Interfaces and Hardware Component Configuration Guide for Cisco 8000 Series Routers, Cisco IOS XR Releases

Cisco 8000 Series Routers
Updated July 31, 2026

Topics included

Communications, Services, and Additional Information

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

Cisco Bug Search Tool

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

1 YANG Data Models for Interfaces and Hardware Component

Topics:

This chapter provides information about the YANG data models for interfaces and hardware component features.

Using YANG Data Models

You will learn about the YANG data models for interface and hardware component features.

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using YANG data models. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

The data models are available in the release image, and are also published in the Github repository. Navigate to the release folder of interest to view the list of supported data models and their definitions. Each data model defines a complete and cohesive model, or augments an existing data model with additional XPaths. To view a comprehensive list of the data models supported in a release, navigate to the *Available-Content.md* file in the repository.

You can also view the data model definitions using the YANG Data Models Navigator tool. This GUI-based and easy-to-use tool helps you explore the nuances of the data model and view the dependencies between various containers in the model. You can view the list of models supported across Cisco IOS XR releases and platforms, locate a specific model, view the containers and their respective lists, leaves, and leaf lists presented visually in a tree structure. This visual tree form helps you get insights into nodes that can help you automate your network.

To get started with using the data models, see the *Programmability Configuration Guide*.

2 Physical interfaces preconfiguration

Topics:

This chapter provides information on how to preconfigure physical interfaces on Cisco routers, including benefits, guidelines, and step-by-step configuration procedures.

Interface preconfigurations

Explains what physical interface preconfiguration is, what it requires, and how the router applies preconfigured settings when hardware is installed.

Physical interface preconfiguration is the process of configuring interfaces before they are present in the router.

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

Preconfiguration enables you to:

- configure line cards before you install them in the router,
- upon installation, the router automatically applies the preconfigured settings, and
- stores this information in a dedicated database tree on the Route Processor, known as the preconfiguration directory, which is separate from the standard interface configuration.

The router supports preconfiguration for these interfaces:

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

Some preconfiguration data requires the presence of a line card for verification because the verification processes run exclusively on the line card. You can verify this data after inserting the line card and initiating the verification process. The router rejects the configuration if it detects errors during the transfer from the preconfiguration area to the active configuration.

Benefits of physical interface preconfiguration

Introduces benefits of physical interface preconfiguration and explains the core idea and technical context. Highlights the main characteristics and usage guidance needed before configuration.

These are the benefits of physical interface preconfiguration: are benefits of physical interface preconfiguration topics that explain .

- preconfigurations reduce downtime when you add new cards to the router.
- during a card replacement, when you remove the line card, you can still see the previous configuration and make modifications.

Configuration guidelines for physical interface preconfiguration

Provides configuration guidelines for physical interface preconfiguration, including verification, dual-Route Processor behavior, and saving preconfigured changes.

These guidelines apply for physical interface preconfiguration:

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

- We recommend filling out preconfiguration information in your site planning guide. This allows you to compare the anticipated configuration with the actual preconfigured interfaces when you install the line card and the interfaces are up.
- You do not need to configure anything to guarantee that the standby interface configurations are maintained.

Active and standby RPs and virtual interface guidelines

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

- failure is detected by a watchdog
- it is administratively commanded to take over
- the active RP is removed from the router

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

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

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

In a dual RP system, the interface state sync library monitors the state between active and standby RPs. An alarm is raised when a port is down on one RP and the same port is up on the other RP.

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

Clear the alarm when there is no inconsistency between both RP port states.

If behavior deviates, review the preconfiguration, verify interface state on both RPs, and align the configuration with this principle.

Restrictions for physical interface preconfiguration

This topic summarizes restrictions for physical interface preconfiguration that affect planning, validation, and execution.

These restrictions apply for physical interface preconfiguration:

- You must provide names during preconfiguration that matches with the name of the interface that is created. If the interface names do not match, the system does not apply preconfiguration when the interface is created.
- The interface names must begin with the interface type that is supported by the router and for which drivers have been installed. However, the slot, port, subinterface number, and channel interface number information cannot be validated.
- It is possible that you are not able to verify some configurations until you insert the line card is inserted.

- Specifying an interface name that already exists and is configured , or an abbreviated name like Hu0/3/0/0 is not permitted.
- Do not enter the **no shutdown** command for new preconfigured interfaces, because the no form of this command removes the existing configuration, and there is no existing configuration.

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

Preconfigure a physical interface

Provides the steps to preconfigure a physical interface and verify the resulting running configuration.

Use this task for the most basic preconfiguration of an interface that is not yet present in the router.

Before you begin

Before preconfiguring physical interfaces, ensure that you meet the following conditions:

- Preconfiguration drivers and files are installed. Although it might be possible to preconfigure physical interfaces without a preconfiguration driver installed.
- Ensure the preconfiguration files are configured to define the interface definition file, as this file is required to supply the strings for valid interface names.

Procedure

1. Use the **interface preconfigure** command in global configuration mode to preconfigure interfaces that are not yet present in the system.

   The command places the router in interface configuration mode, where you can add the supported interface commands for the preconfigured interface. The verifiers registered for the preconfigured interfaces verify the configuration. The preconfiguration is complete when you enter the **end** command, or any matching exit or global configuration mode command.

2. Enter global configuration mode.

   Example

   ```
   Router:hostname# configure
   ```

3. Enter interface preconfiguration mode for an interface by using the **interface preconfigure** *type interface-path-id* command, where *type* specifies the supported interface type that you want to configure and *interface-path-id* specifies the location where the interface is located in rack/slot/module/port notation.

   Example

   ```
   Router(config)# interface preconfigure HundredGigE 0/3/0/2
   ```

4. Assign an IP address and mask to the interface. Use one of the following commands:
   - **ipv4 address** *ip-address subnet-mask*

- **ipv4 address** *ip-address/prefix*

Example

```
Router(config-if-pre)# ipv4 address 192.168.1.2/31
```

5. End or save your changes.

Example

```
Router(config-if-pre)# end
or
Router(config-if-pre)# commit
```

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

3 Management ethernet interface

Topics:

This chapter covers Management Ethernet interface configuration, a network management process enabling secure router access and out-of-band traffic control.

Management ethernet interface configuration

Explains what Management Ethernet interface configuration is, when you need it, and which management protocols rely on the dedicated interface.

Management Ethernet interface configuration is the setup of the dedicated router management interface so you can reach the router through the LAN and use out-of-band management protocols.

Additional details

To configure Management Ethernet interfaces, understand how the dedicated management interface provides router access and carries management traffic.

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

Note

By default, Management Ethernet interfaces are present on the router. Configure these interfaces to:

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

Requirements for configuring Management Ethernet interfaces

This topic lists the prerequisites and conditions you must meet before configuring a Management Ethernet interface.

Before you perform the Management Ethernet interface configuration procedure, ensure that you meet these requirements:

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

Note

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

Default Management Ethernet interface settings

This topic lists the default Management Ethernet interface settings that affect configuration and verification.

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

Table 1: Management Ethernet interface default settings

Parameter	Default value	Configuration file entry
Speed in Mbps	Default speed is 1G with autonegotiation enabled.	Speed is not configurable.
Duplex mode	Default duplex mode is **full-duplex** with autonegotiation enabled.	Duplex mode is not configurable.
MAC address	MAC address comes from the hardware burned-in address (BIA).	MAC address is not configurable.

Use these defaults as a baseline when you compare manual changes with the values that the system applies automatically.

Configure a Management Ethernet interface

This topic provides the steps to configure a Management Ethernet interface, save the changes, and verify the resulting operational state.

Configure the Management Ethernet interface with an IPv4 address and subnet mask, optionally adjust the MTU, and bring the interface online for management traffic.

Use this procedure for the minimal configuration that is required on the Management Ethernet interface.

Replace *ip-address* with the primary IPv4 address for the interface.

Specify the subnet mask as either a dotted-decimal mask, such as `255.255.0.0`, or a prefix length, such as `/16`.

Before you begin

Complete the initial Management Ethernet interface setup and make sure that you understand the generalized interface naming format *rack/slot/module/port*.

Procedure

1. Enter the configuration mode.

   Example

   ```
   Router# configure
   ```

2. Specify the Ethernet interface name and notation as, `rack/slot/module/port`.

   Example

   ```
   Router(config)# interface MgmtEth 0/RP0/CPU0/0
   ```

3. Assign the primary IPv4 address and subnet mask to the interface.

   Example

   ```
   Router(config-if)# ipv4 address 1.76.18.150/16 (or)
   ipv4 address 1.76.18.150 255.255.0.0
   ```

You can specify the network mask in one of the two ways:

- The network mask is represented in a four-part dotted decimal format. For instance, a mask of 255.255.0.0 signifies that each bit set to 1 indicates the corresponding bit in the address is part of the network address.
- The system indicates the network mask as a slash (/) and number. For example, /16 in the example indicates that the first 16 bits of the mask are ones, and the corresponding bits of the address are the network address.

4. Opyional: Specify the mtu value.

   Example

   ```
   Router(config-if)# mtu 1488
   ```

   Note

   - The maximum transmission unit (MTU) value for the management interface is 9678 bytes.
   - The default is 1514 bytes.
   - The valid range for Management Ethernet interface **mtu** values is from 64 through 9678 bytes.

5. Remove the forced administrative shutdown so the interface can transition to an operational up or down state.

   Example

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

6. End and save your configuration.

   Example

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

   a) When you issue **end**, the system prompts you to commit changes before exiting.

      Example

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

   b) Enter **yes** to save the configuration, exit the configuration session, and return to EXEC mode.
   c) Enter **no** to exit the configuration session without committing the changes.
   d) Enter **cancel** to remain in the current configuration session without exiting or committing the changes.
   e) Use **commit** instead of **end** when you want to save the changes and remain in configuration mode.

7. Verify the operational state of the interface.

   ```
   Router# show running-config interface MgmtEth 0/RP0/CPU0/0
   MgmtEth0/RP0/CPU0/0 is up, line protocol is up
   ```

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

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

```
Router# configure
Router(config)# interface MgmtEth 0/RP0/CPU0/0
Router(config)# ipv4 address 1.76.18.150/16
Router(config-if)# no shutdown
Router(config-if)# commit
Router:Mar 26 01:09:28.685 :ifmgr[190]:%LINK-3-UPDOWN :Interface
MgmtEth0/RP0/CPU0/0, changed state to Up
Router(config-if)# end
```

This example shows VRF configuration and verification of the Management Ethernet interface on the router processor with the source address:

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

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

Router# show run vrf
Wed Jan 30 14:59:00.014 UTC
vrf httpupload
!
```

What to do next

Use these verification results and examples as a baseline for related Management Ethernet interface procedures.

4 Ethernet interfaces

Topics:

This chapter describes the introductory information for configuring Ethernet interfaces on Cisco 8000 Series Routers, including the module scope and the prerequisites that must be met before configuration.

Ethernet interface configuration

Describes the scope of Ethernet interface configuration on Cisco 8000 Series Routers, including the distributed 10-Gigabit, 25-Gigabit, 40-Gigabit, 100-Gigabit, and 400-Gigabit Ethernet architectures and features that interconnect the router with core routers, edge routers, and Layer 2 and Layer 3 switches in POPs.

This module describes the configuration of Ethernet interfaces.

The distributed 10-Gigabit, 25-Gigabit Ethernet, 40-Gigabit, 100-Gigabit Ethernet, 400-Gigabit Ethernet architecture and features deliver network scalability and performance, while enabling service providers to offer high-density, high-bandwidth networking solutions designed to interconnect the router with other systems in POPs, including core and edge routers, Layer 2 switches and Layer 3 switches.

Table 2: Feature history table

Feature Name	Release Information	Feature Description
Introduction of IP MTU	26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Introduction of IP MTU	25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Introduction of IP MTU	25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
Introduction of IP MTU	24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *The IP MTU functionality is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM
Introduction of IP MTU on Q200-based Systems	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 *Programmability Configuration Guide for Cisco 8000 Series Routers*.

Prerequisites for configuring Ethernet interfaces

Lists the prerequisites that must be met before you configure Ethernet interfaces on Cisco 8000 Series Routers, including access to supported hardware, knowledge of the interface IP address, and the generalized interface naming notation.

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.

- Specify the generalized interface name with the standard notation of *rack/slot/module/port*.

An ACL-dependent feature refers to a capability in network systems that relies on Access Control Lists (ACLs) for its operation. These features include:

- global configuration, such as Lawful Intercept (LI) and BGP Flow Specification (BGPFS)
- interface-level configurations, such as Quality of Service with ACL (QoS-ACL), Security ACL, SPAN ACL, Qos Policy Propagation via BGP (QPPB), Policy Based Routing (PBR), Peering QoS, and L2 ACL for packets with L3 payload.

An interface, whether physical or virtual, supports the configuration of up to four ACL dependent features, such as ACLs, QoS with ACL, BGP Flow Specification, SPAN ACL, and Lawful Intercept. To add a new feature, such as Policy-Based Routing, you must first remove one of the existing features and then configure the new feature.

Ethernet interface fundamentals

This topic covers the fundamentals of Ethernet interface, such as supported hardware, ethernet protocol standards, flow control and so on.

Cisco 8000 modular line cards

Describes the modular line cards supported on Cisco 8800 Series Routers, including the 36-port QSFP56-DD 400 GbE line card and the 48-port QSFP28 100 GbE line card, and the forwarding ASICs that these line cards use.

The Cisco 8800 Series Routers support these 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.

Gigabit Ethernet protocol standards

Lists the IEEE Ethernet protocol standards supported by the Gigabit Ethernet interfaces on Cisco 8000 Series Routers, including IEEE 802.3, IEEE 802.3ae, and IEEE 802.3ba.

The Gigabit Ethernet interfaces on Cisco 8000 Series Routers support these IEEE Ethernet protocol standards.

Table 3: Gigabit Ethernet protocol standards

Standard	Description
IEEE 802.3 physical Ethernet infrastructure	Defines 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.

Standard	Description
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.

MAC address

Defines a MAC address as a unique 6-byte address that identifies the interface at Layer 2.

A MAC address is a unique 6-byte address that

- identifies the interface at Layer 2 and
- describes the platform limit on distinct MAC most significant bytes (MSBs) across Layer 3 interfaces on Cisco 8000 Series Routers.

MAC address usage on Layer 3 interfaces

To ensure system stability on Cisco 8000 Series routers running IOS XR, restrict the total number of distinct 4-byte MAC (MSBs) to no more than 15 across all Layer 3 interfaces, including physical interfaces, bundle interfaces, BVI interfaces, and PWHE interfaces.

If you exceed this limit, creation of additional Layer 3 interfaces, such as Bundle-Ether interfaces may fail during hardware programming. The interface may appear to be created and then be immediately removed, and you may see *No space left on device* errors in system logs from the forwarding plane. Once the platform resource limit is reached, you cannot program more interfaces in hardware until resources are freed.

Adhering to this requirement prevents resource exhaustion and ensures reliable operation when configuring multiple Layer 3 interfaces.

Default configuration values for 100-Gigabit Ethernet

Lists 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, including flow control, MTU, and MAC address defaults.

This table describes the default interface configuration parameters that are present when an interface is enabled on a 36-port Line Card or a 48-port Line Card.

Note

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

Table 4: 100-Gigabit Ethernet line card default configuration values

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

Flow control on Ethernet interfaces

Describes flow control on 10-Gigabit Ethernet interfaces, which periodically sends pause frames, differs from the full and half-duplex flow control used on standard management interfaces, and can be activated or deactivated for ingress traffic only.

The 10-Gigabit Ethernet flow control is a mechanism that:

- sends flow control pause frames periodically
- differs from standard full and half-duplex flow control, and
- manages traffic flow on 10-Gigabit Ethernet interfaces.

The router manages traffic flow through these configurations:

- You can activate or deactivate flow control for only ingress traffic.
- The router automatically implements flow control for egress traffic.

Configure physical Ethernet interfaces

Describes how to create a basic Ethernet interface configuration on a Cisco 8000 Series Router, including how to assign an IPv4 address, enable flow control, set the MTU, remove the shutdown state, and verify the configured interface.

Before you begin

Check the current software version and the configured interface, and check the status of each interface port by using these commands:

- **show version** - displays the current software version, and can also be used to confirm that the router recognizes the line card.
- **show interfaces** [**TenGigE FortyGigE HundredGigE FourHundredGigE**] *interface-path-id* - displays the configured interface and checks the status of each interface port.

Use this procedure to create a basic Ethernet interface configuration.

Procedure

1. Enter the global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter the interface configuration mode and specify the Ethernet interface name and notation *rack/slot/module/port***show interfaces** [**TenGigE FortyGigE HundredGigE FourHundredGigE**] *interface-path-id*

   Example

   ```
   Router(config)# interface HundredGigE 0/1/0/1
   ```

   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

3. Assign an IP address and subnet mask to the interface by using **ipv4 address** *ip-address mask* command.

   Example

   ```
   Router(config-if)# ipv4 address 172.18.189.38 255.255.255.224
   ```

- 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.

4. Optional: Enable the sending and processing of flow control pause frames by using the**flow-control {bidirectional| egress | ingress}** command.

   Example

   ```
   Router(config-if)# flow control ingress
   ```

   - **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.

5. Use **mtu** *bytes* to set the MTU value for the interface.

   Example

   ```
   Router(config-if)# mtu 1448
   ```

   - 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.

6. Use **no shutdown** to remove the shutdown configuration, which forces an interface administratively down.

   Example

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

7. Save configuration changes. **end** or **commit**

   Example

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

   or

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

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

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

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

Network interface speed and autonegotiation

Describes autonegotiation as an IEEE 802.3 Ethernet standard that enables connected routers to determine the highest common transmission parameters, such as mode, speed, and flow control, to optimize link bandwidth utilization.

Autonegotiation is an IEEE 802.3 Ethernet standard that

- enables a router to connect with its peer by automatically determining and negotiating to the highest common supported transmission parameters between them,
- exchanges parameters, such as mode, speed, and flow control, between the two connected routers, and
- supports autonegotiation on 1 G copper ports and 1 G SFP copper transceivers that allow an SFP port to connect to a copper Ethernet network at 1 Gbps using a standard RJ45 connector.

Table 5: Feature history table

Feature Name	Release Information	Feature Description
Network interface speed support with autonegotiation	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) You can ensure compatibility between new and legacy routers by enabling autonegotiation on the Gigabit interfaces of the connected routers. This process allows the routers to exchange key parameters such as mode, speed, and flow control, enabling them to agree on common transmission settings. As a result, the link achieves optimal bandwidth utilization. *This feature is supported on: • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D

Network interface speed configuration methods

Lists the three methods available to configure network interface speed on Cisco 8000 Series Routers, and notes the Cisco recommendation to configure network interface speed in autonegotiation mode.

You can configure the network interface speed using one of these methods:

- Configure with **speed** option
- Configure with **negotiation auto** option
- Configure with **speed** and **negotiation auto** options

> **Note**
>
> Cisco recommends configuring network interface speed in autonegotiation mode.

Configure with speed option

Describes how to force a 1 Gbps network interface to operate at a specified value by using the **speed** command, including how to check the existing speed, apply the configuration, and verify that the speed is set and autonegotiation is disabled.

When you configure a 1 Gbps network interface using the **speed** command, the interface is forced to operate at the specified value.

Procedure

1. Run the **show interfaces GigabitEthernet** *interface-name* command to check the existing speed.

   Example

