Configuring Ethernet Interfaces

This module describes the configuration of Ethernet interfaces.

The distributed 10-Gigabit, 25-Gigabit Ethernet, 40-Gigabit, 100-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, Layer 2 switches and Layer 3 switches.

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Introduction of IP MTU on Q200-based Systems	Release 7.5.2	You can configure IP MTU for IPv4 and IPv6 on a Layer 3 interface. Depending on your specific network requirements, this ability to specify IP MTU settings helps optimize router data transmission. Use the `show ipv4/ipv6 interfaces` command to view the IP MTU configurations.

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.
- Starting with Cisco IOS XR Release 24.1.1, the `openconfig-if-ip.yang` open config model supports a new leaf, `oc-if:interfaces/oc-if:interface/oc-if:subinterfaces/oc-if:subinterface/ipv4/addresses/address[ip]/config/type`. When using the `openconfig-if-ip.yang` model to update or revise the IP address for an interface, you must specify the IP address as either primary or secondary. You can have only one IP address as primary, but you can have multiple addresses configured as secondary.
- 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 2: 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)

Layer 2 VPN on Ethernet Interfaces

Layer 2 Virtual Private Network (L2VPN) connections emulate the behavior of a LAN across an L2 switched, IP or MPLS-enabled IP network, allowing Ethernet devices to communicate with each other as if they were connected to a common LAN segment.

The L2VPN feature enables service providers (SPs) to provide Layer 2 services to geographically disparate customer sites. Typically, an SP uses an access network to connect the customer to the core network. On the router, this access network is typically Ethernet.

Traffic from the customer travels over this link to the edge of the SP core network. The traffic then tunnels through an L2VPN over the SP core network to another edge router. The edge router sends the traffic down another attachment circuit (AC) to the customer's remote site.

On the router, an AC is an interface that is attached to an L2VPN component, such as a bridge domain.

The L2VPN feature enables users to implement different types of end-to-end services.

Switching takes place through local switching where traffic arriving on one AC is immediately sent out of another AC without passing through a pseudowire.

Keep the following in mind when configuring L2VPN on an Ethernet interface:

- L2VPN links support QoS (Quality of Service) and MTU (maximum transmission unit) configuration.
- If your network requires that packets are transported transparently, you may need to modify the packet’s destination MAC (Media Access Control) address at the edge of the Service Provider (SP) network. This prevents the packet from being consumed by the devices in the SP network.

Use the **show interfaces** command to display AC information.

To attach Layer 2 service policies, such as QoS, to the Ethernet interface, refer to the appropriate Cisco IOS XR software configuration guide.

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).

IP MTU

In IP protocol, Maximum Transmission Unit (MTU) refers to the maximum size of an IP packet that the system transmits without fragmentation over a given medium. The size of an IP packet includes IP headers but excludes headers from the data link layer, also known as the Ethernet headers. The default IP MTU on all router interfaces is 1500 bytes, when IP is enabled by using the IP address configuration commands. However, you can configure the IP MTU to different value as well.

Starting Cisco IOS XR Release 7.5.2, the system supports IP MTU (IPv4 and IPv6) on Q200-based systems on the following Cisco 8000 Series router and line card:

- 8201-32FH
- 88-LC0-36FH-M

How is Ethernet MTU different from IP MTU?

Ethernet MTU defines the maximum packet size that an interface supports, while IP MTU defines the MTU size of an IP packet.

How is IP MTU calculated?

The following scenarios provide information on how the system calculates IP MTU size, when:

- Ethernet MTU is not configured, the system sets:
  - Default value as 1514 bytes on a physical or bundle main interface
  - 1518 bytes for single tag VLAN interface
  - 1522 bytes for double tag VLAN (QinQ) interface

Note In this case, IP MTU value will be 1500 bytes for IPv4 and IPv6 MTUs.

- Ethernet MTU is configured as X bytes, the system sets:
  - X bytes on a physical or bundle main interface
  - X+4 bytes for single tag VLAN interface
  - X+8 bytes for double tag VLAN (QinQ) interface

Note In this case, IP MTU is X-14 bytes for IPv4 and IPv6 MTUs.

The following are some of the important guidelines for IP MTU size:

- When no Ethernet MTU and IP MTU is configured, the default value is 1500B.

- When Ethernet MTU and no IP MTU is configured, IP MTU value in the hardware is Ethernet MTU-14B.
- IP MTU value can't be more than Ethernet MTU value. For example, if the Ethernet MTU value is 3000 bytes and you configure IP MTU value as 5000 bytes. The system sets the IP MTU value as 2986 (3000-14)

What is Maximum Receive Unit (MRU) and how is it different from MTU?

MRU is the largest packet size that an interface can receive. This is an ingress parameter. Usually, MRU equals MTU. However, you can't configure an MRU value. The Ethernet MTU, also known as a Layer2 (L2) value that you configure on a physical interface is also applied as the MRU of that physical interface.

The following table lists Ethernet MTU, IPv4, and IPv6 MTU support across various platforms and their limitations as applicable:

Table 3: IP MTU Support Across Platforms

ASIC	Ethernet MTU Check	IPv4 and IPv6 MTU
Q100-based	Ethernet MTU check on an egress interface is not supported.	Supported and the system derives IP MTU value from Ethernet MTU.
Q200-based	Ethernet MTU check on an egress interface is not supported.	Supported **Note** IPv4 and IPv6 can have their own separate MTU values.

IP MTU Checks

Cisco IOS XR supports MRU checks, ethernet MTU checks, and IP MTU checks. These checks decide if an IP payload packets needs fragmentation or not. Ethernet MTU and IP MTU check is applied on all egress traffic for each packet that flows in the system. In the forwarding plane, also known as dataplane, the IP packet length is compared against IP MTU value applied on the L3 forwarding interface. Full packet length is compared against ethernet MTU value applied on the physical port.

The following table describes IP MTU check to each forwarding flow on Q200-based and Q100-based systems.

Table 4: IP MTU Check

ASIC	IP MTU Check
Q100-based	These interfaces implement IP MTU checks and provide fragmentation for all IP payload packets. A single value for both IPv4 and IPv6 is supported.
Q200-based	These interfaces implement IP MTU checks and provide fragmentation for all IP payload packets.
Mixed System with Q100 and Q200-based line cards	Interfaces with Q100-based system provides its behavior and interfaces with Q200-based systems provides its behavior as described in this table.

IP MTU Configuration Guidelines

An Ethernet interface has a default of 1514 bytes. IP MTU is always derived from the default Ethernet MTU. If Ethernet MTU is configured, IP MTU is derived from the previously configured Ethernet MTU.

The following are the configuration guidelines for IP MTU across various platforms:

Guidelines for Q200-based Systems

- If you don't define any Ethernet MTU configuration, the system assigns the default value of Ethernet packet size of 1514 bytes.
- For IPv4 and IPv6 MTUs, 1500 bytes is used, which is derived by subtracting 14 bytes of Ethernet header size from its default value.
- On any interface, if an Ethernet MTU is configured, the new configuration takes higher precedence than the default interface configuration. However, in such a scenario, the new configuration doesn't get applied on the subinterfaces. The rest of the interfaces continue to work with the default Ethernet MTU configurations.
- For the double tag (QinQ) packets, MRU and Ethernet MTU value are derived by adding 8 bytes to the configured value.
- IPv4 and IPv6 MTU values are derived by subtracting 14 bytes of the Ethernet header size out of the configured value.
- If an IPv4 or IPv6 MTU is configured, it is applied to IP MTU value. MRU or MTU values remain unaffected by this configuration.
- Configuration restrictions apply to validate that the IP MTU value is smaller than Ethernet MTU value that is configured on an interface.

