coherent Line Cards of IOS XR Release.

Following controller configuration options are supported on the router:

- breakout - Configure breakout mode (‘breakout 4x10’ only.)
- clear - Clear the uncommitted configuration.
- commit - Commit the configuration changes to running.
- do - Run an exec command.
- end - Exit from configure mode.
- exit - Exit from this submode.
- ext-description - Set ext-description for this controller.
- no - Negate a command or set its defaults.
- pwd - Commands used to reach current submode.
- root - Exit to the global configuration mode.
- show - Show contents of configuration.

- How to Configure Controllers, on page 117

How to Configure Controllers

This section contains the following procedures:

Configuring Optics Controller

Configuring optics controller of breakout 4x10:

```
RP/0/RP0/CPU0:uut#configure
Fri Oct 11 16:22:31.222 UTC
RP/0/RP0/CPU0:uut(config)#controller optics 0/1/0/28
RP/0/RP0/CPU0:uut(config-Optics)#breakout 4x10
RP/0/RP0/CPU0:uut(config-Optics)#commit
Fri Oct 11 16:23:26.868 UTC
RP/0/RP0/CPU0:uut(config-Optics)#end
RP/0/RP0/CPU0:uut#
RP/0/RP0/CPU0:uut#show running-config controller optics 0/1/0/28
Fri Oct 11 16:23:41.273 UTC
controller Optics0/1/0/28
breakout 4x10
!
```

Configuring Line Loopback on Grey Optics

Grey optics, which implies no color, is an optical transceiver that uses only one or two wavelengths of light to transmit and receive data. Grey optics are reasonably priced and ideal for short-distance transmission.

From IOS XR Release 7.5.4 onwards, it is possible to enable line loopback functionality on grey optics. Line loopback is the routing of traffic or data or signals received on an optical module back to their source. The source can be another router within the same network. When the traffic is received back at the source, you can utilize the received traffic to troubleshoot physical connection issues or network issues, such as traffic loss or a faulty optics. Line loopback is also known as media side input loopback.

Consider a scenario where there are two routers in a network, Router-A and Router-B. Router-A has a port and this port is connected to an optical module, OM-A. Similarly, Router-B is connected to another optical module OM-B. OM-A and OM-B are connected via a cable. In this setup, Router-A is the host side and OM-A is the media side. The traffic travels from Router B via OM-B to OM-A. If you have line loopback enabled on OM-A, the traffic that is received at OM-A (media side), is routed or looped back to Router B via OM-B.

You can now utilize the traffic received at Router-B to troubleshoot if there was any data loss. Or, if there is no traffic or signal received at Router-B, it could indicate that OM-A has physical connection issues or has malfunctioned.

Guidelines and Limitations

- Line loopback mode is supported only on Cisco 8000 Series routers, and Q100 and Q200-based line cards.
- Host side line loopback isn't supported.

Configure Line Loopback

To enable line loopback on the grey optics, use the **controller optics** *r/s/i/p* **loopback line** command.

This example shows how to enable line loopback configuration on grey optics:

```
Router#config
Router(config)#controller optics 0/4/0/4
Router(config-CoDSP)#loopback line
Router(config-CoDSP)#secondary-admin-state maintenance
Router(config-CoDSP)#commit
```

Running Configuration

This example shows the running configuration on grey optics:

```
Router#show run controller optics 0/4/0/4
Mon Jan 23 17:07:29.034 UTC
controller Optics0/4/0/4
 loopback line
 sec-admin-state maintenance
!

Router#
```

Verification

This example shows how to verify the line loopback configuration on grey optics:

```
Router#show controller optics 0/4/0/4 information loopback
Fri Jan 20 17:18:05.541 UTC

 Supported Loopback Types :
============================
 [1.] Media Line
 [2.] Host Line

 Unsupported Loopback Types :
==============================
 [1.] Media Internal
 [2.] Host Internal
 Media Configured Loopback : Media Loopback Line
 Media Applied Loopback    : Media Loopback Line

Router#
```

Configuring Low Power Mode on Grey Optics

Overview

Starting Release 7.5.4, Cisco IOS XR supports low power mode feature on the controller optics that allows you to configure ports used for grey optics in a low power mode. Use the **controller optics** *R/S/I/P* **shutdown** command to configure the low power mode. To disable the low power mode, use the **no** form of this command.