```
#show interfaces GigabitEthernet 0/0/0/31
GigabitEthernet0/0/0/31 is up, line protocol is up
  Interface state transitions: 7
  Hardware is GigabitEthernet, address is 0035.1a00.e62c (bia 0035.1a00.e62c)

  Internet address is Unknown
  MTU 1514 bytes, BW 100000 Kbit (Max: 100000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 1000 Mb/s, TFD, 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 00:00:30
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters never
  30 second input rate 1000 bits/sec, 1 packets/sec
  30 second output rate 0 bits/sec, 1 packets/sec
     90943 packets input, 11680016 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 90943 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     61279 packets output, 4347618 bytes, 0 total output drops
     Output 0 broadcast packets, 8656 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
8 carrier transitions
```

   Checks the existing speed. Observe that the speed is 1000 Mbps.

2. Use **interface GigabitEthernet** *interface-path-id*, **speed** *speed-value* command to set the configuration that forces 1 Gbps interface speed to 100 Mbps.

   Example

```
 (config)#interface GigabitEthernet 0/0/0/31
 (config-if)#speed 100
 (config-if)#commit
 (config-if)#end
```

   Sets the configuration that forces the 1 Gbps interface speed to 100 Mbps. If the remote end is not also configured for 100 Mbps, the link goes down.

3. Run the **show controllers GigabitEthernet** *interface-path-id* and **show interfaces GigabitEthernet** *interface-path-id* commands to verify if the speed is configured to 100 Mbps and autonegotiation is disabled.

   Example

```
#show controllers GigabitEthernet 0/0/0/31
Operational data for interface GigabitEthernet0/0/0/31:
State:
    Administrative state: enabled
```

```
    Operational state: Up
    LED state: Green On
Phy:
    Media type: Four-pair Category 5 UTP PHY, full duplex
    Optics:
        Vendor: CISCO
        Part number: SBCU-5740ARZ-CS1
        Serial number: AVC194525HW
        Wavelength: 0 nm
MAC address information:
    Operational address: 0035.1a00.e67c
    Burnt-in address: 0035.1a00.e62c
Autonegotiation disabled.

Operational values:
    Speed: 100 Mbps       /*Gig interface speed is set to 100 Mbps */
    Duplex: Full Duplex
    Flowcontrol: None
    Loopback: None (or external)
    MTU: 1514
    MRU: 1514
    Forward error correction: Disabled
```

```
#show interfaces GigabitEthernet 0/0/0/31
GigabitEthernet0/0/0/31 is up, line protocol is up
  Interface state transitions: 7
  Hardware is GigabitEthernet, address is 0035.1a00.e62c (bia 0035.1a00.e62c)

  Internet address is Unknown
  MTU 1514 bytes, BW 100000 Kbit (Max: 100000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 100 Mb/s, TFD, 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 00:00:30
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
      carrier transitions
```

The show output confirms that GigabitEthernet0/0/0/31 is up and its line protocol is up. This occurs when both ends of the link are operating at 100 Mbps. If the remote end is not also configured for 100 Mbps, the link goes down.

Verifies that the speed is configured to 100 Mbps and autonegotiation is disabled.

Configure with negotiation auto option

Describes how to autonegotiate the network interface speed with the remote end by using the **negotiation auto** command, which enables the speed capability of 10 Mbps, 100 Mbps, or 1 Gbps to be negotiated with the peer, and how to verify that the speed is autonegotiated.

Procedure

---

1. Set the configuration for `interface` speed to auto-negotiate. The router sets its interface speed to match the highest common speed supported by the peer router.

Example

```
Router#configuration
 Router(config)#interface GigabitEthernet 0/0/0/31
 Router(config-if)#negotiation auto
 Router(config-if)#commit
 Router(config-if)#end
```

Sets the configuration for the interface speed to autonegotiate. The router sets its interface speed to match the highest common speed supported by the peer router. Autonegotiation is not enabled by default. Use the **negotiation auto** command to enable autonegotiation.

2. Run the **show controllers GigabitEthernet** *interface-path-id* and **show interfaces GigabitEthernet** *interface-path-id* commands to verify if the speed is autonegotiated.

Example

```
Router#show interfaces GigabitEthernet 0/0/0/31
GigabitEthernet0/0/0/31 is up, line protocol is up
  Interface state transitions: 10
  Hardware is GigabitEthernet, address is 0035.1a00.e62c (bia 0035.1a00.e62c)

  Internet address is Unknown
  MTU 1514 bytes, BW 100000 Kbit (Max: 100000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 100Mb/s, TFD, link type is autonegotiation
  output flow control is off, input flow control is off
  Carrier delay (up) is 10 msec
  loopback not set,
  Last link flapped 00:00:01
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters never
  30 second input rate 1000 bits/sec, 1 packets/sec
  30 second output rate 0 bits/sec, 0 packets/sec
     91005 packets input, 11687850 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 91005 multicast packets
              0 runs, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     61307 packets output, 4350024 bytes, 0 total output drops
     Output 0 broadcast packets, 8668 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     15 carrier transitions
```

```
 #show controllers GigabitEthernet 0/0/0/31
Operational data for interface GigabitEthernet0/0/0/31:

State:
    Administrative state: enabled
    Operational state: Up
    LED state: Green On

Phy:
    Media type: Four-pair Category 5 UTP PHY, full duplex
    Optics:
        Vendor: CISCO
        Part number: SBCU-5740ARZ-CS1
        Serial number: AVC194525HW
        Wavelength: 0 nm
```

```
MAC address information:
    Operational address: 0035.1a00.e67c
    Burnt-in address: 0035.1a00.e62c

Autonegotiation enabled:
    No restricted parameters

Operational values:
    Speed: 100 Mbps
    Duplex: Full Duplex
    Flowcontrol: None
    Loopback: None (or external)
    MTU: 1514
    MRU: 1514
    Forward error correction: Disabled
```

Verifies that the speed is autonegotiated.

Configure with speed and negotiation auto options

Describes how to force a 1 Gbps network interface to operate at a specified speed while autonegotiating other parameters, such as full duplex and pause, by using both the **speed** and **negotiation auto** commands, and how to verify the resulting configuration.

Procedure

---

1. Set the configuration for Gigabit interface speed to 100 Mbps and autonegotiate other parameters. The interface speed at remote end is set to 100 Mbps.

   Example

   ```
   Router(config)#interface GigabitEthernet 0/0/0/31
   Router(config-if)#negotiation auto
   Router(config-if)#speed 100
   Router(config-if)#end
   ```

   Sets the configuration for the Gigabit interface speed to 100 Mbps and autonegotiates the other parameters. The interface speed at the remote end is set to 100 Mbps. The interface does not attempt to negotiate the speed with its peer; it simply tries to establish a link at 100 Mbps. The autonegotiation process might still occur for other parameters like duplex and flow control, but the speed itself is fixed by the **speed** command.

2. Use the **show controllers GigabitEthernet** *interface-path-id* and **show interfaces GigabitEthernet** *interface-path-id* command to verify if the link is up, speed is forced to 100 Mbps, and auto-negotiation is enabled.

   Example

   ```
   Router#show interfaces GigabitEthernet 0/0/0/31
   GigabitEthernet0/0/0/31 is up, line protocol is up
     Interface state transitions: 9
     Hardware is GigabitEthernet, address is 0035.1a00.e62c (bia 0035.1a00.e62c)

     Internet address is Unknown
     MTU 1514 bytes, BW 100000 Kbit (Max: 100000 Kbit)
        reliability 255/255, txload 0/255, rxload 0/255
   ```

```
  Encapsulation ARPA,
  Full-duplex, 100 Mb/s, TFD, link type is autonegotiation
  output flow control is off, input flow control is off
  Carrier delay (up) is 10 msec
  loopback not set,
  Last link flapped 00:00:03
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters never
  30 second input rate 0 bits/sec, 1 packets/sec
  30 second output rate 0 bits/sec, 0 packets/sec
     90968 packets input, 11683189 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 90968 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     61287 packets output, 4348541 bytes, 0 total output drops
     Output 0 broadcast packets, 8664 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     12 carrier transitions
```

The show output confirms that GigabitEthernet0/0/0/31 is up and line protocol is up. This occurs because the speed at both ends is 100 Mbps.

```
#show controllers GigabitEthernet 0/0/0/31
Operational data for interface GigabitEthernet0/0/0/31:

State:
    Administrative state: enabled
    Operational state: Up
    LED state: Green On

Phy:
    Media type: Four-pair Category 5 UTP PHY, full duplex
    Optics:
        Vendor: CISCO
        Part number: SBCU-5740ARZ-CS1
        Serial number: AVC194525HW
        Wavelength: 0 nm

MAC address information:
    Operational address: 0035.1a00.e67c
    Burnt-in address: 0035.1a00.e62c

Autonegotiation enabled:
    Speed restricted to: 100 Mbps  /* autonegotiation is enabled and speed
is forced to 100 Mbps*/

Operational values:
    Speed: 100 Mbps
    Duplex: Full Duplex
    Flowcontrol: None
    Loopback: None (or external)
    MTU: 1514
    MRU: 1514
    Forward error correction: Disabled
```

Verifies that the link is up, the speed is forced to 100 Mbps, and autonegotiation is enabled.

QoS with autonegotiated speeds

Describes QoS with autonegotiated speeds as a congestion management feature that manages congestion on interfaces which automatically adjust their speed to connect with legacy devices, and that prioritizes critical applications, controls bandwidth allocation, and minimizes packet loss on lower-speed links.

QoS with autonegotiated speeds is a congestion management feature that

- manages congestion on interfaces that automatically adjust their speed—such as 10 Mbps or 100 Mbps—to connect with legacy devices,
- addresses reduced bandwidth and increased congestion potential when operating at lower speeds, and
- prioritizes critical applications, controls bandwidth allocation, and minimizes packet loss or delay, even when connecting to legacy devices or working in mixed-speed environments.

Table 6: Feature history table

Feature Name	Release Information	Feature Description
QoS with auto-negotiated speeds	Release 25.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) You can now maintain optimal network efficiency by applying QoS on Gigabit Ethernet interfaces that operate at lower speeds through autonegotiation. Autonegotiation detects and matches the highest common speed between connected routers, allowing seamless interoperability. With this enhancement, you can use QoS on interfaces connecting to legacy devices that support only 10 Mbps or 100 Mbps speeds. This capability improves traffic management, leading to enhanced network performance and reliability across diverse deployment scenarios. *This feature is supported on 8011-4G24Y4H-I.

Benefits of QoS with autonegotiated speeds

Lists the benefits of applying QoS on interfaces that operate at autonegotiated speeds, including maximized bandwidth utilization when the interface autonegotiates to a higher speed and optimized resource allocation when the interface autonegotiates to a lower speed.

Applying QoS policies on interfaces that operate at autonegotiated speeds provides the following benefits.

- Maximized bandwidth utilization: When an interface autonegotiates to a higher speed, the QoS policy dynamically scales up its allocations and limits. This ensures that the newly available bandwidth is fully leveraged by all traffic classes according to their defined priorities, preventing under-utilization of network resources.
- Optimized resource allocation at lower speeds: If an interface autonegotiates down to a lower speed, QoS mechanisms, such as shapers and bandwidth allocations, automatically adjust downward, preventing enforcement of rates that exceed physical link capacity. This minimizes packet drops, control plane overhead, and inconsistent state conditions.

Supported interfaces for QoS with autonegotiation

Lists the interface types on which QoS with autonegotiation is supported, including Gigabit Ethernet interfaces that connect to legacy devices and bundle interfaces that include Gigabit Ethernet member links operating at autonegotiated speeds.

QoS with autonegotiation is supported on the following interface types.

- Gigabit Ethernet interfaces that need to connect to legacy devices operating at 10 Mbps or 100 Mbps speeds.
- Bundle interfaces that include Gigabit Ethernet interfaces operating at autonegotiated speeds of 10 Mbps, 100 Mbps, or 1 Gbps.

Restrictions for QoS with autonegotiated speeds

Lists the restrictions that apply when you configure QoS on interfaces that operate at autonegotiated speeds, including restrictions on absolute rate values on bundle interfaces and Hierarchical QoS (HQoS) support on 10 or 100 M ports of specific systems.

- Absolute rate values, such as fixed, specific numerical bandwidth or rate limits, cannot be configured on bundle interfaces.
- Hierarchical QoS (HQoS) policies are not supported on 10 or 100 M ports on main and sub-interfaces of 8011-4G24Y4H-I, 8011-32Y8L2H2FH, and 8011-32Y8L2H2FH (ASIC: A100) systems.

Enable QoS with autonegotiated speeds

Describes how to enable QoS with autonegotiated speeds on Cisco IOS XR routers by using relative values, including how to configure the interface to autonegotiate, define class maps and a policy map that uses bandwidth percentages, apply the policy to the interface, and verify the configuration.

The purpose of this task is to enable QoS with autonegotiated speeds using relative values on Cisco IOS XR routers.

Before you begin

Confirm that the interface you intend to configure supports autonegotiation and QoS.

Procedure

1. Configure the interface to allow auto-negotiation.

   Example

   ```
   Router(config)#interface GigabitEthernet0/0/0/28
   Router(config)#negotiation auto
   ```

   Configures the interface to allow autonegotiation.
2. Define the class-maps for traffic classification.

   Example

   ```
   Router(config)#class-map match-any voice_traffic
   Router(config-cmap)#match dscp ef
   Router(config-cmap)#end-class-map
   Router(config)#class-map match-any data_traffic
   Router(config-cmap)#match dscp af21
   Router(config-cmap)#end-class-map
   ```

3. Define the policy map using bandwidth percentages.

   Example

```
Router(config)#policy-map adaptive_qos_policy
Router(config-pmap)#class voice_traffic
Router(config-pmap)#priority level 1
Router(config-pmap)#bandwidth percent 20  ! Allocates 20% of the *negotiated*
 link speed
Router(config-pmap)#class data_traffic
Router(config-pmap)#bandwidth percent 50  ! Allocates 50% of the *negotiated*
 link speed
Router(config-pmap)#class class-default
Router(config-pmap)#end-policy-map
```

   We recommend using the percentile of port speed for interfaces with auto-negotiated speeds, as this allows the policy map to automatically adjust to any changes in speed.

4. Apply the policy to the auto-negotiating interface.

   Example

```
Router(config)#interface GigabitEthernet0/0/0/28
Router(config)#service-policy output adaptive_qos_policy
```

   Applies the policy to the autonegotiating interface.

5. Execute the **show qos inconsistency detail location** *location* command to identify any conflict between a configured QoS policy and the interface's current operational state.

   Example

```
Router# show qos inconsistency detail location 0/rp0/cpu0
Fri Mar 28 14:53:36.542 UTC
Inconsistency Type: Policer conform rate greater than parent reference rate
=============================================================================
Interface                                  Direction      Policy Name
-----------------------------------------------------------------------------
GigabitEthernet0/0/0/28                     input          p1-ing-negative