The following sample table explains how the system calculates the Ethernet MTU, IPv4, and IPv6 MTU values when configured on various interfaces that are Q200-based systems.

Table 5: Interfaces and Configurations

Interface Type	**Default Configurations** *when, no Ethernet or IP MTU configuration is applied*	**Ethernet MTU Configurations** *for example, MTU = 1614*	**IPv4/IPv6 Configurations** *for example, IPv4 | IPv6 MTU = 1550, Ethernet MTU = 1614*
Any Main Interface	NA	Configuration that is applied on any main interface and its subinterfaces is effective. Configuration that is applied on any subinterface is not effective.	NA
Physical Port	MRU = 1514 + 8	MRU = 1614 + 8	MRU = 1614 + 8

Interface Type	Default Configurations *when, no Ethernet or IP MTU configuration is applied*	Ethernet MTU Configurations *for example, MTU = 1614*	IPv4/IPv6 Configurations *for example, IPv4 | IPv6 MTU = 1550, Ethernet MTU = 1614*
L3 Main Interface (Physical or bundle)	Ethernet MTU = 1514 + 8 IPv4/IPv6 MTU = 1500	Ethernet MTU = 1614 + 8 IPv4/IPv6 MTU = 1614 - 14	Ethernet MTU = 1614 + 8 IPv4/IPv6 MTU = 1550
L3 Sub-interface (Physical or bundle)	Ethernet MTU = 1514 + 8 IPv4/IPv6 MTU = 1500	**Note** Configuration not applicable on subinterfaces. Takes main interface configuration. Ethernet MTU = Not applicable IPv4/IPv6 MTU = 1614 - 14	Ethernet MTU = 1614 + 8 IPv4/IPv6 MTU = 1550
SVI/BVI	IPv4/IPv6 MTU = 1500	Ethernet MTU = Not applicable IPv4/IPv6 MTU = 1600	IPv4/IPv6 MTU = 1550
Ethernet Main Interface	Ethernet MTU = 1514 + 8	Ethernet MTU = 1614 + 8 IPv4/IPv6 MTU = Not applicable	Not applicable Ethernet MTU = 1614 + 8
Ethernet Sub-interface	Ethernet MTU = 1514 + 8	Ethernet MTU = 1614 + 8 IPv4/IPv6 MTU = Not applicable	Not applicable Ethernet MTU = 1614 + 8

Guidelines for Q100-based Systems

- If you don’t define any MTU configuration in the system, the system assigns default value of 1514 bytes to the Ethernet packets.
- An MRU of 1522 bytes is set to allow for any double tag packets on an interface.
- Ethernet MTU check on egress interface is not supported.
- For IPv4 and IPv6 MTUs, 1500 bytes is used, which is derived by subtracting 14 bytes of Ethernet header size from its default value.
- If you configure an IP MTU (IPv4/IPv6 MTU) on any interface, the configuration does not take effect.

The following table lists the MTU configurations and MTU calculations on various interfaces for Q100-based systems:

Table 6: Interfaces and Configurations

Interface Type	Default Configurations	Interface MTU Configurations ***where, MTU = 1614***
Any Main Interface	NA	Configuration applied on any main interface and its subinterfaces is effective. Configuration applied on any sub-interface is not effective.
Physical Port	MRU = 1514 + 8	MRU = 1614 + 8
L3 Main Interface (Physical or bundle)	Ethernet MTU check on egress interface is not supported. IPv4/IPv6 MTU = 1522 applied on full packet size	Ethernet MTU check on egress interface is not supported. IPv4/IPv6 MTU = 1622 applied on full packet size
L3 Subinterface (Physical or bundle)	Ethernet MTU check on egress interface is not supported. IPv4/IPv6 MTU = 1522 applied on full packet size	**Note** Configuration not applicable on sub-interfaces. Takes main interface configuration. Ethernet MTU = Not applicable IPv4/IPv6 MTU = 1622 applied on full packet size
SVI and BVI	Ethernet MTU is not applicable IPv4/IPv6 MTU = 1522 applied on full packet size	
Ethernet Main Interface	Ethernet MTU check on egress interface is not supported. IPv4/IPv6 MTU is not applicable	
Ethernet Subinterface	Ethernet MTU check on egress interface is not supported. IPv4/IPv6 MTU is not applicable	
GRE tunnel	Ethernet MTU check on egress interface is not supported. IPv4/IPv6 MTU = 1522 applied on full packet size	

IP MTU Limitations and Feature Support

The following are the limitations and feature support for IP MTU:

- MPLS MTU is not supported.
- GRE/IPinIP MTU is not supported.

Note GRE interface uses the configured L3 MTU. If an egress packet is above the configured L3 MTU value, the packet is discarded.

- Multicast MDT MTU is not supported.
- Ethernet MTU configuration on sub-interfaces is not supported.
- Ethernet MTU configuration on BVI interfaces is not supported.

Important Although CLIs for the unsupported features might be available on your router, they are not functional.

IP MTU Scale

The following are the scale support and limitations for IP MTU configurations:

- Ethernet MTU configuration is allowed on every physical or bundle interface without any scale limits.
- IPv4/IPv6 MTU configuration is allowed on every L3 forwarding interface. However, the system applies it as an IP MTU profile with a combination of <IPv4 MTU and IPv6 MTU> set. Each profile takes a unique set of these two MTUs, and supports eight such unique MTUs. The IP MTU profiles are stored identically across all NPUs of a given line card.
- When system runs out of resources (OOR), the system generates syslog messages while the continuing to use the previously configured IP MTU values.

Configure IP MTU

To configure IPv4, or IPv6 MTU, you must be in the configuration mode on L3 interface.

To configure IP MTU, use the following configuration steps:

```
Router#config t

Router#interface HundredGigE0/0/0/33

Router:abc(config)#

Router:abc(config-if)#ipv4 mtu 2000

Router:abc(config-subif)#commit

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

Running Configuration

```
Router:abc#
Router:abc#sh ipv4 int HundredGigE0/0/0/33
Thu Apr 28 16:54:10.820 UTC
HundredGigE0/0/0/33 is Shutdown, ipv4 protocol is Down
  Vrf is default (vrfid 0x60000000)
  Internet address is 10.10.10.91/29
  MTU is 9642 (2000 is available to IP)
```

Verification

```
Router:abc#sh int HundredGigE0/0/0/33
Thu Apr 28 16:57:23.390 UTC
HundredGigE0/0/0/33 is administratively down, line protocol is administratively down
  Interface state transitions: 0
  Hardware is HundredGigE, address is e41f.7bde.123c (bia e41f.7bde.123c)
  Internet address is 194.19.242.94/29
  MTU 9642 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 100000Mb/s, 100GBASE-AOC, link type is force-up
  output flow control is off, input flow control is off
  Carrier delay (up) is 10 msec
  loopback not set,
  ARP type ARPA, ARP timeout 04:00:00
  Last input never, output never
  Last clearing of "show interface" counters never
  30 second input rate 0 bits/sec, 0 packets/sec
  30 second output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
              0 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