Note The **shutdown** option in the **controller optics** *R/S/I/P* **shutdown** command imples that the optical module will be set to the low power mode; there is no traffic transmission but you can still execute the **show controllers optics** command to view the optic details.

Guidelines and Limitations

- Low power mode is supported only on the 400GbE ports on the Cisco 8000 Series routers, and Q100 and Q200-based line cards.

Low Power Mode Configuration Example

The following example shows how to configure the low power mode on the optics controller:

```
Router#config
Router(config)#controller optics 0/4/0/4
Router(config-Optics)#shutdown
Router(config-Optics)#commit
Router(config-Optics)#exit
Router(config)#exit
```

Running Configuration

This example shows the running configuration for the optics controller:

```
Router#show running-config controller optics 0/4/0/4
Mon Jan 23 17:07:29.034 UTC
controller Optics0/4/0/4
 shutdown
!
```

Verification

This example shows how to verify the low power mode set on an optic:

```
Router#show controller optics 0/4/0/4
Tue Jan 24 00:14:28.531 UTC

Controller State: Administratively Down /* This indicates that the optics is in low power
mode */

Transport Admin State: Out Of Service /* This indicates that the optics is in low power
mode */

Laser State: Off /* This indicates that the optics is in low power mode */

LED State: Not Applicable

FEC State: FEC ENABLED

Optics Status

Optics Type: QSFPDD 400G FR4
Wavelength = 1301.00 nm

Alarm Status:
-------------
Detected Alarms:
LOW-TX1-PWR LOW-TX2-PWR LOW-TX3-PWR LOW-TX4-PWR
LOW-TX1_LBC LOW-TX2_LBC LOW-TX3_LBC LOW-TX4_LBC

LOS/LOL/Fault Status:
Detected LOS/LOL/FAULT: RX-LOS RX-LOL TX-LOL



Performance Monitoring: Disable

THRESHOLD VALUES
----------------

Parameter High Alarm Low Alarm High Warning Low Warning
----------------------- ---------- --------- ------------ -----------
Rx Power Threshold(dBm) 6.4 -40.0 6.4 -40.0
Rx Power Threshold(mW) 4.3 0.0 4.3 0.0
Tx Power Threshold(dBm) 6.4 -4.3 6.4 -4.3
Tx Power Threshold(mW) 4.3 0.3 4.3 0.3
LBC Threshold(mA) 120.00 20.00 110.00 30.00
Temp. Threshold(celsius) 75.00 10.00 75.00 10.00
Voltage Threshold(volt) 3.56 3.03 3.56 3.03

Configured Tx Power = 0.00 dBm
Configured Tx Power(mW) = 1.00 mW
Configured CD High Threshold = 0 ps/nm
Configured CD lower Threshold = 0 ps/nm
Configured OSNR lower Threshold = 0.00 dB
```

```
Configured DGD Higher Threshold = 0.00 ps
Polarization parameters not supported by optics

Lane Laser Bias TX Power RX Power Output Frequency
---- ---------- ---------- ---------- ----------------
0 0.0 mA -40.00 dBm -40.00 dBm N/A
1 0.0 mA -40.00 dBm -40.00 dBm N/A
2 0.0 mA -40.00 dBm -40.00 dBm N/A
3 0.0 mA -40.00 dBm -40.00 dBm N/A

Temperature = 28.78 Celsius
Voltage = 3.31 V

Transceiver Vendor Details

Form Factor : QSFP-DD
Optics type : QSFPDD 400G FR4
Name : XYZ
OUI Number : 44.7c.7f
Part Number : T-DQ4CNT-NFB
Rev Number : 50
Serial Number : INLBYI552537
PID : T-DQ4CNT-NFB
VID : 50
Firmware Version : Major.Minor.Build
Active : 234.147.0
Inactive : 0.0.0
Date Code(yy/mm/dd) : 22/06/16
```