Inconsistency Type: Shape bandwidth rate greater than parent reference rate
=============================================================================
Interface                                 Direction      Policy Name
-----------------------------------------------------------------------------
GigabitEthernet0/0/0/28                     output         p1-negative
```

Autonegotiation parameters

Lists the Ethernet transmission parameters, such as speed, duplex, and pause that connected routers exchange during autonegotiation, and provides description for each parameter.

These parameters are crucial for Ethernet transmission because they directly define how data is sent and received over the link, impacting performance, efficiency, and reliability.

- Speed – 1G copper ports and 1G SFP copper transceivers support autonegotiation, allowing them to establish the highest common speed and duplex mode with connected routers.
- Duplex – Full.
- Pause – Receive part (RX) and Transmit part (TX).

Configuration guidelines and restrictions for autonegotiation and speed

Summarizes how speed is configured or autonegotiated on the interface and lists the guidelines and restrictions to follow when you enable or disable autonegotiation on Cisco 8000 Series Routers.

Speed and autonegotiation guidelines

- Speed can be manually set or configured to autonegotiate with the remote interface.
- In autonegotiation mode, the interface automatically determines whether to operate at 10 Mbps, 100 Mbps, or 1 Gbps based on the capabilities of the connected device.
- Other settings such as full duplex and pause are also negotiated during this process.
- Autonegotiation is an optional function of the Fast Ethernet standard that enables devices to automatically exchange information about speed and duplex abilities.
- It is especially useful for ports where devices with different capabilities are connected and disconnected regularly.
- Autonegotiation is enabled, by default, on integrated copper ports and copper transceivers on most devices.
- Do not enable autonegotiation when you configure a link on K100-based systems. Ensure autonegotiation is disabled on both ends of the link.
- For Cisco 8711-32FH-M routers, an autonegotiation mismatch between source and destination interfaces does not cause the link to go down; the link remains active. To avoid potential loss of traffic due to an autonegotiation mismatch, enable autonegotiation on both source and destination interfaces.

Interfaces and subinterfaces on the router

In Cisco IOS XR, interfaces are, by default, main interfaces (also known as trunk interfaces) on Cisco 8000 Series Routers, under which the system creates logical subinterfaces for efficient network segmentation and management.

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

Feature history table
Table 7: Feature history table

Feature Name	Release Information	Feature Description
Interfaces and subinterfaces	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100]) (select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.

Feature Name	Release Information	Feature Description
Interfaces and subinterfaces	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Interfaces and subinterfaces	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Interfaces and subinterfaces	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) This feature that allows for the configuration of main trunk interfaces that are either physical or bundle interfaces, automatic creation of physical interfaces, and creation of logical subinterfaces with unique IDs for efficient network segmentation and management is now supported on the following hardware. *This feature is now supported on the Cisco 8712-MOD-M routers.
Interfaces and subinterfaces	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *The support for Interfaces and subinterfaces is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-36EH • 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 the Cisco 88-LC1-36EH line cards.

Trunk interfaces

Trunk interfaces on Cisco 8000 Series Routers are the main interfaces under which you create subinterfaces, and are either physical interfaces that the system creates automatically when the router recognizes a card or bundle interfaces that you create at configuration time.

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; you must create them at the time of configuration.

Trunk interface configuration samples

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

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

Create a subinterface

Identify a trunk interface, assign a unique subinterface number, and configure the subinterface under the trunk interface, so you can create logical interfaces with unique identifiers under a physical or bundle trunk interface for efficient network segmentation.

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

Procedure

1. 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.

2. 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.

Example

In this 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#
```

Example

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
 : Configuration committed by user 'root'. Use 'show configuration commit
changes 1000000149' 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
!
```

Display subinterfaces

Use the **show run** and **show interface** commands to display a trunk interface followed by its subinterfaces in ascending numerical order and to view the unique counters for a configured subinterface.

Procedure

1. Use the **show run** command to display the trunk interface first, then the subinterfaces in ascending numerical order.

   Example

```
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
!
```

2. 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:

   This 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.

   Example

```
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


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
```

Delete subinterfaces

Delete a subinterface from a trunk interface by using the **no interface** command, so you can remove logical interfaces that you no longer need under a physical or bundle trunk interface.

Procedure

Use the **no interface** command to delete a subinterface.

Example

```
Router#
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
Router(config)# no interface HundredGigE 0/5/0/0.0
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.
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
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

Describes how a trunk interface on Cisco 8000 Series Routers can be either a Layer 2 or Layer 3 interface based on whether the **l2transport** keyword is used, the differences between routed Layer 3 interfaces that accept IP addresses and switched Layer 2 interfaces that must connect to an L2VPN component as access connections, the rule that subinterfaces can only be created under a Layer 3 trunk interface, and how a Layer 2 subinterface implements an Ethernet Flow Point (EFP).

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 subinterfaces in the same way.

Layer 3 trunk interfaces and subinterfaces

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.

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 subinterfaces.

Layer 2 trunk interfaces and subinterfaces

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.

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 does not send any traffic to it.

Ethernet Flow Points (EFPs)

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 subinterfaces. Refer to the appropriate Cisco IOS XR configuration guide for other restrictions.

Example: Layer 2 and Layer 3 interface configuration

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

```
Router# config
Mon Sep 21 12:05:33.142 EDT
Router(config)# interface HundredGigE0/5/0/0
Router(config-if)# ipv4 address 10.0.0.1/24
Router(config-if)# commit
Router: config[65794]: %MGBL-CONFIG-6-DB_COMMIT : Configuration committed by
user 'root'. Use 'show configuration commit changes 1000000160' to view the
changes.
Router(config-if)# end
Router : config[65794]: %MGBL-SYS-5-CONFIG_I : Configured from console by root
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
!
Router#
Router# conf
Mon Sep 21 12:08:07.426 EDT
Router(config)# interface HundredGigE0/5/0/0 l2transport
```

```
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
Router(config-if-l2)# no ipv4 address
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.
Router(config-if)# end
Router# show run interface HundredGigE0/5/0/0
Mon Sep 21 12:09:02.471 EDT
interface HundredGigE0/5/0/0
 negotiation auto
 l2transport
 !
!
Router# conf
Router(config)# interface HundredGigE0/5/0/0.0
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
Router(config-subif)# interface HundredGigE0/5/0/0
Router(config-if)# no l2transport
Router(config-if)# interface HundredGigE0/5/0/0.0
Router(config-subif)# encapsulation dot1q 99
Router(config-subif)# ipv4 address 11.0.0.1/24
Router(config-subif)# interface HundredGigE0/5/0/0.1 l2transport
Router(config-subif)# encapsulation dot1q 700
Router(config-subif)# commit
Router(config-subif)# end
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
!
```

Untagged L2 subinterface

An untagged L2 subinterface is a subinterface on a network device that is not linked to a VLAN tag, and it takes a higher priority than the main interface on service mapping so that untagged traffic from customer edge devices can be processed as l2transport traffic on Cisco 8000 Series Routers.

The untagged L2 subinterface is a subinterface on a network device that

- operates without a VLAN tag, which divides network traffic into smaller, distinct networks

- improves security and manageability in network setups, and
- handles untagged traffic.

Feature history table
Table 8: Feature history table

Feature Name	Release Information	Feature Description
Untagged L2 subinterface	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Untagged L2 subinterface	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Untagged L2 subinterface	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature support is now extended to the 8011-4G24Y4H-I routers.
Untagged L2 subinterface	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) This feature support is now extended to the Cisco 8712-MOD-M routers.
Untagged L2 subinterface	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700)(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature support is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Feature Name	Release Information	Feature Description
Untagged L2 subinterface	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100])(select variants only*) 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 is now supported on: • 88-LC1-36EH This feature introduces the **encapsulation untagged** command.

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 that uses an unsupported VLAN tag format is treated as untagged traffic and mapped to an untagged subinterface; the router recognizes single-tag dot1q, double-tag, and three-or-more-tag VLAN formats as supported.

Traffic using an unsupported VLAN tag format is considered untagged traffic and mapped to an untagged subinterface.

These 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

Untagged L2 subinterfaces let you manage untagged and tagged traffic on the same port to different services and rewrite different VLANs on untagged subinterfaces for both tagged and untagged traffic.

Untagged L2 subinterfaces provide the following benefits.

- 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 the untagged L2 subinterface works

In a point-to-point L2VPN topology between two CE devices, the PE nodes map untagged traffic to L2 subinterfaces and translate VLAN tags across the pseudowire so that traffic reaches each CE with the correct encapsulation.

Consider a sample topology in which the P2P L2VPN service is deployed between two CE devices.

Summary

In this topology:

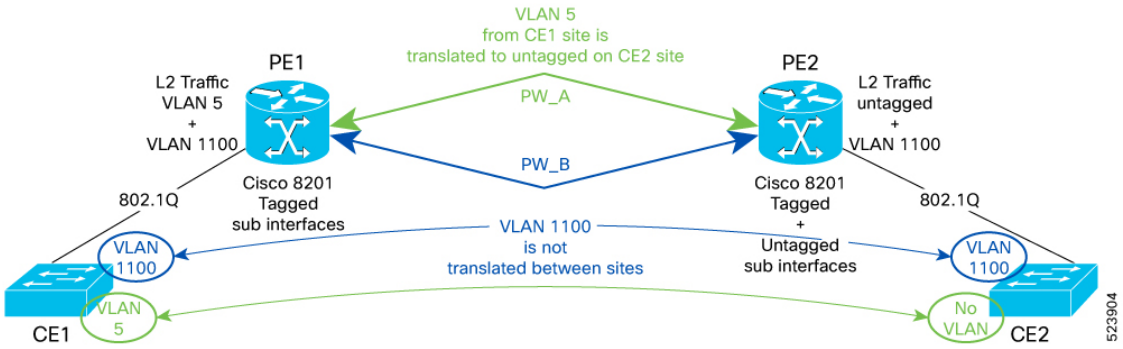
- VLAN 1100 is a common VLAN that is shared between CE1 and CE2.

  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.

Workflow

Figure 1: Untagged L2 subinterface in a P2P L2VPN topology



1. 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.
2. When 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 an untagged L2 subinterface

Create an untagged L2 subinterface under an L2 transport main interface by using the **encapsulation untagged** keyword, then verify that untagged encapsulation is configured, so the router can efficiently map untagged frames from customer edge devices to a subinterface and apply Layer 2 services to that traffic.

To manage traffic efficiently, configure an untagged L2 subinterface.

Procedure

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

   This feature 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
```

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)

Lists the **performance-mgmt** and **show performance-mgmt** commands that support the **interface basic-counters** keyword for performance statistics collection and display on Layer 2 subinterfaces (EFPs), and describes the Layer 2 basic-counter attributes reported by these commands.

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

Commands that support the interface basic-counters keyword

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**

Layer 2 basic-counter attributes

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 output.

Table 9: Layer 2 basic-counter attributes

Attribute	Description
InOctets	Bytes received (64-bit)
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)

Layer 2 VPN on Ethernet interfaces

Describes Layer 2 Virtual Private Network (L2VPN) connections that emulate LAN behavior across a Layer 2 switched, IP, or MPLS-enabled IP network.

Layer 2 Virtual Private Network (L2VPN) is a network connection that

- emulates LAN behavior across L2 switched, IP, or MPLS-enabled IP networks
- enables service providers to provide Layer 2 services to geographically disparate customer sites, and
- connects Ethernet devices as if they exist on a common LAN segment.

L2VPN in service provider networks

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.

Configuration guidelines for L2VPN on Ethernet interfaces

This topic cover the guidelines when you configure Layer 2 VPN on an Ethernet interface.

These guidelines apply when you configure L2VPN on Ethernet interfaces:

- 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.

Layer 2 VPN AC configuration example

Provides a configuration example that shows how to configure a Layer 2 VPN attachment circuit (AC).

This 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
```

Carrier delay on physical interfaces

Describes how the carrier delay introduces a delay in processing interface link-state notifications on physical Ethernet interfaces so that the interface link has time to stabilize.

The carrier delay is a ethernet interfaces configuration that

- processes interface link-state notifications
- provides time for the interface link to stabilize, and
- avoids unnecessary interface state changes.

Hardware links experience link flaps, which are conditions where a physical interface fluctuates between up and down states. These flaps cause the network to reestablish routing paths and lead to resource exhaustion on routers. The system manages link stability through these timer processes:

- The `carrier-delay up` timer starts when the link state transitions to an up state.
- The `carrier-delay down` timer starts when the link state transitions to a down state.
- The Ethernet interface maintains its current state during the delay period until the timer expires.

Feature history

Feature name	Release information	Feature description
Default carrier delay value on physical interfaces		Introduced in this release on: Modular Systems (8800 [LC ASIC: K100])(select variants only*) *This feature is supported on Cisco 88-LC1-48Y8F-EM line cards.
Default carrier delay value on physical interfaces	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Default carrier delay value on physical interfaces	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100])(select variants only*) *This feature is now supported on: • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Default carrier delay value on physical interfaces	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Default carrier delay value on physical interfaces	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) You can now set the carrier-delay up default value on the listed hardware that provides sufficient time to establish a stable hardware link state. If you have not set the default timer, the default carrier delay automatically delays the hardware link-up notifications by 200 ms. *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-36EH+A8:B12 • 88-LC1-52Y8H-EM

Feature name	Release information	Feature description
Default carrier delay value on physical interfaces	Release 24.3.1	Introduced in this release on: Fixed Systems(8200, 8700)(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*). 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. *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Default carrier delay value on physical interfaces	Release 24.2.11	Introduced in this release on: ; Modular Systems (8800 [LC ASIC: P100])(select variants only*). 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. *This feature is now supported on 88-LC1-36EH.

Feature name	Release information	Feature description
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**

Guidelines for carrier delay

Lists the usage guidelines that apply when you set the carrier delay on physical interfaces.

These guidelines apply when you configure carrier delay.

- You can configure carrier-delay for only link-up, only link-down, or both link-up and link-down notifications.
- From Release 24.2.11, the default value of the carrier-delay up parameter is changed to 200 ms. To restore your original configuration, configure the parameter explicitly by using the **carrier-delay** command.
- If not configured, the carrier-delay up parameter defaults to 200 ms and the carrier-delay down parameter defaults 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 by using the **carrier-delay up** (milliseconds) **down** (milliseconds) command.

Restrictions of carrier delay

Lists the restrictions that apply when you set the carrier delay on physical interfaces.

These restrictions apply when you configure carrier delay:

- 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.

Configure the carrier-delay timer

Configure the carrier-delay up and carrier-delay down timers on a physical Ethernet interface.

This example shows that when you bring up an interface without configuring the **carrier-delay** command, the carrier-delay configuration for the interface defaults to 200 ms:

```
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 the default carrier-delay value:

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

To change the carrier-delay up and carrier-delay down values on an interface, perform the following steps. In this example, link-up notifications are delayed by 1000 ms and link-down notifications are delayed by 150 ms.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router#configure
   ```

2. Enter interface configuration mode for the target physical interface.

   Example

   ```
   Router(config)#interface HundredGigE 0/0/0/0
   ```

3. Configure the carrier-delay up and carrier-delay down values on the interface.

   Example

   ```
   Router(config-if)#carrier-delay up 1000 down 150
   ```

4. Commit the configuration, then verify the running configuration and the carrier-delay values on the interface.

   Example

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

   The following running configuration shows the carrier-delay up and carrier-delay down values on interface HundredGigE0/0/0/0:

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

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

Example

```
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
```

Interface statistics

This topic covers interface statistics, traffic rate displays, and real-time monitoring commands.

Interface counters report

Describes the interface counters report that summarizes the statistics for all interfaces configured on the router, including 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 Interface Counters report is a functionality you can use on interfaces that

- provides statistics for all configured interfaces
- displays interface status and packet counts, and
- calculates traffic rates over a configurable load interval.

The report displays these statistics:

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

The **show interfaces** command provides detailed statistics based on the following configurations:

- The command displays the input and output rate for each interface.
- 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.

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

> **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.

Instant display of traffic rates for physical interfaces

Describes how the **show interfaces counters rates physical** command displays a snapshot of the traffic throughput and traffic rate on all physical interfaces on the router.

The **show interfaces counters rates physical** command is a diagnostic tool that

- provides a snapshot of physical interface statistics
- presents data in a tabular format for analysis, and
- displays real-time link utilization.

Feature history

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 router provides real-time statistics through the following process:

- The command displays input and output rates in Mbps and Kpps.
- The router calculates statistics over a 5-second interval.
- The router adjusts the time interval during high processing loads, such as routing updates.

View the statistics

Use the **show interfaces counters rates physical** command to view statistics of all physical interfaces.

```
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
```

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.

Display of traffic rates for bundle interfaces

Describes how the **show interfaces counters rates bundle** command displays a snapshot of the traffic throughput and traffic rate on all bundle interfaces on the router at a given instant in a tabular format for easy analysis.

The **show interfaces counters rates bundle** command displays a snapshot of statistics for all the bundle interfaces at a given instant for your quick reference. The statistics display is in a tabular format facilitating easy analysis.

Feature history

Feature name	Release information	Description
Display of traffic rates for bundle interfaces	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Display of traffic rates for bundle interfaces	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Display of traffic rates for bundle interfaces	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature name	Release information	Description
Display of traffic rates for bundle interfaces	Release 24.4.1	Routers can now display a snapshot of the traffic throughput and traffic rate on all bundle interfaces over the last few seconds, facilitating easy analysis. These statistics are presented in a tabular format for quick reference. The feature introduces these changes: **CLI:** • **show interfaces counters rates bundle** **YANG Data Models:** • The existing Xpath `interface-rate` in `Cisco-IOS-XR-infra-statsd-oper.yang` is extended to retrieve bundle interface statistics. (see GitHub, YANG Data Models Navigator)

Guidelines and restrictions for the display of traffic rates for bundle interfaces

Lists the guidelines and restrictions that apply when you display traffic rates on bundle interfaces.

These configuration guidelines and restrictions apply for the display of traffic rates for bundle interfaces:

- The displayed traffic rate represents the real-time link utilization for a given time interval. This interval is system-determined and may vary depending on the system processing load. During events such as routing updates, the time interval may increase.
- The bundle main interface displays Ethernet Layer 2 counters that are derived from bundle MIB statistics. Bundle subinterfaces display Layer 3 protocol statistics. On bundle subinterfaces, byte counts exclude Ethernet overhead, CRC, and VLAN encapsulation bytes. As a result, the input and output bits-per-second values on the bundle main interface can differ from the values on its subinterfaces.