Router:abc#
```

Configure IPv4 MTU

To configure IPv4 MTU, use the following configuration steps:

```
Router#config t

Router#interface HundredGigE0/0/0/33

Router:abc(config)#

Router:abc(config-if)#ipv4 mtu 2000

Router:abc(config-subif)#commit

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

Running Configuration

```
Router:abc#sh run int HundredGigE0/0/0/33
Thu Apr 28 16:22:06.796 UTC
interface HundredGigE0/0/0/33
mtu 9642
ipv4 mtu 2000
ipv4 address 10.10.10.1 255.255.255.248
ipv6 address 10::1:10:9/112
load-interval 30
!
```

Verification

```
Router:abc#sh ipv4 int HundredGigE0/0/0/33
Thu Apr 28 16:54:10.820 UTC
HundredGigE0/0/0/33 is Shutdown, ipv4 protocol is Down
  Vrf is default (vrfid 0x60000000)
  Internet address is 10.10.10.1/29
  MTU is 9642 (2000 is available to IP)
```

Configure IPv6 MTU

To configure IPv6 MTU, use the following configuration steps:

```
Router#config t

Router#interface HundredGigE0/0/0/33

Router:abc(config)#

Router:abc(config-if)#ipv6 mtu 3000

Router:abc(config-subif)#commit

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

Running Configuration

```
Router:abc#sh run int HundredGigE0/0/0/33
Thu Apr 28 16:23:09.141 UTC
interface HundredGigE0/0/0/33
mtu 9642
ipv4 mtu 2000
ipv4 address 10.10.10.1 255.255.255.248
ipv6 mtu 3000
ipv6 address 10::10:10:9/112
load-interval 30
!

Router:abc#
```

Verification

```
Router:abc#sh ipv6 int HundredGigE0/0/0/33
Thu Apr 28 16:54:41.222 UTC
HundredGigE0/0/0/33 is Shutdown, ipv6 protocol is Down, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::e61f:7bff:fede:123c [TENTATIVE]
  Global unicast address(es):
    194::19:242:94, subnet is 10::10:10:0/112 [TENTATIVE]
  Joined group address(es): ff02::2 ff02::1
  MTU is 9642 (3000 is available to IPv6)
```

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.

Interfaces and Subinterfaces on the Router

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
Interfaces and Subinterfaces	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) The support for Interfaces and subinterfaces is now extended to these fixed systems and line cards: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Interfaces and Subinterfaces	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) The interfaces are initially configured as main trunk interfaces, which can be either physical or bundle types, with physical interfaces being automatically created by the system. The benefit of this setup is the flexibility it provides, allowing users to create logical subinterfaces under trunk interfaces with unique identifiers, enabling efficient network segmentation and management. This feature is now supported on routers with 88-LC1-36EH line cards.

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
!
```

Layer 2, Layer 3, and EFPs

On the router, a trunk interface can be either a Layer 2 or Layer 3 interface. A Layer 2 interface is configured using the **interface** command with the **l2transport** keyword. When the **l2transport** keyword is not used, the interface is a Layer 3 interface. Subinterfaces are configured as Layer 2 or Layer 3 subinterface in the same way.

A Layer 3 trunk interface or subinterface is a routed interface and can be assigned an IP address. Traffic sent on that interface is routed.

A Layer 2 trunk interface or subinterface is a switched interface and cannot be assigned an IP address. A Layer 2 interface must be connected to an L2VPN component. Once it is connected, it is called an access connection.

Subinterfaces can only be created under a Layer 3 trunk interface. Subinterfaces cannot be created under a Layer 2 trunk interface.

A Layer 3 trunk interface can have any combination of Layer 2 and Layer 3 interfaces.

The following example shows an attempt to configure a subinterface under an Layer 2 trunk and the commit errors that occur. It also shows an attempt to change the Layer 2 trunk interface to an Layer 3 interface and the errors that occur because the interface already had an IP address assigned to it.

```
RP/0/RP0/CPU0:router# config
Mon Sep 21 12:05:33.142 EDT
RP/0/RP0/CPU0:router(config)# interface HundredGigE0/5/0/0
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.0.0.1/24
RP/0/RP0/CPU0:router(config-if)# commit
RP/0/RP0/CPU0:Sep 21 12:05:57.824 : config[65794]: %MGBL-CONFIG-6-DB_COMMIT : Configuration
 committed by user 'root'. Use 'show configuration commit changes 1000000160' to view the
changes.
RP/0/RP0/CPU0:router(config-if)# end
RP/0/RP0/CPU0:Sep 21 12:06:01.890 : config[65794]: %MGBL-SYS-5-CONFIG_I : Configured from
console by root
RP/0/RP0/CPU0:router# show run | begin HundredGigE0/5/0/0
Mon Sep 21 12:06:19.535 EDT
Building configuration...
interface HundredGigE0/5/0/0
 ipv4 address 10.0.0.1 255.255.255.0
 negotiation auto
!
interface HundredGigE0/5/0/1
 shutdown
!
RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router# conf
Mon Sep 21 12:08:07.426 EDT
RP/0/RP0/CPU0:router(config)# interface HundredGigE0/5/0/0 l2transport
RP/0/RP0/CPU0:router(config-if-l2)# commit

% Failed to commit one or more configuration items during a pseudo-atomic operation. All
changes made have been reverted. Please issue 'show configuration failed' from this session
 to view the errors
RP/0/RP0/CPU0:router(config-if-l2)# no ipv4 address
RP/0/RP0/CPU0:router(config-if)# commit
RP/0/RP0/CPU0:Sep 21 12:08:33.686 : config[65794]: %MGBL-CONFIG-6-DB_COMMIT : Configuration
 committed by user 'root'. Use 'show configuration commit changes 1000000161' to view the
changes.
RP/0/RP0/CPU0:router(config-if)# end
RP/0/RP0/CPU0:Sep 21 12:08:38.726 : config[65794]: %MGBL-SYS-5-CONFIG_I : Configured from
console by root
RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router# show run interface HundredGigE0/5/0/0
Mon Sep 21 12:09:02.471 EDT
interface HundredGigE0/5/0/0
 negotiation auto
 l2transport
 !
!
RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router# conf
Mon Sep 21 12:09:08.658 EDT
```

```
RP/0/RP0/CPU0:router(config)# interface HundredGigE0/5/0/0.0
                                                        ^
RP/0/RP0/CPU0:router(config)# interface HundredGigE0/5/0/0.0
RP/0/RP0/CPU0:router(config-subif)# commit

% Failed to commit one or more configuration items during a pseudo-atomic operation. All
changes made have been reverted. Please issue 'show configuration failed' from this session
 to view the errors
RP/0/RP0/CPU0:router(config-subif)#
RP/0/RP0/CPU0:router(config-subif)# interface HundredGigE0/5/0/0
RP/0/RP0/CPU0:router(config-if)# no l2transport
RP/0/RP0/CPU0:router(config-if)# interface HundredGigE0/5/0/0.0
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 99
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 11.0.0.1/24
RP/0/RP0/CPU0:router(config-subif)# interface HundredGigE0/5/0/0.1 l2transport
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 700
RP/0/RP0/CPU0:router(config-subif)# commit
RP/0/RP0/CPU0:Sep 21 12:11:45.896 : config[65794]: %MGBL-CONFIG-6-DB_COMMIT : Configuration
 committed by user 'root'. Use 'show configuration commit changes 1000000162' to view the
changes.
RP/0/RP0/CPU0:router(config-subif)# end
RP/0/RP0/CPU0:Sep 21 12:11:50.133 : config[65794]: %MGBL-SYS-5-CONFIG_I : Configured from
console by root
RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router# show run | b HundredGigE0/5/0/0
Mon Sep 21 12:12:00.248 EDT
Building configuration...
interface HundredGigE0/5/0/0
 negotiation auto
!
interface HundredGigE0/5/0/0.0
 ipv4 address 11.0.0.1 255.255.255.0
 encapsulation dot1q 99
!
interface HundredGigE0/5/0/0.1 l2transport
 encapsulation dot1q 700
!
interface HundredGigE0/5/0/1
 shutdown
!
```

All subinterfaces must have unique encapsulation statements, so that the router can send incoming packets and frames to the correct subinterface. If a subinterface does not have an encapsulation statement, the router will not send any traffic to it.

In Cisco IOS XR, an Ethernet Flow Point (EFP) is implemented as a Layer 2 subinterface, and consequently, a Layer 2 subinterface is often called an EFP.

A Layer 2 trunk interface can be used as an access connection. However, a Layer 2 trunk interface is not an EFP because an EFP, by definition, is a substream of an overall stream of traffic.

Cisco IOS XR also has other restrictions on what can be configured as a Layer 2 or Layer 3 interface. Certain configuration blocks only accept Layer 3 and not Layer 2. For example, OSPF only accepts Layer 3 trunks and subinterface. Refer to the appropriate Cisco IOS XR configuration guide for other restrictions.

Untagged L2 Subinterface

Table 8: Feature History Table

Feature Name	Release Information	Feature Description

Untagged L2 Subinterface	Release 24.2.11	You can now use untagged L2 subinterfaces to effectively manage and process traffic from customer edge (CE) devices that do not employ VLAN tagging. This capability allows you to apply services to untagged packets, which would not have been possible if the packets were to be logically received on the main interface. As a result, you can now push a dot1q or other supported Layer 2 encapsulation on the received frame. This feature introduces the **encapsulation untagged** command.

Untagged L2 subinterface

An untagged L2 subinterface is a subinterface on a network device that isn't linked to a VLAN tag. The VLAN tags divide network traffic into smaller, distinct networks, which helps improve security and manageability in a bigger network setup.

Challenges in Processing Traffic without Untagged L2 Subinterfaces

You cannot perform the following actions on an untagged frame received on the main interface:

- Apply commands available only to l2transport subinterfaces, such as popping or pushing a dot1q or dot1ad header.
- Attach a service policy because you cannot simultaneously attach a service policy to the main interface and subinterfaces of that interface.

Untagged L2 Subinterface Offerings

The untagged subinterface takes a higher priority than the main interface on service mapping. When both interfaces are created, the untagged traffic is mapped to the untagged subinterface.

Untagged Traffic Mapped to an Untagged Subinterface

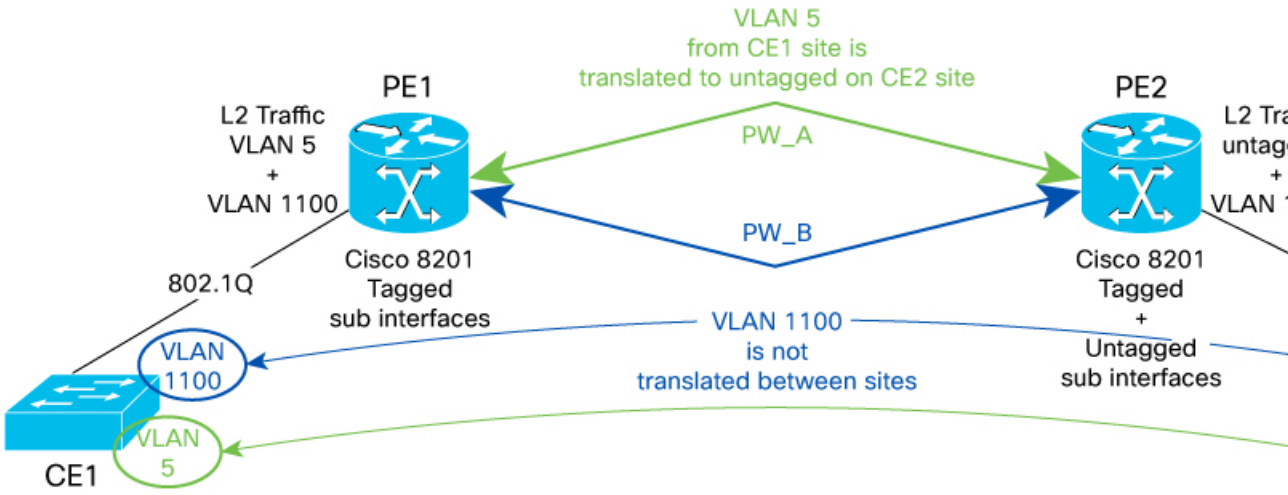
Traffic using an unsupported VLAN tag format is considered untagged traffic and mapped to an untagged subinterface. The following VLAN tag formats are supported:

- Single tag format - dot1q
- Double tag format - outer dot1ad paired with inner dot1q tag, or outer dot1q tag paired with inner dot1q tag
- Three or more tags - outermost dot1ad tag followed by dot1q tag, or outermost dot1q tag followed by dot1q tag

Benefits of Untagged L2 Subinterfaces

- Manages untagged and tagged traffic on the same port to different services.
- Allows rewriting of different VLANs on untagged subinterfaces for both tagged and untagged traffic.

How does the Untagged L2 subinterface Work?



Let's understand the feature using this sample topology, in which the P2P L2VPN service is deployed between the two CE devices.

In this topology,

- VLAN 1100 is a common VLAN that is shared between CE1 and CE2.

  PE nodes do not require VLAN translation. However, if a VLAN ID is not common on CE1 and CE2, PE must perform VLAN translation.

- For instance, VLAN 5 is known only to CE1, and for the same service, CE1 uses VLAN 5, while CE2 uses no VLAN. In this situation, PE1 performs VLAN translation.

- When traffic is sent from CE1 to CE2, PE1 removes the VLAN 5 tag on the traffic from CE1 and then sends untagged traffic to PE2 through the pseudowire (PW). PE2 disposes the PW header and forwards untagged traffic to CE2.

- When a traffic is sent from CE2 to CE1, PE2 maps untagged traffic from CE2 to an L2 subinterface and sends the traffic to PE1 through the PW. PE1 disposes of the PW header and adds the VLAN 5 tag before switching the traffic to CE1.

Configure Untagged L2 Subinterface

To manage traffic efficiently, configure untagged L2 subinterface.