View bundle interface statistics

Use the **show interfaces counters rates bundle** command to view a snapshot of statistics for all bundle interfaces on the router at a given instant.

Procedure

Use the **show interfaces counters rates bundle** command to view the statistics of all bundle interfaces.

Example

```
Router#show interfaces counters rates bundle
Wed Aug 14 19:54:04.842 EDT
InterfaceName                Intval     InMbps    InBW%    InKpps     OutMbps
  OutBW%    OutKpps
Bundle-Ether111                0:03    38454.9    40.0%     8064.8         0.0
     0.0%         0.0
```

```
Bundle-Ether112                0:03         0.0      0.0%         0.0          0.0
      0.0%          0.0
Bundle-Ether1201               0:03         0.0      0.0%         0.0       9741.6
      3.4%       2002.2
Bundle-Ether1202               0:03         0.0      0.0%         0.0       9703.0
      5.0%       1995.4
Bundle-Ether1203               0:03         0.0      0.0%         0.0       9841.9
      5.1%       2023.0
Bundle-Ether1204               0:03         0.0      0.0%         0.0      10011.3
     10.4%       2058.5
Bundle-Ether1205               0:03         0.0      0.0%         0.0          0.0
      0.0%          0.0
Bundle-Ether1206               0:03         0.0      0.0%         0.0          0.0
      0.0%          0.0
Bundle-Ether1207               0:03         0.0      0.0%         0.0          0.0
      0.0%          0.0
Bundle-Ether1208               0:03         0.0      0.0%         0.0          0.0
      0.0%          0.0
Bundle-Ether1209               0:03         0.0      0.0%         0.0          0.0
      0.0%          0.0
Bundle-Ether1210               0:03         0.0      0.0%         0.0          0.0
      0.0%          0.0
Bundle-Ether1301               0:03     38698.3     13.4%      8062.7          0.0
      0.0%          0.0
Bundle-Ether1302               0:03         0.0      0.0%         0.0          0.0
      0.0%          0.0
Bundle-Ether1303               0:03         0.0      0.0%         0.0      38741.7
     28.8%       8070.8
```

Frequency Synchronization and SyncE

This topic describe details about Synchronous Ethernet (SyncE) on Cisco 8000 Series Routers for accurate network timing distribution.

Cisco IOS XR Software supports SyncE-capable Ethernet on the router. The Synchronous Ethernet (SyncE) is a network synchronization technology that

- distributes precision clock signals across network infrastructures
- injects highly accurate timing signals from external sources into routers, and
- enables peer routers to recover and transfer frequency signals through physical interfaces.

Cisco IOS XR software provides several capabilities for managing SyncE:

- The system utilizes external timing technologies such as Cesium atomic clocks or GPS to maintain accuracy.
- The software passes timing signals to the physical interfaces of the router.
- Peer routers recover the precision frequency from the line to transfer it throughout the network.
- The feature supports Ethernet-based synchronization on Cisco 8000 Series Routers.

This feature is traditionally applicable to SONET or SDH networks, but is now available on Ethernet for Cisco 8000 Series Routers with SyncE capability. For more information, see *Cisco 8000 Series Router System Management Configuration Guide*.

5 MTU

Topics:

This chapter describes the maximum transmission unit (MTU) on Cisco 8000 Series Routers, including Ethernet MTU that governs the largest frame size on the Ethernet network, IP MTU that governs the largest IP packet size without fragmentation, MTU checks and scale limits, platform-specific configuration guidelines, and the procedure to configure IPv4 and IPv6 MTU on an interface.

Ethernet MTU

This topic describes Ethernet Maximum Transmission Unit (MTU), fragmentation, and MTU discovery processes for Cisco IOS XR software-based network configurations.

The Ethernet Maximum Transmission Unit (MTU) is a network parameter that

- represents the size of the largest frame excluding the 4-byte frame check sequence, varies across different physical networks along a packet's path, and
- facilitates efficient data transmission across networks.

Frame forwarding processes

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

- Fragmentation for IPv4 packets fragments packets as necessary to fit within the MTU of the next-hop physical network.
- 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.

Default Ethernet MTU values

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).

Restrictions for Ethernet MTU

This topic describes Ethernet MTU limitations and supported IPv4 and IPv6 configurations for BVI interfaces in Cisco IOS XR software.

These restrictions apply to ethernet MTU:

- Cisco IOS XR software does not support Ethernet MTU on BVI interfaces.
- IPv6 does not support fragmentation.
- Cisco IOS XR software supports **IPv4 mtu** and **IPv6 mtu** commands on BVI interfaces for Layer 3 to Layer 2 direction.
- Cisco IOS XR software supports packet fragmentation for traffic routed over BVI interfaces for Layer 3 to Layer 2 direction.

IP MTU

This topic defines IP MTU, explains differences from Ethernet MTU, provides calculation guidelines, and introduces Maxinum Receive Unit (MRU) for Cisco routers.

In IP protocol, Maximum Transmission Unit (MTU) refers to the maximum size of an IP packet. The IP MTU is a network parameter that

- defines the maximum size of an IP packet transmitted without fragmentation
- includes IP headers while excluding data link layer headers, also known as Ethernet headers, and

- defaults to 1500 bytes on router interfaces when IP is enabled. However, you can configure the IP MTU to different value as well.

The system supports IP MTU, IPv4 and IPv6, on Q200-based systems on these Cisco 8000 Series router and line card(s):

- 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 Layer 2 (L2) value that you configure on a physical interface is also applied as the MRU of that physical interface.

IP MTU support across platforms

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

Table 10: 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.
P100-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

This topic describes MRU, Ethernet MTU, and IP MTU checks for packet forwarding and fragmentation across supported systems.

Cisco IOS XR supports

- MRU checks
- Ethernet MTU checks
- 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.

This table describes IP MTU check to each forwarding flow on different systems.

Table 11: 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.
P100-based	These interfaces implement IP MTU checks and provide fragmentation for all IP payload packets.

IP MTU scale

This topic describes IP MTU scale support, profile limitations, and out-of-resource behavior on Cisco platforms.

These 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 continuing to use the previously configured IP MTU values.

IP MTU configuration guidelines for Q200-based systems

This topic describes IP MTU configuration guidelines, derived values, and platform behavior for Q200-based and P100-based systems.

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.

- 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.

> **Note**
>
> P100-based systems follow the same guidelines as Q200-based systems.

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 12: 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
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

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*
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

IP MTU configuration guidelines for Q100-based systems

This topic describes IP MTU guidelines, default values, and configuration limitations for Q100-based systems.

These are the IP MTU configuration 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 13: 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

Interface type	Default configurations	Interface MTU configurations *where, MTU = 1614*
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	

Limitations for IP MTU

This topic describes IP MTU limitations and unsupported features on Cisco 8000 Series Routers running IOS XR.

These 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 CLI command for the unsupported features might be available on your router, they are not functional.

Configure IP MTU

This topic explains how to configure IPv4 or IPv6 MTU on an L3 interface in Cisco IOS XR.

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

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router#configure terminal
   ```

2. Use **interface** *interface-path-id* to enter interface configuration mode for the Layer 3 interface. **commit**, **end**

   Example

   ```
   Router#interface HundredGigE0/0/0/33
   ```

3. Use the **ipv4 mtu** *mtu-value* command to configure the IPv4 MTU value.

   Example

   ```
   Router(config-if)# ipv4 mtu 9642
   ```

   To configure MTU value for IPv6, use **ipv6 mtu** *mtu-value* command.

4. Save the configuration.

   Example

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

5. Use **show ipv4 interface** *interface-path-id*, **show interfaces** *interface-path-id* commands to verify the configuration

   Example

   ```
   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)
   ```

   ```
   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#
```

Verifies that the configured IP MTU value is available to IP on the interface.

Verify these output values

- `Internet address is 194.19.242.94/29`.
- `MTU 9642 bytes` appears in the interface output.

6 LLDP

Topics:

This chapter the Link Layer Discovery Protocol (LLDP) chapter, which describes how the router uses IEEE 802.1AB LLDP as a neighbor discovery protocol to advertise device information over Layer 2, explains features, such as LLDP frame and TLV formats, user-defined TLV values, LLDP operation, and so on.

Note

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

Link Layer Discovery Protocol

Introduces the IEEE 802.1AB Link Layer Discovery Protocol (LLDP), which the router supports in addition to Cisco Discovery Protocol (CDP) to enable interoperability with non-Cisco devices.

The Cisco Discovery Protocol (CDP) is a device discovery protocol that

- runs over Layer 2 on all Cisco-manufactured devices, such as routers, bridges, access servers, and switches
- allows network management applications to automatically discover and learn about Cisco devices, and
- supports network visibility.

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 a neighbor discovery protocol that

- supports interoperability between non-Cisco devices
- advertises device information to other network devices, and
- operates over the Data Link Layer.

Feature history

Feature name	Release information	Feature description
LLDP snooping	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
LLDP snooping	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature name	Release information	Feature description
LLDP snooping	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM
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.

TLV attributes and mandatory TLVs

LLDP uses a set of attributes known as Type-Length-Value (TLV) to exchange information between neighbor devices, such as configuration information, device capabilities, and device identity. In addition to the mandatory TLVs, such as Chassis ID, Port ID, End of LLDPDU, and Time-to-Live, the router supports the following basic management TLVs:

- 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.

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

Describes how LLDP frames use the IEEE 802.3 format.

LLDP frames use the IEEE 802.3 format, which consists of these 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

Describes how LLDP TLVs carry information about neighboring devices within the LLDP PDU.

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

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

LLDP operation

Describes LLDP as a one-way protocol in which a device that is enabled for transmit sends periodic LLDP frames to a receiving device.

The Link Layer Discovery Protocol (LLDP) is a one-way protocol that

- sends periodic advertisements in LLDP frames to receiving devices
- identifies devices using a combination of Chassis ID and Port ID to create an MSAP (MAC Service Access Point), and
- stores neighbor information in a remote cache until the TTL expires.

Operational characteristics

LLDP supports these 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 during these scenarios:
  - 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

This topic explains LLDP functions, management address advertisement, priority order, and interface support for Cisco routers.

The router supports these LLDP functions:

- The router advertises both IPv4 and IPv6 addresses if they are available, and the system gives preference 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/RP0/CPU0/0
  - MgmtEth0/RP0/CPU0/1
  - MgmtEth0/RP1/CPU0/0
  - MgmtEth0/RP1/CPU0/1
  - Loopback interfaces

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

- The router populates the TLV with an address from another interface if the transmitting interface does not have a configured address.
- 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

This topic explains LLDP functions, management address advertisement, priority order, and interface support for Cisco routers.

These are the unsupported LLDP functions 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.
- LLDP is not supported on the FP-X line cards.

LLDP default configuration

This topic lists the default values of LLDP configuration on the router.

LLDP default values

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.

User-defined LLDP TLV values

Describes how you can override the system default values for some of the mandatory LLDP Type-Length-Values (TLVs) that routers advertise to their directly connected neighboring devices so that they can assign user-defined meaningful names for their identity and capabilities, lists the **lldp system-name**, **lldp system-description**, **lldp chassis-id-type**, and **lldp chassis-id** commands that assign these values, and lists the chassis-id-type values that are objects in the MIB and become part of the TLVs in LLDP packets sent to neighbors.

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. The LLDP TLV values are network management settings that

- allow routers to override default system values
- enable the assignment of user-defined identification names, and
- facilitate the transmission of custom identity data to neighboring devices.

While advertising their identity and capabilities, routers can assign user-defined meaningful names instead of autogenerated values. Using these CLIs, you can specify the 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.

chassis-id-type values

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

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*.

Example: Configure user-defined LLDP TLVs

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
```

The following running configuration shows the resulting **show lldp** output on the router:

```
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
```

Enable LLDP globally

Enable LLDP on the router at the global level so that all interfaces that support LLDP are automatically enabled for both transmit and receive operations, and configure the LLDP holdtime, reinit, and timer attributes to control the neighbor holdtime, initialization delay, and packet rate.

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 *Disable 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)

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router#configure
   ```

2. Enable LLDP globally with the **lldp** command.

   Example

   ```
   Router(config) # lldp
   ```

3. Save your configuration with **end** or **commit**.

   Example

   ```
   Router(config)# commit
   ```

4. Use **show run lldp** command to check the running configuration.

   Example

   ```
   Router#show run lldp
   Fri Dec 15 20:36:49.132 UTC
   lldp
   !

   Router#show lldp neighbors
   Fri Dec 15 20:29:53.763 UTC
   Capability codes:
           (R) Router, (B) Bridge, (T) Telephone, (C) DOCSIS Cable Device
           (W) WLAN Access Point, (P) Repeater, (S) Station, (O) Other

   Device ID       Local Intf          Hold-time  Capability     Port ID
   SW-NOSTG-I11-PUB.cis Mg0/RP0/CPU0/0      120        N/A            Fa0/28

   Total entries displayed: 1

   Router#show lldp neighbors mgmtEth 0/RP0/CPU0/0
   Fri Dec 15 20:30:54.736 UTC
   Capability codes:
           (R) Router, (B) Bridge, (T) Telephone, (C) DOCSIS Cable Device
   ```

```
        (W) WLAN Access Point, (P) Repeater, (S) Station, (O) Other

Device ID       Local Intf          Hold-time  Capability     Port ID
SW-NOSTG-I11-PUB.cis Mg0/RP0/CPU0/0      120        N/A            Fa0/28

Total entries displayed: 1
```

Example

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
```

Enable LLDP per interface

This topics describes how to enable LLDP on a specific interface so that LLDP transmit and receive operations run only on the interfaces where you have activated LLDP explicitly.

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.

Procedure

1. Enter interface configuration mode for the target Ethernet interface and bring it up.

   Example

   ```
   Router(config)# int HundredGigE 0/2/0/0
   Router(config-if)# no sh
   Router(config-if)# commit
   ```

2. Enable LLDP on the interface with the **lldp enable** command and commit the configuration.

   Example

   ```
   Router(config-if)# lldp ?
   Router(config-if)# lldp enable
   Router(config-if)# commit
   ```

3. Use **show running-config** command to check the running configuration.

   ```
   Router#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 HundredGigE0/0/0/0
    shutdown
   !
   ```

```
interface HundredGigE0/0/0/1
 shutdown
!
interface HundredGigE0/0/0/2
 shutdown
!
interface HundredGigE0/0/0/3
 Shutdown
!
interface HundredGigE0/0/0/4
 shutdown
!
interface HundredGigE0/0/0/5
 shutdown
!
end
```

Configure global LLDP operational characteristics

This topic describes how to modify the default global LLDP operational characteristics on the router, such as the LLDP neighbor information holdtime, the LLDP initialization delay for an interface, and the LLDP packet rate.

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

Use these steps to modify the global LLDP operational characteristics such as the LLDP neighbor information holdtime, initialization delay, or packet rate,.

Procedure

1. Enter the global configuration mode.

   Example

   ```
   Router# configure
   ```

2. (Optional) Use the **lldp holdtime** *seconds* command to specify the length of time that information from an LLDP packet should be held by the receiving device before aging and removing it.

   Example

   ```
   Router(config)#lldp holdtime 60
   ```

3. (Optional) Use the **lldp reinit** *seconds* command to specify the length of time to delay initialization of LLDP on an interface.

   Example

   ```
   Router(config)# lldp reinit 4
   ```

4. (Optional) Use the **lldp timer** *seconds* command to specify the LLDP packet rate.

Example

```
Router(config)#lldp reinit 60
```

5. Save the configuration changes.

Example

```
Router(config)# end
```

or

```
Router(config)# commit
```

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

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

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

Disable transmission of optional LLDP TLVs

This topic describes how to suppress transmission of certain 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

1. Enter global configuration mode.

Example

```
Router# configure
```

2. (Optional)Use **lldp tlv-select** *tlv-name* **disable** to specify that transmission of the selected TLV in LLDP packets is disabled.

Example

```
Router(config)# lldp tlv-select system-capabilities disable
```

The *tlv-name* can be one of the following LLDP TLV types:

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

3. Save the configuration changes.

Example

```
Router(config)# end
```

or

```
Router(config)# commit
```

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

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

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

Disable LLDP receive and transmit operation for an interface

This topic describes how to override the default behavior of LLDP on a specific interface by disabling the LLDP receive operation, transmit operation, or both.

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

1. Enter global configuration mode.

   Example

   ```
   Router#configure
   ```

2. Enter the interface configuration mode and specify the Ethernet interface name. Use the *rack/slot/module/port* notation.

   Example

   ```
   Router(config)#interface HundredGigE 0/2/0/0
   ```

   Possible interface types for this procedure are:

   - HundredGigE
   - TenGigE

3. (Optional) Enter the LLDP configuration mode for the specified interface.

   Example

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

4. (Optional) Disable the LLDP receive operations on the interface.

   Example

   ```
   Router(config-lldp)#receive disable
   ```

5. (Optional) Disable the LLDP transmit operations on the interface.

   Example

   ```
   Router(config-lldp)#transmit disable
   ```

6. Save the configuration changes.

   Example

   ```
   Router(config)# end
   ```

   or

   ```
   Router(config)# commit
   ```

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

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

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

Enable LLDP snoop

This topic describes how to enable Link Layer Discovery Protocol (LLDP) snoop on an L2 interface to troubleshoot problems at the client ports.