Procedure

Step 1 Create an untagged L2 subinterface with the **encapsulation untagged** keyword under an L2 transport main interface.

This feautre is only applicable to L2 transport main and subinterfaces and does not support any other encapsulation types on the same L2 subinterface.

Example:

```
Router# configure
Router(config)# interface HundredGigE0/0/16.1500 l2transport
```

```
Router(config-subif)# encapsulation untagged
Router(config-subif)# commit
```

Step 2 Verify that the untagged encapsulation is configured on the L2 subinterface.

Example:

```
Router# show interfaces HundredGigE0/0/16.1500
HundredGigE0/0/16.1500 is up, line protocol is up
Interface state transitions: 1
Hardware is VLAN sub-interface(s), address is 0029.c201.3f0c
Internet address is 40.40.50.1/24
MTU 1522 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation Untagged Virtual LAN,
Last link flapped 00:01:25
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
```

Enhanced Performance Monitoring for Layer 2 Subinterfaces (EFPs)

Beginning in Cisco IOS XR Release 7.2.12, the router adds support for basic counters for performance monitoring on Layer 2 subinterfaces. This section provides a summary of the new support for Layer 2 interface counters.

The **interface basic-counters** keyword has been added to support a new entity for performance statistics collection and display on Layer 2 interfaces in the following commands:

- **performance-mgmt statistics interface basic-counters**
- **performance-mgmt threshold interface basic-counters**
- **performance-mgmt apply statistics interface basic-counters**
- **performance-mgmt apply threshold interface basic-counters**
- **performance-mgmt apply monitor interface basic-counters**
- show performance-mgmt monitor interface basic-counters
- show performance-mgmt statistics interface basic-counters

The **performance-mgmt threshold interface basic-counters** command supports the following attribute values for Layer 2 statistics, which also appear in the **show performance-mgmt statistics interface basic-counters** and **show performance-mgmt monitor interface basic-counters** command:

Attribute	Description
InOctets	Bytes received (64-bit)

Attribute	Description
InPackets	Packets received (64-bit)
InputQueueDrops	Input queue drops (64-bit)
InputTotalDrops	Inbound correct packets discarded (64-bit)
InputTotalErrors	Inbound incorrect packets discarded (64-bit)
OutOctets	Bytes sent (64-bit)
OutPackets	Packets sent (64-bit)
OutputQueueDrops	Output queue drops (64-bit)
OutputTotalDrops	Outband correct packets discarded (64-bit)
OutputTotalErrors	Outband incorrect packets discarded (64-bit)

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 signal 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

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
LLDP Snooping	Release 7.3.3	With this release, you can further leverage the Link Layer Discovery Protocol (LLDP) information for directly attached devices or equipment in an L2 (Layer 2) network via LLDP snoop. In order to utilize the LLDP snoop functionality, the neighbouring devices must exchange the LLDP packets with the L2 network. With the help of the LLDP snoop functionality, you can identify the cabling and modeling failures and isolate faults. To enable LLDP snoop, enable the LLDP command on an interface while the outgoing (TX) traffic is disabled.

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.

Note MAcsec encrypts LLDP packets by default. You can enable exceptions in the MACsec policy using the **allow lldp-in-clear** command to retain the LLDP packets unencrypted with MACsec. For more information, see *MACsec Policy Exceptions for Link Layer Discovery Protocol Packets* section in the *Configuring MACsec* chapter of the *System Security Configuration Guide for Cisco 8000 Series Routers*.

Note For LLDP to work on any bundle member, enable LLDP on the bundle main interface either globally or on the interface itself. You can then choose to disable LLDP transmission on bundle main interface by using the `lldp transmit disable` command.

You can also control LLDP transmit or receive on each bundle member interface as desired.

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)

Specifying User-Defined LLDP TLV Values

It is possible to override the system default values for some of the mandatory LLDP Type-Length-Values (TLVs) that are advertised by routers to their directly connected neighboring devices. While advertising their identity and capabilities, routers can assign user-defined meaningful names instead of autogenerated values. Using the following CLIs you can specify these user-defined values:

- `Router(config)#lldp system-name` *`system-name`*
- `Router(config)#lldp system-description` *`system-description`*

- `Router(config)#lldp chassis-id-type chassis-type`
- `Router(config)#lldp chassis-id local-chassis-id`

Note The **chassis-id** value is configurable only when the **chassis-id-type** is set as **Local**. If there is a mismatch, you encounter a configuration failed error message.

The configured values, such as the system name, system description, chassis-id, chassis-type become part of the TLV in the LLDP packets that are sent to its neighbors. Values are transmitted only to LLDP enabled interfaces to which the router is connected.

You can assign any of the following values for the `chassis-id-type`. The chassis-id-types are objects that are part of the management information base (MIB). Depending on the selected chassis-id-type, values are assigned to these objects, and they are advertised by the router to its neighboring devices.

chassis-id-type	Description
chassis-component	Chassis identifier based on the value of entPhysicalAlias object that is defined in IETF RFC 2737.
interface-alias	Chassis identifier based on the value of ifAlias object as defined in IETF RFC 2863.
interface-name	Chassis identifier based on the name of the interface.
local	Chassis identifier based on a locally defined value.
mac-address	Chassis identifier based on the value of a unicast source address.
network-address	Chassis identifier based on a network address that is associated with a particular chassis.
port-component	Chassis identifier based on the value of entPhysicalAlias object defined in IETF RFC 2737 for a port or backplane component.

Tip You can programmatically modify default values of LLDP TLVs by using the `openconfig-lldp` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

Configuration Example

This example shows the configuration for the LLDP TLVs that will be advertised by routers to their directly connected neighboring devices.

```
Router(config)#lldp system-name cisco-xr
Router(config)#lldp system-description cisco-xr-edge-device
```

```
Router(config)#lldp chassis-id-type local
Router(config)#lldp chassis-id ce-device9
```

Running Configuration

```
Router#show lldp
Tue Sep 13 16:03:44.550 +0530
Global LLDP information:
Status: ACTIVE
LLDP Chassis ID: ce-device9
LLDP Chassis ID Subtype: Locally Assigned Chassis Subtype
LLDP System Name: cisco-xr
LLDP advertisements are sent every 30 seconds
LLDP hold time advertised is 120 seconds
LLDP interface reinitialisation delay is 2 seconds
```

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 global 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.
- LLDP snoop is supported on L2 interfaces, when the incoming (RX) traffic is enabled and outgoing (TX) traffic is disabled.

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.

Setting the Carrier Delay on Physical Interfaces

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
Default Carrier Delay Value on Physical Interfaces	Release 24.2.11	We have introduced the carrier-delay up default value to ensure enough time to establish a stable hardware link state. If you haven't configured the timer, the default carrier delay automatically delays the hardware link-up notifications by 200 ms. Previously, we recommended that you set the carrier delay-up timer to 10 ms. If you want to change the delay of the interface state change notification, you can use the **carrier-delay** command to set a different value.
Setting the Carrier Delay on Physical Interfaces	Release 7.5.4	You can configure the Ethernet interfaces to delay the processing of hardware link-down and link-up notifications. With this functionality, the interface state remains stable for the configured delay duration, even if the hardware link state fluctuates. This prevents interface flapping and improves network reliability. Use the following CLI command in interface configuration mode to configure the delay time: **carrier-delay**

Hardware links take time to stabilize after a state change and may experience link flaps. Link flap is a condition where a physical interface frequently fluctuates between an up and a down state.

During link flaps, the network reestablishes and updates routing paths after a disruption, which leads to resource exhaustion on routers. To overcome the problem, we recommend waiting until the link state is stable before taking action.

The carrier delay introduces a delay in processing interface link-state notifications in the router to provide enough time for the interface link to stabilize.

When there is a change in the link state, the carrier-delay timer starts. If the link state goes up, the **carrier-delay up** timer starts. Similarly, when the link state goes down, the **carrier-delay down** timer starts. During this delay period, the Ethernet interface state remains unchanged even if the link is physically restored. Setting a delay timer ensures the link state is established before the interface becomes operational again and avoids unnecessary interface state changes and associated traffic rerouting.

Guidelines and Restrictions for Setting the Carrier Delay on Physical Interfaces

The following usage guidelines and restrictions are applicable for setting the carrier delay on physical interfaces:

- You can configure carrier-delay for only link-up, only link-down, or both link-up and link-down notifications.
- If the **carrier-delay down** *milliseconds* command is configured on a physical link that fails and cannot be recovered, link down detection time increases, and it may take longer for the routing protocols to reroute the traffic around the failed link.
- Loss of Signal (LOS) is not supported on carrier delay.
- From Release 24.2.11, the default value of carrier-delay up parameter is changed to 200 ms. To restore your original configuration, you need to configure the parameter explicitly using **carrier-delay** command.
- If not configured, the carrier-delay up parameter defaults to 200 ms and the carrier-delay down parameter to 0 ms. When carrier-delay down is not configured, the higher-layer protocols are notified immediately when a physical link state changes.
- The **carrier-delay** command overwrites the previous configuration every time you execute the command. For example, if you already have the **carrier-delay up** configured and later configure the **carrier-delay down**, the carrier-delay down overwrites the previously configured carrier-delay up. However, you can configure the carrier delay up and down concurrently using the **carrier-delay up** (milliseconds) **down** (milliseconds) command.

Configure the Carrier-delay Timer

Default Configuration Example

In this example, one interface is brought up to check the default value of link-up notification delay.

```
Router#configure
Router(config)#interface HundredGigE 0/0/0/0
Router(config-if)#no shutdown
Router(config-if)#commit
```

Run the **show interfaces** command to check if the carrier-delay configuration for the interface defaults to 200 ms.

```
Router#show interfaces HundredGigE 0/0/0/0 | include Carrier
Fri Mar 31 07:25:05.273 UTC
Carrier delay (up) is 200 msec
```

Configuration Example

In this example, link-up and link-down notifications are configured to be delayed by 1000 ms and 150 ms using **carrier-delay** command.

```
Router#configure
Router(config)#interface HundredGigE 0/0/0/0
Router(config-if)#carrier-delay up 1000 down 150
Router(config-if)#commit
```

Running Configuration

```
 interface HundredGigE0/0/0/0
 carrier-delay up 1000 down 150
!
```

Verification

Run the **show interfaces** command to see the current state of the carrier-delay configuration for an interface.

```
Router#show interfaces HundredGigE 0/0/0/0 | include Carrier
Fri Mar 31 07:25:05.273 UTC
Carrier delay (up) is 1000 msec, Carrier delay (down) is 150 msec
```

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*.

Note For LLDP to work on any bundle member, enable LLDP on the bundle main interface either globally or on the interface itself. You can then choose to disable LLDP transmission on bundle main interface by using the `lldp transmit disable` command.

You can also control LLDP transmit or receive on each bundle member interface as desired.

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

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

To enable LLDP globally, complete the following steps:

1. `RP/0//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:

Procedure

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:

Procedure

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:

Procedure

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.

Command	Description
show lldp neighbors	Displays information about LLDP neighbors.
show lldp traffic	Displays statistics for LLDP traffic.

To collect or clear LLDP interface statistics, you can use the following commands:

Command	Description
show lldp traffic interface *interface_name*	Displays LLDP traffic statistics for the specified interface.
clear lldp counters interface *interface_name*	Clears LLDP traffic statistics for the specified interface. Global statistics remains intact. Similarly, clearing global statistics does not impact the interface statistics.

Examples for LLDP Interface Statistics

This example shows interface statistics for **gigabitEthernet0/0/0/0**:

```
Router#show lldp traffic interface gigabitEthernet0/0/0/0
```

This example clears the interface statistics for **gigabitEthernet0/0/0/0**.

```
Router#show lldp traffic interface gigabitEthernet0/0/0/0
```

Running Configuration

```
Router#show lldp traffic interface gigabitEthernet 0/2/0/8
Wed Aug 24 17:38:11.829 IST

LLDP Interface statistics:
        Total frames out: 28786
        Total frames in: 38417
        Total frames received in error: 0
        Total frames out error: 0
        Total frames discarded: 0
        Total TLVs discarded: 0
        Total TLVs unrecognized: 0
```

Configuring LLDP Snoop

If you have LLDP enabled on all Ethernet interfaces, the system enables Link Layer Discovery Protocol (LLDP) snoop on all L2 interfaces by default. You can use LLDP snooping to troubleshoot problems at the client ports.

Note LLDP snoop is enabled only when LLDP RX is enabled and LLDP TX (transmit) is disabled either on interface or global LLDP configuration.

To enable LLDP snoop on an L2 interface, perform the following steps:

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

```
RP/0/RSP0/CPU0:ios(config)# interface FourHundredGigE 0/0/0/5

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

RP/0/RSP0/CPU0:ios(config-if)#enable

RP/0/RSP0/CPU0:ios(config-if)#transmit disable

RP/0/RSP0/CPU0:ios(config-if)#commit
```