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:

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router#configure
   ```

2. Enter interface configuration mode for the target L2 interface.

   Example

   ```
   Router(config)# interface FourHundredGigE 0/0/0/5
   ```

3. Enter LLDP configuration mode for the interface.

   Example

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

4. Enable LLDP on the interface.

Example

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

5. Disable LLDP transmit operation on the interface.

Example

```
Router(config-if)#transmit disable
```

6. Commit the configuration.

Example

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

7. Verify the configuration.

Example

```
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#
```

```
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#
```

LLDP interface configuration verification

Verify the LLDP interface status and configuration using the **show lldp interface** command, monitor and maintain LLDP on the system with commands such as **clear lldp counters**, **show lldp entry**, **show lldp errors**, **show lldp neighbors**, and **show lldp traffic**, and collect or clear LLDP interface statistics with the **show lldp traffic interface** and **clear lldp counters interface** commands.

Verify the LLDP interface configuration

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

```
Router# show lldp interface HundredGigE 0/1/0/7
Wed Apr 13 13:22:30.501 DST


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

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

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

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
```

7 Configure Ethernet Link OAM

Topics:

This chapter describes the Ethernet Link OAM (IEEE 802.3ah), which covers neighbor discovery, error disable forwarding, MIB retrieval, miswiring detection, SNMP traps, link monitoring, and remote loopback, and includes tasks to configure an Ethernet OAM profile.

Ethernet Link OAM configuration

This topic describes how Ethernet Link OAM (IEEE 802.3ah) operates on a single physical link so that service providers can monitor connection quality, watch for specific events, and take actions on those events.

Ethernet as a Metro Area Network (MAN) or a Wide Area Network (WAN) technology benefits greatly from the implementation of Operations, Administration and Maintenance (OAM) features. The Ethernet Link OAM is a management mechanism that

- allows Service Providers to monitor connection quality on MANs or WANs
- tracks specific events, and
- enables event-based actions.

Table 14: Feature History Table

Feature Name	Release Information	Feature Description
Ethernet Link OAM on Physical Interface– (802.3ah) Link Monitoring	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Ethernet Link OAM on Physical Interface– (802.3ah) Link Monitoring	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100])(select variants only*) *This feature is now supported on: - 8011-32Y8L2H2FH - 8011-12G12X4Y-A/D - 8711-48Z-M
Ethernet Link OAM support	Release 25.2.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100]))(select variants only*) This release supports: - Ethernet Link OAM on physical interface– (802.3ah) link monitoring. *This feature is now supported on Cisco 8712-MOD-M routers.
Ethernet Link OAM on Physical Interface– (802.3ah) Link Monitoring	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
Ethernet Link OAM on Physical Interface– (802.3ah) Link Monitoring	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) Ethernet link OAM operates on a single, physical link and it can be configured to monitor either side or both sides of that link. Ethernet OAM supports link monitoring: *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM
Ethernet Link OAM	Release 7.3.1	This feature allows service providers to monitor the quality of the connections on a MAN or WAN. Service providers can monitor specific events, and take actions on events. Ethernet link OAM operates on a single, physical link and it can be configured to monitor either side or both sides of that link.

You can configure Ethernet link OAM in these ways:

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

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

An Ethernet Link OAM profile simplifies the configuration process for EOAM features across multiple interfaces. An Ethernet OAM profile and its features allow other interfaces to reference and inherit the defined settings. You can configure individual Ethernet link OAM features on specific interfaces without using a profile, and these individual configurations always override the profile features. The preferred method for configuring custom EOAM settings involves creating an EOAM profile in Ethernet configuration mode and attaching it to one or more interfaces. These standard Ethernet Link OAM features are supported on the router.

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

- Neighbor Discovery
- Ethernet Fault Detection
- MIB Retrieval

- Miswiring Detection
- SNMP Traps
- Link Monitoring
- Remote Loopback

Neighbor discovery

This topic describes how each end of an Ethernet Link OAM link learns the OAM capabilities of the other end and establishes an OAM peer relationship, and how you configure actions to take when there is a capabilities conflict or the discovery process times out.

The Neighbor discovery is an OAM process that

- enables each end of a link to learn the OAM capabilities of the other end
- establishes an OAM peer relationship, and
- enforces specific capability requirements before the initiation of a session.

You can configure certain actions to be taken if there is a capabilities conflict or if a discovery process times out by using the **action capabilities-conflict** or **action discovery-timeout** commands.

Ethernet Fault Detection

This topic describes how Ethernet Fault Detection (EFD) lets Ethernet OAM protocols control the line protocol state of an Ethernet interface so that when a defect such as Alarm Indication Signal (AIS) or loss of continuity is detected, the interface is shut down to stop traffic flow.

Ethernet Fault Detection (EFD) is a mechanism that

- allows Ethernet OAM protocols to control the `line protocol` state of an interface
- enables EOAM to shut down an interface when it detects a defect with an expected peer MEP, and
- triggers higher-level protocols to react and reconverge around the problem.

How EFD works

This topic describes how Ethernet Fault Detection (EFD) works.

Unlike many other interface types, Ethernet interfaces do not have an independent line protocol; the physical-layer Ethernet protocol itself determines whether the interface is available. EFD lets EOAM act as the line protocol for Ethernet interfaces, so that EOAM defects such as Alarm Indication Signal (AIS) or loss of continuity can shut down the interface and trigger higher-level protocols to reconverge.

- EFD can only be used for down MEPs.
- When EFD shuts down the interface, EOAM frames continue to flow so that EOAM can detect when the problem is resolved and bring the interface back up automatically.

Summary

The following actors and components participate in this process:

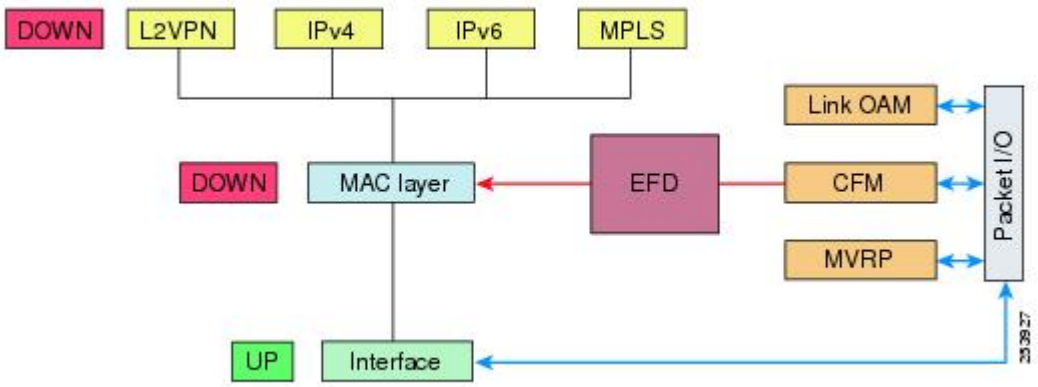
- EOAM: Detects defects on its sessions with the expected peer MEP, and signals EFD when a defect is detected or cleared.
- EFD: Controls the line protocol state of the Ethernet interface based on signals from EOAM.

- MAC layer: Transitions to the down state when EFD signals an error, and back to the up state when EFD signals recovery.
- Higher-level protocols: Layer 2 pseudowires, IP protocols, MSTP, IGP, and similar protocols that react to the MAC state change and reconverge around the fault.

When EOAM detects an error on one of its sessions, EFD signals the MAC layer to go down. This triggers higher-level protocols to reconverge. When EOAM detects that the fault is cleared, EFD reverses the signal and all protocols become active again.

Workflow

Figure 2: EOAM Error Detection and EFD Trigger



These stages describe how EFD reacts to an EOAM defect and to its subsequent recovery:

1. Stage 1 – EOAM detects a defect. EOAM identifies a defect (such as AIS or loss of continuity) on one of its sessions with an expected peer MEP.
2. Stage 2 – EFD signals the MAC layer. EFD receives the defect notification from EOAM and signals the error to the corresponding MAC layer for the interface.
3. Stage 3 – MAC goes down and higher-level protocols reconverge. The MAC transitions to the down state, which triggers higher-level protocols to go down and reconverge where possible.
   - For Layer 2 interfaces, the MAC table is cleared and MSTP reconverges.
   - For Layer 3 interfaces, the ARP cache is cleared and, potentially, the IGP reconverges.
4. Stage 4 – EOAM detects clearance and EFD reactivates the interface. Because EOAM frames continue to flow while the interface is shut down, EOAM can detect when the defect is resolved. EOAM signals EFD, and all protocols become active again.

MIB retrieval

This topic describes how an OAM peer on one side of an interface retrieves MIB variables from the remote side of the link.

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

Miswiring detection (Cisco-proprietary)

This topic describes the Cisco-proprietary miswiring detection feature.

Miswiring Detection is a Cisco-proprietary feature that uses the 32-bit vendor field in every Information OAMPDU to identify potential miswiring cases.

SNMP traps

This topic describes support for enabling or disabling Simple Network Management Protocol (SNMP) traps.

SNMP traps can be enabled or disabled on an Ethernet Operations, Administration, and Maintenance (OAM) interface.

Link monitoring

This topic describes how link monitoring lets an OAM peer detect faults that degrade the quality of a link.

Link monitoring enables an OAM peer to monitor faults that cause the quality of a link to deteriorate over time. When you enable link monitoring, an OAM peer can be configured to take action when the configured thresholds are exceeded.

Remote loopback

This topic describes how remote loopback lets one side of a link put the remote side into loopback mode for testing.

Remote loopback enables one side of a link to put the remote side of the link into loopback mode for testing. When remote loopback is enabled, all packets initiated by the primary side of the link are looped back to the primary side, unaltered by the remote side. In remote loopback mode, the remote side is not allowed to inject any data into the packets.

Configuration guidelines for Ethernet Link OAM

This topic describes the guidelines to follow when you configure Ethernet Link OAM, including the preferred use of Ethernet OAM profiles, how interface-level settings override profile settings, and the accepted values for link-monitor window sizes.

Use an Ethernet OAM profile for multiple interfaces

- Create an Ethernet OAM profile in Ethernet configuration mode and attach it to each interface that requires the same settings, instead of configuring Ethernet OAM features directly on each interface.
- An Ethernet OAM profile and its features can be referenced by other interfaces, which lets those interfaces inherit the settings from the profile. This method simplifies the configuration of EOAM features across multiple interfaces.

Interface configuration overrides profile settings

- When an interface is attached to an Ethernet OAM profile and one or more Ethernet OAM features are also configured directly on that interface, the interface-level configuration takes effect and overrides the profile settings for those features.
- This principle applies when an interface uses an Ethernet OAM profile and also has one or more Ethernet OAM features configured directly on it.

Configure link-monitor window sizes as multiples of the polling interval

- The IEEE 802.3 standard defines the window size as a number of symbols rather than a time duration. These two formats can be converted either way by using a knowledge of the interface speed and encoding.
- The default value is 1000.
- If specified in milliseconds, the range is 1000 to 60000.

- If not specified as a multiple of 1 second, the actual window used is rounded up to the nearest second, with thresholds scaled accordingly.
- If specified in symbols, the range is interface speed dependent (must be between the maximum number of symbols that could be received in 1 second and the maximum number of symbols that could be received in 1 minute).
- Again the actual window used is rounded up to the nearest second, with thresholds scaled accordingly.
- The default value is 1000 milliseconds.

Guidelines for symbol-period threshold { ppm[low threshold] [high threshold] | symbols [low threshold [thousand | million | billion]] [high threshold [thousand | million | billion]]} command

- At least one of the high and low thresholds must be specified.
- If the low threshold is not specified, the default value is used.
- If the high threshold is not specified, no action is performed in response to an event.
- The high threshold must not be smaller than the low threshold.
- If specified in ppm, the range (for both thresholds) is 1 to 1000000. If specified in symbols, the range (for both thresholds) is 1 to the maximum window size in symbols, see **symbol-period window**.
- The default low threshold is 1 symbol.

Guidelines for frame-period window { milliseconds window | frames window [thousand | million | billion]} command

- The range is from 100 to 60000, if defined in milliseconds.
- If the window is defined as say, 200ms, and the interface could receive at most say 10000 minimum size frames in 200ms, then the actual window size used will be the time taken to receive 10000 frames, rounded up to the nearest second. The thresholds will be scaled accordingly.
- If specified in frames, the range is interface speed dependent, but must be between the number of minimum size frames that could be received in 100ms and the number of minimum size frames that could be received in 1 minute.
- If the window is defined as 20000 frames, the actual window size used will be the time taken to receive 20000 frames, rounded up to the nearest second. The thresholds will be scaled accordingly.
- The default value is 1000 milliseconds.

Guidelines for frame-period threshold { ppm [low threshold] [high threshold] | frames [low threshold [thousand | million | billion]] [high threshold [thousand | million | billion]]} command

- When using this command at least one of the high and low thresholds must be specified.
- If the low threshold is not specified, the default value is used.
- If the high threshold is not specified, no action is performed in response to an event.
- The high threshold must not be smaller than the low threshold.
- The range for both thresholds is from 1 to 1000000 if specified in ppm. If specified in frames, the range is from 1 to the maximum frame-period window size in frames, see **frame-period window**.
- The default low threshold is 1 ppm.

Guidelines for frame-seconds threshold [low threshold] [high threshold] command

- When using this command at least one of the high and low thresholds must be specified.
- If the low threshold is not specified, the default value is used.

- If the high threshold is not specified, no action is performed in response to an event.
- The high threshold must not be smaller than the low threshold.
- The range is 1 to 900
- The default value is 1.

Configure an Ethernet OAM profile

This topic describes how to configure an Ethernet OAM profile by initializing the profile, configuring link monitoring error events, configuring session parameters and remote requirements, configuring interface actions for OAM events, and finalizing the OAM configuration.

You can configure custom EOAM settings in an EOAM profile in Ethernet configuration mode and apply the profile to multiple interfaces by attaching it to individual interfaces. The profile configuration takes effect only after you attach the profile to an interface. After you attach the profile, you can configure individual EOAM features on the interface to override the profile settings when needed.

Perform these steps to configure an Ethernet OAM profile.

Procedure

1. Initialize an Ethernet OAM profile.
2. Configure link monitoring error events.
3. Configure session parameters and remote requirements.
4. Configure interface actions for OAM events.
5. Finalize the OAM configuration.

Initialize an Ethernet OAM profile

This topic describes how to initialize an Ethernet OAM profile by entering global configuration mode, creating a new Ethernet OAM profile, and entering the Ethernet OAM link monitor configuration mode.

Perform these steps to initialize an Ethernet OAM profile before you configure link monitoring, session parameters, and interface actions.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure terminal
   ```

2. Create a new Ethernet Operations, Administration and Maintenance (OAM) profile and enter Ethernet OAM configuration mode by using the **ethernet oam profile** *profile-name* command.

Example

```
Router(config)# ethernet oam profile Profile_1
```

Configure link monitoring error events

This topic describes how to configure link monitoring error events in an Ethernet OAM profile by setting the window sizes and thresholds.

Perform these steps in Ethernet OAM link monitor configuration mode to configure the window sizes and thresholds that trigger symbol-period, frame, frame-period, and frame-seconds error events.

Before you begin

Refer the configuration guidelines.

Procedure

1. Enter the Ethernet OAM link monitor configuration mode.

   Example

   ```
   Router(config-eoam)# link-monitor
   ```

2. Optional: Configure the window size (in milliseconds) for an Ethernet OAM symbol-period error event by using the **symbol-period window** { **milliseconds** *window* | **symbols** *window* [**thousand** | **million** | **billion**]} command.

   Example

   ```
   Router(config-eoam-lm)# symbol-period window 60000
   ```

3. Configure the thresholds that trigger an Ethernet OAM symbol-period error event, in symbols or ppm (errors per million symbols) by using the **symbol-period threshold** { **ppm**[**low** *threshold*] [**high** *threshold*] | **symbols** [**low** *threshold* [**thousand** | **million** | **billion**]] [**high** *threshold* [**thousand** | **million** | **billion**]]} command.

   Example

   ```
   Router(config-eoam-lm)# symbol-period threshold ppm low 100 high 1000000
   ```

4. Optional: Configure the frame window size (in milliseconds) of an OAM frame error event by using the **frame window milliseconds** *window*

   Example

   ```
   Router(config-eoam-lm)# frame window milliseconds 60
   ```

   - The range is from 1000 to 60000.
   - The default value is 1000.

5. Optional: Configure the thresholds (in symbols) that trigger an Ethernet OAM symbol-period error event by using the **symbol-period threshold low** *threshold* **high** *threshold*command.

Example

```
Router(config-eoam-lm)# symbol-period threshold low 10000000 high 60000000
```

High threshold is optional and is configurable only in conjunction with low threshold.

- The range is 1 to 60000000.
- The default low threshold is 1.

6. Configure the window size (in milliseconds) for an Ethernet OAM frame-period error event by using the **frame-period window** { **milliseconds** *window* | **frames** *window* [**thousand** | **million** | **billion**]} command.

Example

```
Router(config-eoam-lm)# frame-period window milliseconds 60000
```

7. Configure the thresholds (in errors per million frames) that trigger an Ethernet OAM frame-period error event by using the **frame-period threshold** { **ppm** [**low** *threshold*] [**high** *threshold*] | **frames** [**low** *threshold* [**thousand** | **million** | **billion**]] [**high** *threshold* [**thousand** | **million** | **billion**]]} command.

Example

```
Router(config-eoam-lm)# frame-period threshold ppm low 100 high 1000000
```

8. Configure the window size (in milliseconds) for the OAM frame-seconds error event by using the **frame-seconds window milliseconds** *window* command.

Example

```
Router(config-eoam-lm)# frame-seconds window milliseconds 900000
```

- The range is 10000 to 900000.
- The default value is 6000.