Running Configuration

```
RP/0/RP0/CPU0:router#show run
Fri Jan 21 17:45:17.529 UTC
Building configuration...
!! IOS XR Configuration 7.7.1.06I
!! Last configuration change at Fri Jan 21 17:20:27 2022 by cisco
!
hostname router1
logging console disable
username xxxx
 group root-lr
 group cisco-support
 secret 10
$6$JELNK0oJaZZN7K0.$8YmyRWkq3D92i.lJc5QSdDkq4kUjU.g9U7sYIIAV1QVnSBemng5q.5EyYv6xSL9niDxRmKaFEATs9BkitDqpr.
!
line console
 exec-timeout 0 0
 absolute-timeout 0
 session-timeout 0
!
line default
 exec-timeout 0 0
 absolute-timeout 0
 session-timeout 0
!
vty-pool default 0 99 line-template default
call-home
 service active
 contact smart-licensing
 profile CiscoTAC-1
  active
  destination transport-method email disable
  destination transport-method http
 !
!
interface MgmtEth0/RP0/CPU0/0
 shutdown
!
interface FourHundredGigE0/0/0/0
 lldp
  enable
  transmit disable
 !
 l2transport
 !
!
interface FourHundredGigE0/0/0/1
 shutdown
!
interface FourHundredGigE0/0/0/2
 shutdown
!
interface FourHundredGigE0/0/0/3
```

```
 shutdown
!
interface FourHundredGigE0/0/0/4
 shutdown
!
interface FourHundredGigE0/0/0/5
 lldp
  enable
  transmit disable
 !
 l2transport
 !
!
interface FourHundredGigE0/0/0/6
 shutdown
!
interface FourHundredGigE0/0/0/7
 shutdown
!
interface FourHundredGigE0/0/0/8
 shutdown
!
interface FourHundredGigE0/0/0/9
 shutdown
!
interface FourHundredGigE0/0/0/10
 shutdown
!
interface FourHundredGigE0/0/0/11
 shutdown
!
interface FourHundredGigE0/0/0/12
 shutdown
!
interface FourHundredGigE0/0/0/13
 shutdown
!
interface FourHundredGigE0/0/0/14
 shutdown
!
interface FourHundredGigE0/0/0/15
 shutdown
!
interface FourHundredGigE0/0/0/16
 shutdown
!
interface FourHundredGigE0/0/0/17
 shutdown
!
interface FourHundredGigE0/0/0/18
 shutdown
!
interface FourHundredGigE0/0/0/19
 shutdown
!
interface FourHundredGigE0/0/0/20
 shutdown
!
interface FourHundredGigE0/0/0/21
 shutdown
!
interface FourHundredGigE0/0/0/22
 shutdown
!
```

```
interface FourHundredGigE0/0/0/23
 shutdown
!
interface HundredGigE0/0/0/24
 shutdown
!
interface HundredGigE0/0/0/25
 shutdown
!
interface HundredGigE0/0/0/26
 shutdown
!
interface HundredGigE0/0/0/27
 shutdown
!
interface HundredGigE0/0/0/28
 shutdown
!
interface HundredGigE0/0/0/29
 shutdown
!
interface HundredGigE0/0/0/30
 shutdown
!
interface HundredGigE0/0/0/31
 shutdown
!
interface HundredGigE0/0/0/32
 shutdown
!
interface HundredGigE0/0/0/33
 shutdown
!
interface HundredGigE0/0/0/34
 shutdown
!
interface HundredGigE0/0/0/35
 shutdown
!
l2vpn
 bridge group bg1
  bridge-domain bd1
   interface FourHundredGigE0/0/0/0
   !
   interface FourHundredGigE0/0/0/5
   !
  !
 !
!
end

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

Verification

```
router0 <———> router1 <———-> router2
          0/0/0/0           0/0/0/0/5

RP/0/RP0/CPU0:router0#config
Fri Jan 21 17:16:41.713 UTC
RP/0/RP0/CPU0:router0(config)#lldp
RP/0/RP0/CPU0:router0(config-lldp)#exit
RP/0/RP0/CPU0:router0(config)#int hu 0/0/0/0
RP/0/RP0/CPU0:router0(config-if)#no shut
```

```
RP/0/RP0/CPU0:router0(config-if)#end
Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:yes

RP/0/RP0/CPU0:router1#config
Fri Jan 21 17:17:41.459 UTC
RP/0/RP0/CPU0:router1(config)#int FourHundredGigE 0/0/0/0
RP/0/RP0/CPU0:router1(config-if)#no shut
RP/0/RP0/CPU0:router1(config-if)#l2transport
RP/0/RP0/CPU0:router1(config-if-l2)#exit
RP/0/RP0/CPU0:router1(config-if)#lldp
RP/0/RP0/CPU0:router1(config-lldp)#enable
RP/0/RP0/CPU0:router1(config-lldp)#transmit disable
RP/0/RP0/CPU0:router1(config-lldp)#exit
RP/0/RP0/CPU0:router1(config-if)#exit
RP/0/RP0/CPU0:router1(config)#int FourHundredGigE 0/0/0/5
RP/0/RP0/CPU0:router1(config-if)#no shut
RP/0/RP0/CPU0:router1(config-if)#l2transport
RP/0/RP0/CPU0:router1(config-if-l2)#exit
RP/0/RP0/CPU0:router1(config-if)#lldp
RP/0/RP0/CPU0:router1(config-lldp)#enable
RP/0/RP0/CPU0:router1(config-lldp)#transmit disable
RP/0/RP0/CPU0:router1(config-lldp)#exit
RP/0/RP0/CPU0:router1(config-if)#exit
RP/0/RP0/CPU0:router1(config)#l2vpn bridge group bg1
RP/0/RP0/CPU0:router1(config-l2vpn-bg)#bridge-domain bd1
RP/0/RP0/CPU0:router1(config-l2vpn-bg-bd)#interface FourHundredGigE 0/0/0/0
RP/0/RP0/CPU0:router1(config-l2vpn-bg-bd-ac)#exit
RP/0/RP0/CPU0:router1(config-l2vpn-bg-bd)#interface FourHundredGigE 0/0/0/5
RP/0/RP0/CPU0:router1(config-l2vpn-bg-bd-ac)#end
Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:yes


RP/0/RP0/CPU0:router0#config
Fri Jan 21 17:16:41.713 UTC
RP/0/RP0/CPU0:router0(config)#lldp
RP/0/RP0/CPU0:router0(config-lldp)#exit
RP/0/RP0/CPU0:router0(config)#int hu 0/0/0/0
RP/0/RP0/CPU0:router0(config-if)#no shut
RP/0/RP0/CPU0:router0(config-if)#end
Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:yes


RP/0/RP0/CPU0:router0#sh lldp neighbors
Fri Jan 21 17:21:15.857 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
router2            HundredGigE0/0/0/0             120        R
FourHundredGigE0/0/0/5

Total entries displayed: 1

RP/0/RP0/CPU0:router0#

RP/0/RP0/CPU0:router0#sh lldp neighbors
Fri Jan 21 17:21:15.857 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
router2            HundredGigE0/0/0/0             120        R
```

```
FourHundredGigE0/0/0/5

Total entries displayed: 1

RP/0/RP0/CPU0:router0#

RP/0/RP0/CPU0:router2#sh lldp neighbors
Fri Jan 21 17:21:20.998 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
router0            FourHundredGigE0/0/0/5          120        R
HundredGigE0/0/0/0

Total entries displayed: 1

RP/0/RP0/CPU0:router2#


RP/0/RP0/CPU0:router1#show controllers npu stats traps-all instance all location all | inc
 LLDP
Fri Jan 21 17:24:07.964 UTC
LLDP                                          0    22   RPLC_CPU    206   1538  6    4000
      3975       IFG     1520    0                    0
LLDP_SNOOP                                    0    28   RPLC_CPU    206   1538  6    4000
      3862       NPU     N/A     16                   0
RP/0/RP0/CPU0:router1#
```

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 a Layer 2 VPN AC: Example

The following example indicates how to configure a Layer 2 VPN AC on an Ethernet interface:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/0/0/2
RP/0/RSP0/CPU0:router(config-if)# l2transport
RP/0/RSP0/CPU0:router(config-if-l2)# l2protocol  tunnel
RP/0/RSP0/CPU0:router(config-if-l2)# commit
```

Configuring Physical Ethernet Interfaces

Use this procedure to create a basic Ethernet interface configuration.