9. Configure the thresholds (in seconds) that trigger a frame-seconds error event by using the **frame-seconds threshold** [**low** *threshold*] [**high** *threshold*] command and exit back to Ethernet OAM mode.

Example

```
Router(config-eoam-lm)# frame-seconds threshold low 3 threshold high 900
Router(config-eoam-lm)#exit
```

- The range is 1 to 900
- The default value is 1.

Configure session parameters and remote requirements

This topic describes how to configure Ethernet OAM session parameters such as MIB retrieval, connection timeout, hello interval, and mode, and specify remote requirements for the OAM session to become active.

Use this procedure in Ethernet OAM configuration mode to configure the session parameters and remote requirements for an Ethernet OAM profile.

Procedure

1. Enable MIB retrieval in an Ethernet OAM profile or on an Ethernet OAM interface.

   Example

   ```
   Router(config-eoam)# mib-retrieval
   ```

2. Configure the connection timeout period for an Ethernet OAM session by using the **connection timeout** *<timeout>* command.

   Example

   ```
   Router(config-eoam)# connection timeout 30
   ```

   - The range is 2 to 30.
   - The default value is 5.

3. Configure the time interval between hello packets for an Ethernet OAM session. The default is 1 second.

   Example

   ```
   Router(config-eoam)# hello-interval 1s
   ```

4. Configure the Ethernet OAM mode by using the **mode {active|passive}** command. The default is active.

   Example

   ```
   Router(config-eoam)# mode passive
   ```

5. Configure the **require-remote mode {active|passive}** mode on the remote end before the OAM session becomes active.

   Example

   ```
   Router(config-eoam)# require-remote mode active
   ```

6. Configure **require-remote mib-retrieval** on the remote end before the OAM session becomes active.

   Example

   ```
   Router(config-eoam)# require-remote mib-retrieval
   ```

Configure interface actions for OAM events

This topic describes how to specify the actions taken on an interface for Ethernet OAM events, such as capabilities-conflict, critical-event, and discovery-timeout.

Use this procedure in Ethernet OAM configuration mode to specify the action taken on an interface for each Ethernet OAM event.

Procedure

1. Use the **action capabilities-conflict {disable | efd | error-disable-interface | log}** command to specify the action that is taken on an interface when a capabilities-conflict event occurs.

   Example

   ```
   Router(config-eoam)# action capabilities-conflict efd
   ```

   The default action is to create a syslog entry.

2. Use the **action critical-event {disable | error-disable-interface | log}** command to specify the action that is taken on an interface when a critical-event notification is received from the remote Ethernet OAM peer.

   Example

   ```
   Router(config-eoam)# action critical-event error-disable-interface
   ```

   The default action is to create a syslog entry.

3. Use the **action discovery-timeout {disable | efd | error-disable-interface | log}** command to specify the action that is taken on an interface when a connection timeout occurs.

   Example

   ```
   Router(config-eoam)# action discovery-timeout efd
   ```

   The default action is to create a syslog entry.

4. Use the **action dying-gasp {disable | error-disable-interface | log}** command to specify the action that is taken on an interface when a dying-gasp notification is received from the remote Ethernet OAM peer.

   Example

   ```
   Router(config-eoam)# action dying-gasp error-disable-interface
   ```

   The default action is to create a syslog entry.

5. Use the **action high-threshold {disable | error-disable-interface | log}** command to specify the action that is taken on an interface when a high threshold is exceeded.

   Example

   ```
   Router(config-eoam)# action high-threshold error-disable-interface
   ```

   The default is to take no action when a high threshold is exceeded.

6. Use the **action session-down {disable | efd | error-disable-interface | log}** command to specify the action that is taken on an interface when an Ethernet OAM session goes down.

Example

```
Router(config-eoam)# action session-down efd
```

7. Use the **action session-up { disable | log }** command to pecify that no action is taken on an interface when an Ethernet OAM session is established.

   Example

   ```
   Router(config-eoam)# action session-up disable
   ```

   The default action is to create a syslog entry.

8. Use the **action uni-directional link-fault {disable | efd | error-disable-interface | log}** command to specify the action that is taken on an interface when a link-fault notification is received from the remote Ethernet OAM peer.

   The default action is to create a syslog entry.

9. Use the **action wiring-conflict {disable | efd | error-disable-interface | log}** command to specify the action that is taken on an interface when a wiring-conflict event occurs.

   Example

   ```
   Router(config-eoam)# action session-down efd
   ```

   The default is to put the interface into error-disable state.

Finalize the OAM configuration

This topic describes how to finalize the Ethernet OAM configuration by enabling detection of local unidirectional link faults, committing the configuration changes, and ending the configuration session to exit to EXEC mode.

Perform these steps to finalize the Ethernet OAM configuration.

Procedure

1. Enable detection of a local, unidirectional link fault and send notification of that fault to an Ethernet OAM peer.

   Example

   ```
   Router(config-eoam)# uni-directional link-fault detection
   ```

2. Save the configuration changes to the running configuration file. The changes remain within the configuration session.

   Example

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

3. End the configuration session and exit to the EXEC mode.

Example

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

Attach an Ethernet OAM profile to an interface

This topic describes how to enable Ethernet OAM on an Ethernet interface and use the profile command in interface Ethernet OAM configuration mode to attach a previously configured Ethernet OAM profile so that the interface inherits all of the profile settings.

Use this procedure to attach an Ethernet OAM profile to an interface.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure terminal
   ```

2. Enter the interface configuration mode and specify the Ethernet interface name and notation *rack/slot/module/port*.

   Example

   ```
   Router(config)# interface TenGigE 0/1/0/0
   ```

   The example indicates an 8-port 10-Gigabit Ethernet interface in modular services card slot 1.

3. Enable Ethernet OAM and enter interface Ethernet OAM configuration mode.

   Example

   ```
   Router(config-if)# ethernet oam
   ```

4. Attach the specified Ethernet OAM profile (*profile-name*), and all of its configuration to the interface.

   Example

   ```
   Router(config-if-eoam)# profile Profile_1
   ```

5. Save the configuration changes to the running configuration file. The saved changes remain within the configuration session.

   Example

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

6. End the configuration session and exit to the EXEC mode.

Example

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

Configure Ethernet OAM at an interface and override the profile configuration

This topic describes how to enable Ethernet OAM on a specific interface with the ethernet oam command and then use interface Ethernet OAM configuration mode to override individual command settings that are inherited from the applied Ethernet OAM profile for that interface.

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

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

To see all of the default Ethernet OAM configuration settings, see the *Verify the CFM configuration* section.

Use this procedure to configure Ethernet OAM settings at an interface and override the profile configuration.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure terminal
   ```

2. Enter the interface configuration mode and specify the Ethernet interface name and notation *rack/slot/module/port*.

   Example

   ```
   Router(config)# interface TenGigE 0/1/0/0
   ```

   The example indicates an 8-port 10-Gigabit Ethernet interface in modular services card slot 1.

3. Enable Ethernet OAM and enter interface Ethernet OAM configuration mode.

   Example

   ```
   Router(config-if)# ethernet oam
   ```

4. Configure a setting for an Ethernet OAM configuration command and override the setting for the profile configuration, where *interface-Ethernet-OAM-command* is one of the supported commands on the platform in interface Ethernet OAM configuration mode.

Example

```
Router(config-if-eoam)# action capabilities-conflict error-disable-interface
```

5. Save the configuration changes to the running configuration file and remains within the configuration session.

Example

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

6. End the configuration session and exit to the EXEC mode.

Example

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

Verify the Ethernet OAM configuration

This topic describes how to verify the Ethernet OAM configuration for a specific interface or for all interfaces by running the show ethernet oam configuration command, which displays every configured Ethernet OAM setting including hello interval, mode, connection timeout, and thresholds for symbol-period, frame, frame-period, and frame-seconds error events.

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

```
Router# show ethernet oam configuration
Thu Aug  5 22:07:06.870 DST
GigabitEthernet0/4/0/0:
  Hello interval:                                          1s
  Mib retrieval enabled:                                    N
  Uni-directional link-fault detection enabled:             N
  Configured mode:                                     Active
  Connection timeout:                                       5
  Symbol period window:                                     0
  Symbol period low threshold:                              1
  Symbol period high threshold:                          None
  Frame window:                                          1000
  Frame low threshold:                                      1
  Frame high threshold:                                  None
  Frame period window:                                   1000
  Frame period low threshold:                               1
  Frame period high threshold:                           None
  Frame seconds window:                                 60000
  Frame seconds low threshold:                              1
  Frame seconds high threshold:                          None
  High threshold action:                                 None
  Link fault action:                                      Log
  Dying gasp action:                                      Log
  Critical event action:                                  Log
  Discovery timeout action:                               Log
  Capabilities conflict action:                           Log
  Wiring conflict action:                        Error-Disable
  Session up action:                                      Log
  Session down action:                                    Log
  Require remote mode:                                 Ignore
```

```
  Require remote MIB retrieval:                                  N
```

Ethernet OAM configuration and management examples

The examples covered in this topic illustrate the configuration and management procedures for Ethernet OAM. The collection covers interface setup, global profile application, error-disable recovery, statistics management, and SNMP trap activation.

Example: Configure Ethernet OAM features on an individual interface

This topic shows an example on how to configure Ethernet OAM features directly on an individual TenGigE interface by enabling settings, such as Ethernet OAM, setting the connection timeout and MIB retrieval, and so on.

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

```
configure
 interface TenGigE 0/0/0/0
  ethernet oam
  connection timeout 30
  mib-retrieval
  link-monitor
   frame window milliseconds 60000
   frame threshold low 10000000 high 60000000
   frame-period window milliseconds 60000
   frame-period threshold ppm low 100 high 120000
   frame-seconds window milliseconds 900000
   frame-seconds threshold low 3 high 900
   symbol-period window milliseconds 60000
   symbol-period threshold ppm low 1000000 high 1000000
  exit
  require-remote
   mode active
   mib-retrieval
  exit
  action
   critical-event error-disable-interface
   dying-gasp error-disable-interface
   capabilities-conflict error-disable-interface
   wiring-conflict error-disable-interface
   discovery-timeout error-disable-interface
   session-down error-disable-interface
   commit
```

Example: Configure an Ethernet OAM profile globally

This topic shows an example on how to configure an Ethernet OAM profile globally by specifying settings, such as setting the connection timeout, enabling MIB retrieval, and defining link-monitor window that you can then apply to multiple interfaces.

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

```
configure
ethernet oam profile Profile_1
 connection timeout 30
 mib-retrieval
```

```
 link-monitor
  frame window milliseconds 60000
  frame threshold low 10000000 high 60000000
  frame-period window milliseconds 60000
  frame-period threshold ppm low 100 high 1000000
  frame-seconds window milliseconds 900000
  frame-seconds threshold low 3 high 900
  symbol-period window milliseconds 60000
  symbol-period threshold ppm low 100000 high 1000000
 exit
 require-remote
  mode active
  mib-retrieval
 exit
 action
  critical-event error-disable-interface
  dying-gasp error-disable-interface
  capabilities-conflict error-disable-interface
  wiring-conflict error-disable-interface
  discovery-timeout error-disable-interface
  session-down error-disable-interface
  commit
```

Example: Configure Ethernet OAM features to override the profile on an individual interface

This topic describes how to configure Ethernet OAM mode and action settings in a profile, and then override the settings on an individual TenGigE interface so that other interfaces inherit the profile while this interface uses different mode and action responses for its own OAM events.

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

```
configure
 ethernet oam profile Profile_1
  mode passive
  action dying-gasp disable
  action critical-event disable
  action discovery-timeout disable
  action session-up disable
  action session-down disable
  action capabilities-conflict disable
  action wiring-conflict disable
    commit

configure
 interface TenGigE 0/1/0/0
  ethernet oam
   profile Profile_1
    mode active
    action dying-gasp log
    action critical-event log
    action discovery-timeout log
    action session-up log
    action session-down log
    action capabilities-conflict log
    action wiring-conflict log
      commit
```

Example: Recover from error-disable

This topic shows an example on how to recover an interface that is in the error-disable state due to an Ethernet Link OAM session going down by manually clearing the error-disable condition, by administratively shutting down and re-enabling the interface, or by configuring an auto-recovery timer for the link-oam-session-down cause.

You can recover an error-disabled interface due to session-down using one of these methods:

- Manually clear the error-disable using the **clear** command.

```
Router# configure
Router(config)# ethernet oam profile Profile_1
Router(config-eoam)# action
Router(config-eoam-action)# clear session-down error-disable-interface
```

- Disable and then re-enable the network link using administrative shutdown commands to reset the connection.

```
Router# configure
Router(config)# interface TenGigE 0/1/0/0
Router(config-if)# shutdown
Router(config-if)# commit
Router(config-if)# no shutdown
Router(config-if)# commit
```

- Configure an auto-recovery timer for this error-disable reason.

```
Router# configure
Router(config)# error-disable recovery cause link-oam-session-down interval
30
Router(config)# commit
```

Example: Clear Ethernet OAM statistics on an interface

This topic shows an example on how to clear Ethernet OAM statistics on a specific GigabitEthernet interface by running the clear ethernet oam statistics interface command with the interface identifier.

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

```
Router# clear ethernet oam statistics interface gigabitethernet 0/0/0/1
```

Example: Enable SNMP server traps on a router

This topic shows an example on how to enable SNMP server traps on the router for Ethernet OAM events by running the snmp-server traps ethernet oam events command in global configuration mode.

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

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

8 Configure Ethernet CFM

Topics:

This chapter describes the Ethernet Connectivity Fault Management (CFM) chapter, which covers maintenance domains, services, maintenance points, MEP and CFM processing, protocol messages, continuity check, loopback, linktrace, configurable logging, hardware offload, and CFM over bundles, and includes tasks to configure domains, services, MEPs, cross-check, AIS, and Y.1731 AIS.

Ethernet CFM

This topic describes Ethernet Connectivity Fault Management (CFM), a service-level OAM protocol that provides tools for monitoring and troubleshooting end-to-end Ethernet services per VLAN, including proactive connectivity monitoring, fault verification, and fault isolation using standard Ethernet frames across the entire end-to-end Ethernet network.

Ethernet Connectivity Fault Management (CFM) is a service-level OAM protocol that

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

Table 15: Feature History Table

Feature name	Release	Description
Y.1731 support on CFM	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Up MEP and down MEP support in CFM	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
CFM on bundle member link for connectivity check	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Monitoring Layer 3 connectivity using down MEP on L3 interfaces	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Y.1731 support on CFM	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100])(select variants only*) *This feature is supported on: - 8711-48Z-M - 8011-32Y8L2H2FH - 8011-12G12X4Y-A/D

Feature name	Release	Description
Increase in number of CFM sessions	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
CFM on bundle member link for connectivity check	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8711-48Z-M
Up MEP and down MEP support in CFM	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8711-48Z-M
Monitoring Layer 3 connectivity using down MEP on L3 interfaces	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Up MEP and down MEP support in CFM	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D

Feature name	Release	Description
Y.1731 support on CFM	Release 25.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) This feature is now supported on: • 8011-4G24Y4H-I • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Increase in number of CFM sessions	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
CFM on bundle member link for connectivity check	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is now supported on Cisco 8011-4G24Y4H-I routers.
Monitoring Layer 3 connectivity using down MEP on L3 interfaces	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is now supported on Cisco 8011-4G24Y4H-I routers.
Up MEP and down MEP support in CFM	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature name	Release	Description
Increase in number of CFM sessions	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) The number of supported Connectivity Fault Management (CFM) sessions is now increased to 500. This enhancement improves fault detection, network visibility, scalability, and troubleshooting, which are crucial for managing high-performance networks.
CFM on bundle member link for connectivity check	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) (select variants only*) This feature introduces support for Connectivity Fault Management (CFM) on bundle members. * This feature is supported on Cisco 8712-MOD-M routers.
Up MEP and down MEP support in CFM	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) This feature introduces Maintenance End Points (MEP) entities that you can configure in a domain. *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM

Feature name	Release	Description
Monitoring Layer 3 connectivity using down MEP on L3 interfaces	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100]) (select variants only*); Modular Systems (8800 [LC ASIC: P100]) This enhancement expands network diagnostics to L3 interfaces at L2 network termination, simplifying the management and maintenance of multilayer networks. * This feature is supported on Cisco 8712-MOD-M routers.
Monitoring Layer 3 connectivity using down MEP on L3 interfaces	Release 24.2.1	This enhancement expands network diagnostics to L3 interfaces at L2 network termination, simplifying the management and maintenance of multilayer networks. Without impacting the underlying L2 infrastructure, this feature uses CFM packets to verify the connection of L3 paths. Previously, CFM Down MEP support was limited to L2 interfaces associated with cross-connect or bundle members. This feature is supported on both physical main and subinterfaces, bundle main and subinterfaces.
CFM on bundle member link for connectivity check	Release 7.3.15	This feature introduces support for Connectivity Fault Management (CFM) on bundle members. Earlier, network administrators managed networks by using the fault, configuration, account, performance, security model. CFM is one of a suite of the Ethernet OAM protocols, which uses a combination of keepalive packets and MAC-based pings, and traceroutes to detect faults in a network. With the CFM feature, you: - reduce operating expenses for service operators by reducing network faults and errors - provide end-to-end maintenance of networks

Feature name	Release	Description
Up MEP and down MEP support in CFM	Release 7.3.15	This feature introduces Maintenance End Points (MEP) entities that you can configure in a domain. MEPs send either CFM frames from the interface where they are configured or CFM frames that are received on other interfaces. MEPs allow you to perform fault management and carry out performance checks.

CFM is defined in these standards:

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

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

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

  Note

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

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

Configuration guidelines and restrictions for Ethernet CFM

This topic describes the configuration guidelines and restrictions for Ethernet CFM, including the supported CCM timer values for regular and bundle-member sessions, the unsupported CFM configurations such as MIPs and L2 subinterfaces with default encapsulation, and the L2VPN requirement for CFM down MEP configuration.

Supported CCM interval timers and configurations

Configure the CCM interval timer within the supported values for the session type.

- Supported timers for CFM sessions: 1s, 10s, 1m, 10m.
- Supported timers for CFM sessions on bundle members: 100ms, 1s, 10s, 1m, 10m.

From Release 24.4.1, Cisco 8000 routers support 500 CFM sessions.

Unsupported CFM configurations

The following CFM configurations are not supported:

- The system supports only cross-connect.
- MIPs are not supported.
- L3 Interfaces and sub-Interfaces

- Bridge Domain, Release 7.3.1 and earlier
- VPLS, Release 7.3.1 and earlier
- L3 interfaces are not supported except for bundle members.
- Down MEPs are only supported for L2 cross-connect and bundle members.
- Multiple MEPs of different directions are not supported on the same interface or Xconnect.
- CFM is not supported on L2 subinterfaces with default encapsulation.

Include the interface in an L2VPN when configuring CFM down MEP

When you configure a CFM down MEP on an interface, ensure that the interface is included in an L2VPN.

This principle applies when you configure a CFM down MEP on an interface.

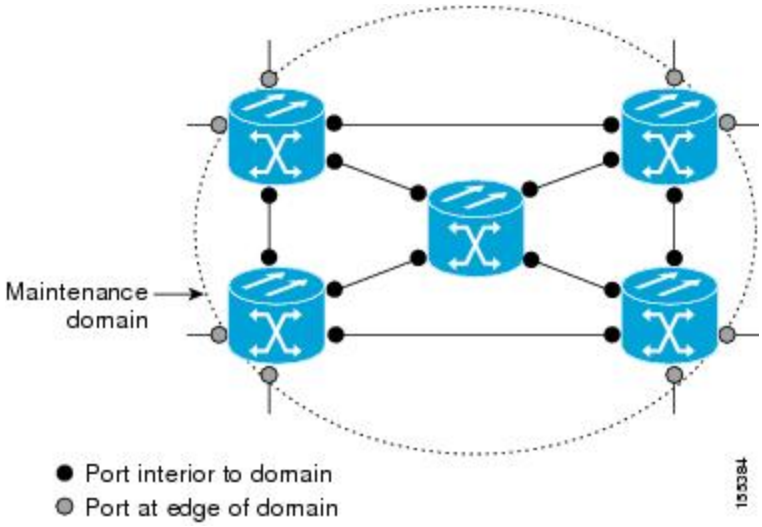
Maintenance domains

This topic describes maintenance domains, their hierarchical levels, and their role in managing and administering CFM network connectivity.

The maintenance domain is a management space that

- manages and administers a network
- operates under a single entity, and
- defines boundaries through specific bridge ports.

Figure 3: CFM Maintenance Domain



Administrators assign maintenance levels between 0 and 7 to define hierarchical relationships. Organizations use CFM maintenance domains in the following ways:

- The customer uses CFM between CE devices to verify and manage connectivity across the entire network.
- The service provider uses CFM between PE devices to verify and manage the services they provide.
- Each operator uses CFM within their operator network to verify and manage connectivity within that network.

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

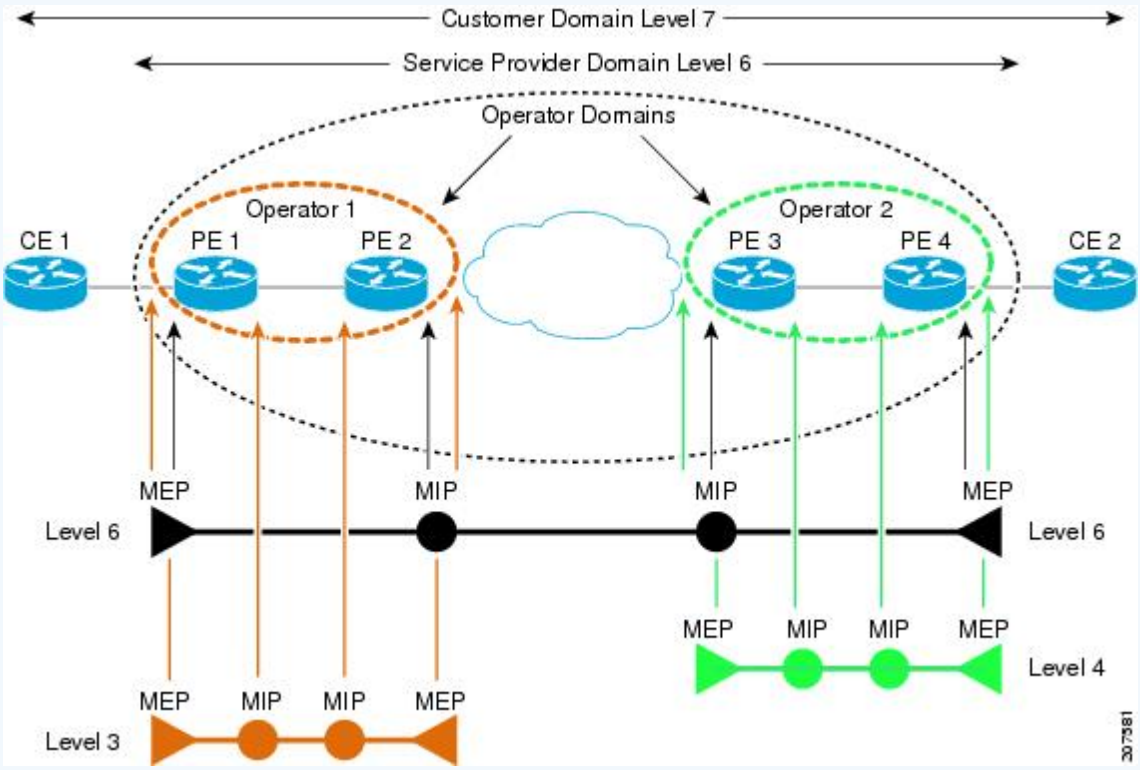
- The customer can use CFM between their CE devices, to verify and manage connectivity across the whole network.
- The service provider can use CFM between their PE devices, to verify and manage the services they are providing.
- Each operator can use CFM within their operator network, to verify and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

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

Note

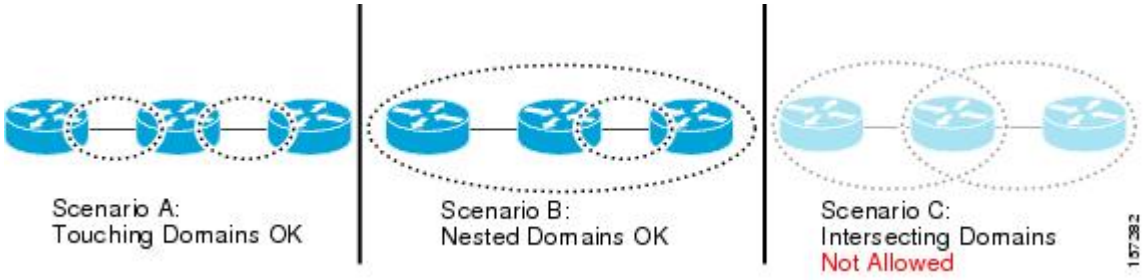
In CFM diagrams, the conventions are that triangles represent MEPs, pointing in the direction that the MEP sends CFM frames, and circles represent MIPs.

Figure 4: Different CFM Maintenance Domains Across a Network



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

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



Configure a CFM maintenance domain

This topic describes how to create and name a CFM maintenance domain container at a specified maintenance level.

Use this procedure to configure a CFM maintenance domain.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter Ethernet Connectivity Fault Management (CFM) configuration mode.

   Example

   ```
   Router(config)# ethernet cfm
   ```

3. Use the **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] command to create, name a container for all domain configurations, and enter CFM domain configuration mode.

   Example

   ```
   Router(config-cfm)# domain Domain_One level 1 id string D1
   ```

   The level must be specified.

   The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.

4. Optional: Use the **traceroute cache hold-time** *minutes* **size** *entries* command to set the maximum limit of traceroute cache entries or the maximum time limit to hold the traceroute cache entries.

   Example

   ```
   Router(config-cfm)# traceroute cache hold-time 1 size 3000
   ```

   The default is 100 minutes and 100 entries.

5. Save the configuration changes to the running configuration file.

Example

```
Router(config-cfm-dmn)# commit
```

6. End the configuration session and exits to the EXEC mode.

Example

```
Router(config-cfm-dmn)# end
```

Configure services for a CFM maintenance domain

This topic describes how to configure a service under a Connectivity Fault Management (CFM) maintenance domain.

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

Procedure

---

1. Enter global configuration mode.

Example

```
Router# configure
```

2. Enter Ethernet Connectivity Fault Management (CFM) configuration mode.

Example

```
Router(config)# ethernet cfm
```

3. Use the **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] command to create, name a container for all domain configurations, and enter CFM domain configuration mode.

Example

```
Router(config-cfm)# domain Domain_One level 1 id string D1
```

The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.

4. Use the **down-meps** | **xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id** command to configure and associate a service with the domain and enters CFM domain service configuration mode. You can specify that the service is used only for down MEPs.

Example

```
Router(config-cfm-dmn)# service xconnect group X1
```

The **id** sets the short MA name.

5. Save the configuration changes to the running configuration file.

Example

```
Router(config-cfm-dmn-svc)# commit
```

6. End the configuration session and exits to the EXEC mode.

Example

```
Router(config-cfm-dmn-svc)# end
```

Services

This topic defines CFM services, detailing how they partition maintenance domains to effectively manage and verify network connectivity.

The CFM service is a partition of a maintenance domain that

- organizes connectivity within the network
- operates independently within specific network segments, and
- aligns with the network topology to prevent frame leakage.

A CFM service allows an organization to partition its CFM maintenance domain, according to the connectivity within the network.

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

Each CFM service operates within an associated maintenance domain and adopts that domain's maintenance level. All CFM frames relating to the service carry this maintenance level.

Organizations implement CFM services to manage network connectivity in these ways:

- Organizations partition the maintenance domain according to network connectivity requirements.
- Administrators create a unique service for each virtual LAN to ensure independent operation.
- Service providers assign separate services to customers to verify and manage specific end-point connectivity.

> **Note**
>
> IEEE 802.1ag refers to CFM services as *Maintenance Associations*, while ITU-T Y.1731 refers to CFM services as *Maintenance Entity Groups*.

Maintenance points

This topic covers CFM maintenance points, their role in frame processing, and the function of Maintenance End Points.

A CFM Maintenance Point (MP) is an instance of a particular CFM service on a specific interface that

- resides on a specific interface

- processes CFM frames at the same maintenance level, and
- enforces maintenance domain boundaries.
- Maintenance End Points (MEPs)—Created at the edge of the domain. Maintenance end points (MEPs) are members of a particular service within a domain and are responsible for sourcing and sinking CFM frames. They periodically transmit continuity check messages and receive similar messages from other MEPs within their domain. They also transmit traceroute and loopback messages at the request of the administrator. MEPs are responsible for confining CFM messages within the domain.

A maintenance point is always associated with a particular CFM service, and therefore with a particular maintenance domain at a particular level. CFM maintenance points operate only on interfaces where they are configured; otherwise, the interface forwards CFM frames transparently. Maintenance points process frames at their associated maintenance level, forward frames at higher levels transparently, and drop frames at lower levels.

CFM maintenance points include the following types:

- Maintenance End Points (MEPs) function at the edge of the domain.
- MEPs belong to a specific service within a domain and source or sink CFM frames.
- MEPs periodically transmit and receive continuity check messages to monitor connectivity.
- MEPs transmit traceroute and loopback messages upon administrator request to confine CFM messages within the domain.

Maintenance End Points and Connectivity Fault Management processes

This topic describes how Maintenance End Points (MEPs) are sub-divided into Down MEPs and Up MEPs.

The Maintenance Endpoints (MEPs) are network elements that define the boundary of a domain at an interface rather than at a bridge or host. They are categorized into these types:

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

Characteristics and operational rules of MEPs

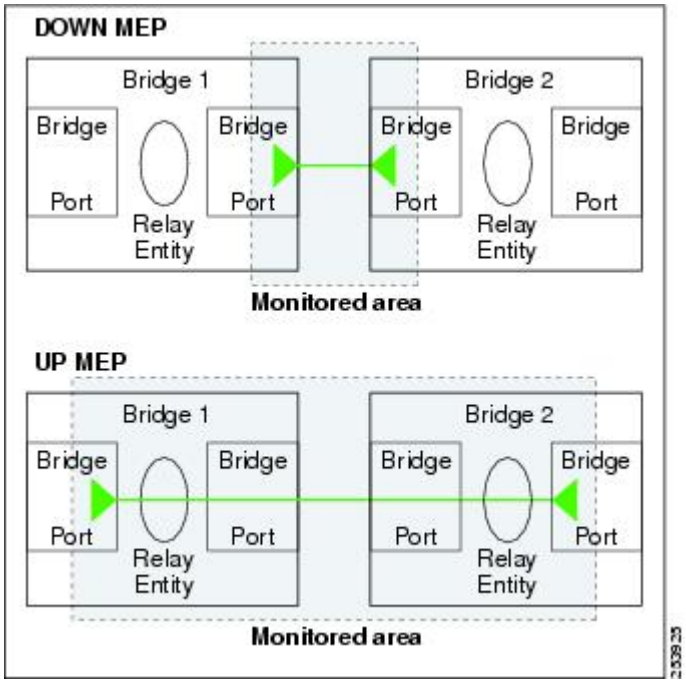
These instructions describe the characteristics and operational rules of MEPs:

- The terms *Down MEP* and *Up MEP* are defined in the IEEE 802.1ag and ITU-T Y.1731 standards, and refer to the direction that CFM frames are sent from the MEP. The terms should not be confused with the operational status of the MEP.
- The router supports only the configuration where the Down MEP level is less than the Up MEP level.
- Up MEPs can only exist on switched (Layer 2) interfaces, because they send and receive frames from the bridge relay function. Down MEPs can be created on switched (Layer 2) interfaces.
- MEPs continue to operate normally if the interface they are created on is blocked by the Spanning Tree Protocol (STP); that is, CFM frames at the level of the MEP continue to be sent and received, according to the direction of the MEP. MEPs never allow CFM frames at the level of the MEP to be forwarded, so the STP block is maintained.

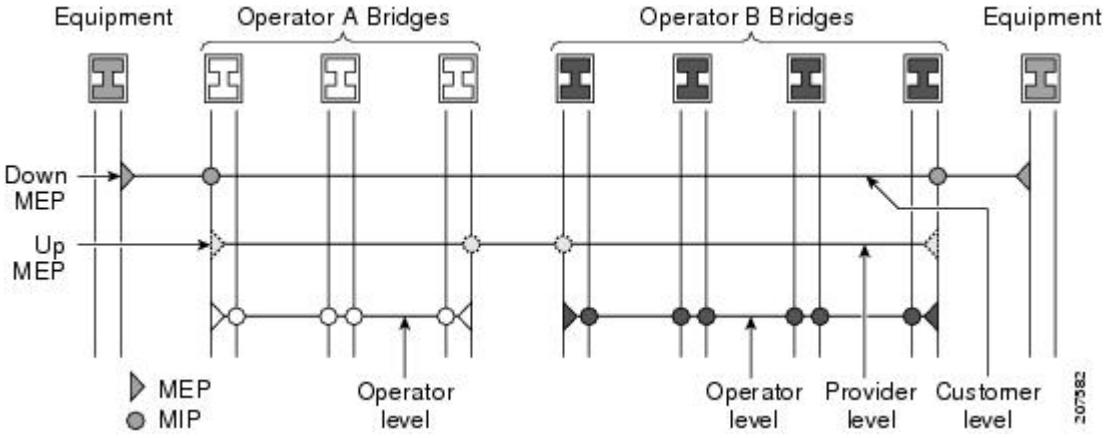
- A separate set of CFM maintenance levels is created every time a VLAN tag is pushed onto the frame. Therefore, if CFM frames are received on an interface which pushes an additional tag, so as to “tunnel” the frames over part of the network, the CFM frames will not be processed by any MPs within the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

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

Figure 5: Monitored Areas for Down and Up MEPs



This figure shows maintenance points at different levels. Because domains are allowed to nest but not intersect, a MEP at a low level often corresponds with a MEP at a higher level.



Configure cross-check on a MEP for a CFM service

This topic describes how to enable cross-check on a Maintenance End Point (MEP) for a Connectivity Fault Management (CFM) service, and specify the expected set of peer MEPs by adding each MEP-ID individually using the appropriate mep-id command form for non-offloaded, software-offloaded, or hardware-offloaded MEPs.

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

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter Ethernet Connectivity Fault Management (CFM) configuration mode.

   Example

   ```
   Router(config)# ethernet cfm
   ```

3. Use the **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] command to create, name a container for all domain configurations, and enter CFM domain configuration mode.

   Example

   ```
   Router(config-cfm)# domain Domain_One level 1 id string D1
   ```

   The level must be specified.

   The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.

4. Use the **service** *service-name* {**down-meps** | **xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id**[**icc-based** *icc-string umc-string*] | [**string** *text*] | [**number** *number*] | [**vlan-id** *id-number*] | [**vpn-id** *oui-vpnid*]] command to configure and associate a service with the domain and enters CFM domain service configuration mode. You can specify that the service is used only for down MEPs, or associate the service with a xconnect where up MEPs will be created.