Procedure

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
!
```

Viewing Interface Counters Report

The Interface Counters report summarizes the statistics for all interfaces configured on the router.

The report displays the interfaces configured, the input and output rate, the total number of packets transmitted and received, the time interval, the current status of each interface, and the packet counts for input and output broadcast, multicast, and errored packets.

The **show interfaces** command, displays statistics per interface with many lines of data. The traffic rate displays the average number of packets received per second over the load interval. The load interval is configurable on the physical and bundle main interface. The report displays the load on the interface for a longer duration of time and does not show a spike in the traffic rate. This rate is the exponentially weighted average with a time constant of the load interval.

Note For the average to be within two percent of the instantaneous rate of a uniform stream of traffic, four times the load interval must pass.

For more information about the use of **show interfaces** command, see *Interface and Hardware Component Command Reference for Cisco 8000 Series Routers*.

Instant Display of Traffic Rates for all the Physical Interfaces

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
Instant display of traffic rates on all physical interfaces	Release 7.5.4	You can now display a snapshot of the traffic throughput and traffic rate on all physical interfaces over the last few seconds. We have introduced a **show** command to view the counters and rate information for the interfaces. The feature introduces these: • CLI: **show interfaces counter rates physical** • YANG Data Model: **Cisco-IOS-XR-infra-statsd-oper.yang** (see GitHub under the 754 folder.)

The new **show** command displays a snapshot of statistics for all the interfaces at a given instant for your quick reference. Here, the display is in a tabular format for easy analysis.

Run the **show interfaces counter rates physical** command to view statistics of all physical interfaces.

View the statistics

```
Router#show interfaces counters rates physical

    InterfaceName           Intvl   InMbps   InBW%  InKpps  OutMbps  OutBW%   OutKpps
    GigabitEthernet0/2/0/0  0:05     0.0     0.0%    0.0       0.0     0.0%       0.0
    GigabitEthernet0/2/0/1  0:05     0.0     0.0%    0.0       0.0     0.0%       0.0
    GigabitEthernet0/2/0/2  0:05     0.0     0.0%    0.0       0.0     0.0%       0.0
    GigabitEthernet0/2/0/3  0:05   235.0    22.0%   23.5      87.0     9.5%       7.2
    GigabitEthernet0/3/0/0  0:05    88.0     9.3%    7.0     100.0    10.0%      10.5
    GigabitEthernet0/3/0/1  0:05     0.0     0.0%    0.0       0.0     0.0%       0.0
```

The statistics for each physical interface is calculated for the time interval of 5 sec. Hence, the input and output rate (in Mbps and Kpps) is the real-time statistics.

Note The traffic rate displayed is the real-time link utilization of the time interval. The time interval is determined by the system and may vary based on the system processing load. The time interval increases during events where the system is handling, for example, performing routing updates.

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 you're using a Q100-based Cisco 8200 Series Router and want to set up a 4x10G breakout configuration, you need to use even numbered ports from 24 to 35. These include ports 24, 26, 28, 30, 32, and 34. Once you do this, the system automatically disables the odd numbered ports in this range - ports 25, 27, 29, 31, 33, and 35.
- 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.
- For the 88-LC0-34H14FH line card, you must breakout only 3 ports instead of 4 ports to avoid the `QOS-DPA_QOSEA-2-TMPORT_PROG_ERROR` issue, which creates partial interfaces during configuration mode.

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

.
```

Ethernet Interface Route Statistics

Table 12: Feature History Table

Feature Name	Release Information	Feature Description
Ethernet Interface Route Statistics	Release 7.3.4	You can view statistics on the number of packets and bytes sent and received in unicast, multicast, and broadcast routes. These statistics help you to monitor the network performance and measure your bandwidth.

Ethernet interface route statistics provide the following information about the unicast, multicast, and broadcast routes:

- The number of packets or bytes received and transmitted.
- The total number of packets or bytes passing through the Ethernet interface.

These statistics are available on Cisco 8000 routers that are built with Cisco Silicon One Q100 and Q200 processors and all Network Processor Unit (NPU) 2.0 devices.

Ethernet interface route statistics may be useful for monitoring the network devices and their traffic. For example, if you are not able to connect to the internet or use some cloud-based applications, these route statistics can help you understand the problems in the network and where they occur.

Viewing the Interface Statistics

Use the `show interface` and the `show controller interface` commands to view these Ethernet interface route statistics. The following is a sample showing both commands.

```
Router#show interfaces HundredGigE 0/0/0/0

<Timestamp>

HundredGigE0/0/0/0 is up, line protocol is up

  Interface state transitions: 93

  Hardware is HundredGigE, address is acbc.d975.0500 (bia acbc.d975.0500)

  Internet address is 100.0.1.1/24

  MTU 1514 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)

     reliability 255/255, txload 0/255, rxload 0/255

  Encapsulation ARPA,

  Full-duplex, 100000Mb/s, 100GBASE-SR4, 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 01:03:01

  ARP type ARPA, ARP timeout 04:00:00

  Last input 3d09h, output 01:02:41

  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
```

```
      2959131434 packets input, 757537646912 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

      2958525230 packets output, 757382468319 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

      93 carrier transitions
```

```
Router#show controllers HundredGigE0/0/0/0 stats

<Timestamp>

Statistics for interface HundredGigE0/0/0/0 (cached values):



Ingress:

    Input total bytes           = 757537646912

    Input good bytes            = 757537646912


    Input total packets         = 2959131434
    Input 802.1Q frames         = 0
    Input pause frames          = 0
    Input pkts 64 bytes         = 1
    Input pkts 65-127 bytes     = 0
    Input pkts 128-255 bytes    = 0
    Input pkts 256-511 bytes    = 2959131433
    Input pkts 512-1023 bytes   = 0
    Input pkts 1024-1518 bytes  = 0
    Input pkts 1519-Max bytes   = 0

    Input good pkts             = 2959131434
```

```
    Input unicast pkts          = 0
    Input multicast pkts        = 0
    Input broadcast pkts        = 0

    Input drop overrun          = 0
    Input drop abort            = 0
    Input drop invalid VLAN     = 0
    Input drop invalid DMAC     = 0
    Input drop invalid encap    = 0
    Input drop other            = 0

    Input error giant           = 0
    Input error runt            = 0
    Input error jabbers         = 0
    Input error fragments       = 0
    Input error CRC             = 0
    Input error collisions      = 0
    Input error symbol          = 0
    Input error other           = 0

    Input MIB giant             = 0
    Input MIB jabber            = 0
    Input MIB CRC               = 0

Egress:
    Output total bytes          = 757382468319
    Output good bytes           = 757382468319

    Output total packets        = 2958525230
    Output 802.1Q frames        = 0
    Output pause frames         = 0
    Output pkts 64 bytes        = 41
    Output pkts 65-127 bytes    = 296
    Output pkts 128-255 bytes   = 746
    Output pkts 256-511 bytes   = 2958524147
    Output pkts 512-1023 bytes  = 0
    Output pkts 1024-1518 bytes = 0
    Output pkts 1519-Max bytes  = 0

    Output good pkts            = 2958525230
    Output unicast pkts         = 0
    Output multicast pkts       = 0
    Output broadcast pkts       = 0

    Output drop underrun        = 0
    Output drop abort           = 0
    Output drop other           = 0

    Output error other          = 0
```