   The **id** sets the short MA name.

5. Enter CFM MEP crosscheck configuration mode.

   Example

   ```
   Router# mep crosscheck mep-id 10
   ```

6. Use the **mep-id** *mep-id-number* command to enable cross-check on a MEP.

   Example

   ```
   Router(config-cfm-xcheck)# mep-id 10
   ```

> **Note**
>
> - For non-offloaded and software-offloaded MEPs, use the **mep-id** *mep-id-number* [**mac-address** *mac-address*] command.
> - For hardware-offloaded MEPs, use the **mep-id** *mep-id-number* command. From Release 24.2.11, **mac-address** *mac-address* option is obsolete for hardware-offloaded MEPs.
> - Repeat this command for every MEP that you want included in the expected set of MEPs for cross-check.

7. **mep-id** *mep-id-number mep-id-number* [**mac-address** *mac-address*]

   Example

   ```
   Router(config-cfm-xcheck)# mep-id 10
   ```

   Enables cross-check on a MEP.

   > **Note**
   >
   > - Repeat this command for every MEP that you want included in the expected set of MEPs for cross-check.

8. Save the configuration changes to the running configuration file and remain within the configuration session.

   Example

   ```
   Router(config-cfm-xcheck)# commit
   ```

9. End the configuration session and exits to the EXEC mode.

   Example

   ```
   Router(config-cfm-xcheck)# end
   ```

Configure other options for a CFM service

This topic describes how to configure the maximum number of MEPs across the network that limits the number of peer MEPs recorded in the database, and enable logging of AIS, continuity check errors, continuity check MEP changes, crosscheck errors, or EFD events for a CFM service.

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

Procedure

---

1. Enter global configuration mode.

Example

```
Router# configure
```

2. Enter Ethernet Connectivity Fault Management (CFM) configuration mode.

   Example

   ```
   Router(config)# ethernet cfm
   ```

3. Use the **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] command to create, name a container for all domain configurations, and enter CFM domain configuration mode.

   Example

   ```
   Router(config-cfm)# domain Domain_One level 1 id string D1
   ```

   The level must be specified.

   The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.

4. Use the **service** *service-name* {**down-meps** | **xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id**[**icc-based** *icc-string umc-string*] | [**string** *text*] | [**number** *number*] | [**vlan-id** *id-number*] | [**vpn-id** *oui-vpnid*]] command to configure and associate a service with the domain and enters CFM domain service configuration mode. You can specify that the service is used only for down MEPs, or associate the service with a xconnect where up MEPs will be created.

   The **id** sets the short MA name.

5. Optional: Use the **maximum-meps** *number* command to configure the maximum number (2 to 8190) of MEPs across the network, which limits the number of peer MEPs recorded in the database.

   Example

   ```
   Router(config-cfm-dmn-svc)# maximum-meps 1000
   ```

6. Use the **log** {**ais**|**continuity-check errors**|**continuity-check mep changes**|**crosscheck errors**|**efd**} command to enables logging of certain types of events.

   Example

   ```
   Router(config-cfm-dmn-svc)# log continuity-check errors
   ```

7. Save the configuration changes to the running configuration file and remain within the configuration session.

   Example

   ```
   Router(config-cfm-dmn-svc)# commit
   ```

8. End the configuration session and exits to the EXEC mode.

Example

```
Router(config-cfm-dmn-svc)# end
```

Configure CFM MEPs

This topic describes how to create a Maintenance End Point (MEP) on an Ethernet interface, Layer 2 interface, or Layer 2 subinterface, and optionally configure the class of service (CoS) for Connectivity Fault Management (CFM) packets generated by the MEP on that interface.

- For every subinterface configured under a Layer 3 parent interface, you must associate a unique 802.1Q or 802.1ad tag. Else, it leads to unknown network behavior.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Specify the type of Ethernet interface on which you want to create a MEP. Enter **HundredGigE** or **TenGigE** and the physical interface or virtual interface.

   Example

   ```
   Router(config)# interface TenGigE 0/0/0/1
   ```

   Note

   - Use the **show interfaces** command to see a list of all interfaces currently configured on the router.
   - L3 interfaces are only supported for bundle member interfaces. Else, you must enable l2transport.

3. Specify the type of Ethernet interface on which you want to create a MEP. Enter **interface** {**HundredGigE** | **TenGigE** | **Bundle-Ether**} *interface-path-id*[**l2transport**](#) and the physical interface or virtual interface followed by the l2transport. L2transport configures the interface as an L2 interface.

   Example

   ```
   Router(config)# interface TenGigE 0/0/0/1
   ```

   Naming convention is *interface-path-id*.*subinterface*. The period in front of the subinterface value is required as part of the notation.

4. Enter interface Ethernet CFM configuration mode.

Example

```
Router(config-if)# ethernet cfm
```

5. Use the **mep domain** *domain-name* **service** *service-name* **mep-id** *id-number* command to create a maintenance end point (MEP) on an interface, and enter interface CFM MEP configuration mode.

   Example

   ```
   Router(config-if-cfm)# mep domain Dm1 service Sv1 mep-id 1
   ```

6. Optional: Configure the class of service (CoS) (from 0 to 7) for all CFM packets generated by the MEP on an interface. If not configured, the CoS is inherited from the Ethernet interface.

   Example

   ```
   Router(config-if-cfm-mep)# cos 7
   ```

   Note

   For Ethernet interfaces, the CoS is carried as a field in the VLAN tag. Therefore, CoS only applies to interfaces where packets are sent with VLAN tags. If the **cos (CFM)** command is executed for a MEP on an interface that does not have a VLAN encapsulation configured, it will be ignored.

7. Save the configuration changes to the running configuration file and remain within the configuration session.

   Example

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

8. End the configuration session and exits to the EXEC mode.

   Example

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

CFM protocol messages

This topic describes the messages used by the Connectivity Fault Management (CFM) protocol.

The CFM messages are protocol messages that

- use the CFM EtherType and carry the CFM maintenance level for the domain to which they apply, and serve different purposes, and
- are defined by IEEE 802.1ag and ITU-T Y.1731 standards.

These topics describe the CFM messages that are used to verify connectivity, isolate faults, and trace paths:

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

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

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

This topic describes how Continuity Check Messages (CCMs) are heartbeat messages exchanged periodically between all the MEPs in a service, allowing each MEP to discover its peer MEPs and verify connectivity, and identifies the CCM intervals, defect conditions, and MAID format requirements for offloaded MEPs.

Continuity Check Messages (CCMs) are CFM protocol messages, also known as multicast heartbeat messages that are exchanged periodically between all Maintenance End Points (MEPs) in a service, allow each MEP to discover its peer MEPs, and verify connectivity between them.

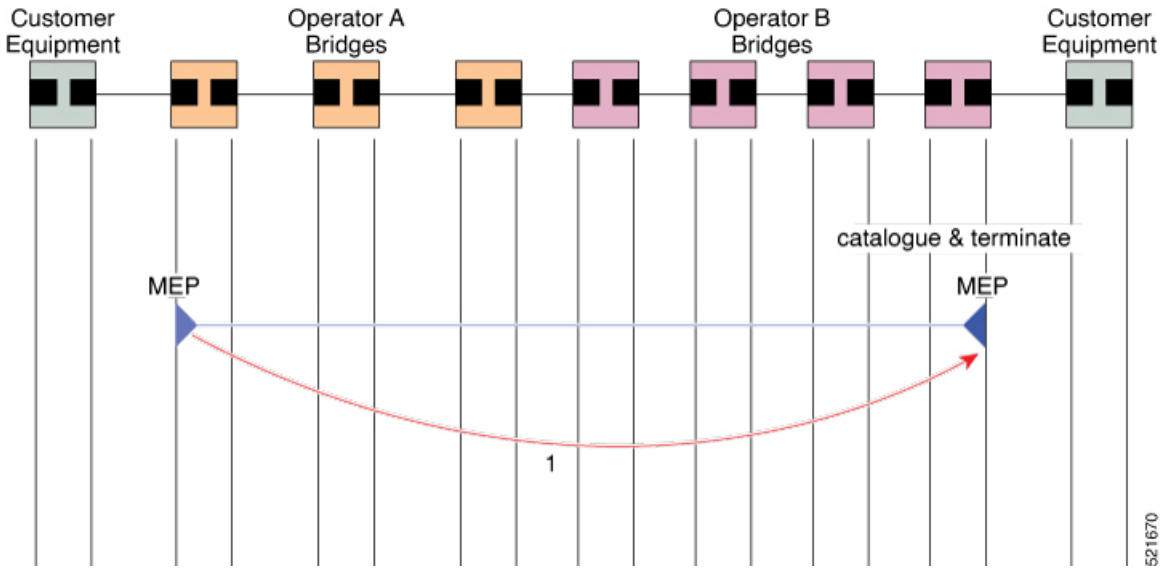
The CCMs carry a variety of information that enables detection of different defects in the service, and they are transmitted at configurable intervals, and cataloged by Maintenance Intermediate Points (MIPs) at the same maintenance level.

Table 16: Feature History Table

Feature Name	Release Information	Feature Description
Continuity Check (IEEE 802.1ag and ITU-T Y.1731)	Release 25.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) This feature is now supported on: • 8011-4G24Y4H-I • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Continuity Check (IEEE 802.1ag and ITU-T Y.1731)	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) Maintenance Endpoints (MEPs) capable of supporting Ethernet OAM send Continuity Check Messages (CCMs) at regular intervals to verify connectivity. If one of them stops receiving CCMs from another endpoint, it infers that there is a connectivity issue or fault in the path. This feature is now supported on: • 8011-4G24Y4H-I

MIPs also receive CCMs. MIPs use the information to build a MAC learning database that is used when responding to Linktrace. For more information about Linktrace, see Linktrace (IEEE 802.1ag and ITU-T Y.1731) on page 144.

Figure 6: Continuity Check Message Flow



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

- A configured numeric identifier for the MEP (the MEP ID). Each MEP in the service must be configured with a different MEP ID.
- In a Remote Defect Indication (RDI), each MEP includes this in the CCMs it is sending, if it has detected a defect relating to the CCMs it is receiving. This notifies all the MEPs in the service that a defect has been detected somewhere in the service.
- The interval at which CCMs are being transmitted.
- The status of the interface where the MEP is operating, for example, whether the interface is up, down, STP blocked, and so on.

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

Characteristics and operational rules of CCMs

These instructions describe the characteristics and operational rules of CCMs:

- All the MEPs in a service must transmit CCMs at the same interval. IEEE 802.1ag defines the following possible intervals that can be used:
  - 100 ms (only supported on bundle members)
  - 1 s
  - 10 s
  - 1 minute
  - 10 minutes

- A MEP detects a loss of connectivity with one of its peer MEPs when some number of CCMs are missed. This occurs when sufficient time has passed during which a certain number of CCMs were expected, given the CCM interval. This number is called the *loss threshold*, and is usually set to 3.
- With the exception of bundle members, CFM is supported only on interfaces that have Layer 2 transport feature enabled.
- CCMs include identifiers such as the Maintenance Domain Identifier (MDID) and the Short MA Name (SMAN), which together form the Maintenance Association Identifier (MAID) that must be configured identically on every MEP in the service.

MAID formats

These are restrictions on the type of MAID that are supported for sessions with time interval of less than 1 minute. The MAID supports two types of formats on offloaded MEPs:

- No Domain Name Format
  - MD Name Format = 1-NoDomainName
  - Short MA Name Format = 3 - 2 bytes integer value
  - Short MA NAme Length = 2 - fixed length
  - Short MA Name = 2 bytes of integer
- 1731 Maid Format
  - MD Name Format = 1-NoDomainName
  - MA Name Format(MEGID Format) = 32
  - MEGID Length = 13 - fixed length
  - MEGID(ICCCode) = 6 Bytes
  - MEGID(UMC) = 7 Bytes
  - ITU Carrier Code (ICC) - Number of different configurable ICC code - 15 (for each NPU)
  - Unique MEG ID Code (UMC) - 4
- Maintenance Association Identifier (MAID) comprises of the Maintenance Domain Identifier (MDID) and Short MA Name (SMAN). MDID only supports **null** value and SMAN only supports ITU Carrier Code (ICC) or a numerical. No other values are supported.
  - An example for configuring domain ID null is: `ethernet cfm domain SMB level 3 id null`
  - An example for configuring SMAN is: `ethernet cfm domain SMB level 3 id null service 901234AB xconnect group 99999 p2p 99999 id number 1`

CCM defects

These defects can be detected from the received CCMs:

- Interval mismatch: The CCM interval in the received CCM does not match the interval that the MEP is sending CCMs.
- Level mismatch: A MEP has received a CCM carrying a lower maintenance level than the MEPs own level.
- Loop: A CCM is received with the source MAC address equal to the MAC address of the interface where the MEP is operating.
- Configuration error: A CCM is received with the same MEP ID as the MEP ID configured for the receiving MEP.

- Cross-connect: A CCM is received with a MAID that does not match the locally configured MAID. This generally indicates a VLAN misconfiguration within the network, such that CCMs from one service are leaking into a different service.
- Peer interface down: A CCM is received that indicates the interface on the peer is down.
- Remote defect indication: A CCM is received carrying a remote defect indication.

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

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

Restrictions on MEPs

These restrictions apply to MEPs:

- Dynamic Remote MEPs are not supported for MEPs with less than 1 min interval. You must configure MEP CrossCheck for all such MEPs.
- Sequence numbering is not supported for MEPs with less than 1 minute interval.
- CCM Tx/Rx statistics counters are not supported for MEPs with less than1 minute intervals.
- Sender TLV and Cisco Proprietary TLVs are not supported for MEPs with less than 1 minute intervals.

Configure continuity check for a CFM service

This topic describes how to enable Continuity Check for a CFM service, specify the CCM transmission interval and loss threshold, configure how long peer MEP information is stored after time out, and optionally trigger an automatic traceroute when a MEP is declared down.

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

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter Ethernet Connectivity Fault Management (CFM) configuration mode.

   Example

   ```
   Router(config)# ethernet cfm
   ```

3. Use the **domain** *domain-name* **level** *level-value* [**id** [**null**] [**dns** *DNS-name*] [**mac** *H.H.H*] [**string** *string*]] command to create, name a container for all domain configurations, and enter CFM domain configuration mode.

Example

```
Router(config-cfm)# domain Domain_One level 1 id string D1
```

The level must be specified.

The **id** is the maintenance domain identifier (MDID) and is used as the first part of the maintenance association identifier (MAID) in CFM frames. If the MDID is not specified, the domain name is used as the MDID by default.

4. Use the **down-meps | xconnect group** *xconnect-group-name* **p2p** *xconnect-name*}[**id** command to configure and associate a service with the domain and enters CFM domain service configuration mode. You can specify that the service is used only for down MEPs.

   Example

   ```
   Router(config-cfm-dmn)# service xconnect group X1
   ```

   The **id** sets the short MA name.

5. Optional: Use the **continuity-check interval** *time* [**loss-threshold** *threshold*] command to enable Continuity Check and specify the time interval at which CCMs are transmitted or to set the threshold limit for when a MEP is declared down.

   Example

   ```
   Router(config-cfm-dmn-svc)# continuity-check interval 100m loss-threshold 10
   ```

6. Optional: Use the **continuity-check archive hold-time** *minutes* command to configure how long information about peer MEPs is stored after they have timed out.

   Example

   ```
   Router(config-cfm-dmn-svc)# continuity-check archive hold-time 100
   ```

7. Optional: Configure automatic triggering of a traceroute when a MEP is declared down.

   Example

   ```
   Router(config-cfm-dmn-svc)# continuity-check loss auto-traceroute
   ```

8. Save the configuration changes to the running configuration file and remain within the configuration session.

   Example

   ```
   Router(config-cfm-dmn-svc)# commit
   ```

9. End the configuration session and exits to the EXEC mode.

   Example

   ```
   Router(config-cfm-dmn-svc)# end
   ```

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

This topic describes how Loopback Messages (LBM) and Loopback Replies (LBR) verify connectivity between a local MEP and a particular remote MP by sending unicast LBMs and receiving LBRs from the target maintenance point.

Loopback Messages (LBM) and Loopback Replies (LBR) are CFM protocol messages that

- verify connectivity between a local MEP and a particular remote MP, and
- sends unicast LBMs to the remote MP when requested by an administrator.

Table 17: Feature History Table

Feature Name	Release Information	Feature Description
Loopback (IEEE 802.1ag and ITU-T Y.1731)	Release 25.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) This feature is now supported on • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Loopback (IEEE 802.1ag and ITU-T Y.1731)	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) You can test connectivity between a local Maintenance End Point (MEP) and a remote Maintenance Point (MP) by sending Loopback Messages (LBM) and Loopback Replies (LBR). The connectivity verification facilitates effective network troubleshooting and maintenance without requiring a detailed hop-by-hop path analysis. This feature is now supported on Cisco 8011-4G24Y4H-I routers.

How CFM loopback works

This topic describes how CFM loopback works by using Loopback Messages (LBM) and Loopback Replies (LBR) to verify connectivity between a local MEP and a remote maintenance point, forwarding unicast messages through the bridge domain until the target replies.

Loopback messages function similarly to an ICMP Echo (ping) and indicate destination reachability but do not allow hop-by-hop path discovery. This process enables administrators to verify connectivity between a local MEP and a remote maintenance point efficiently without detailed path analysis.

Loopback messages can be padded with user-specified data to detect data corruption and carry a sequence number to detect out-of-order frames.

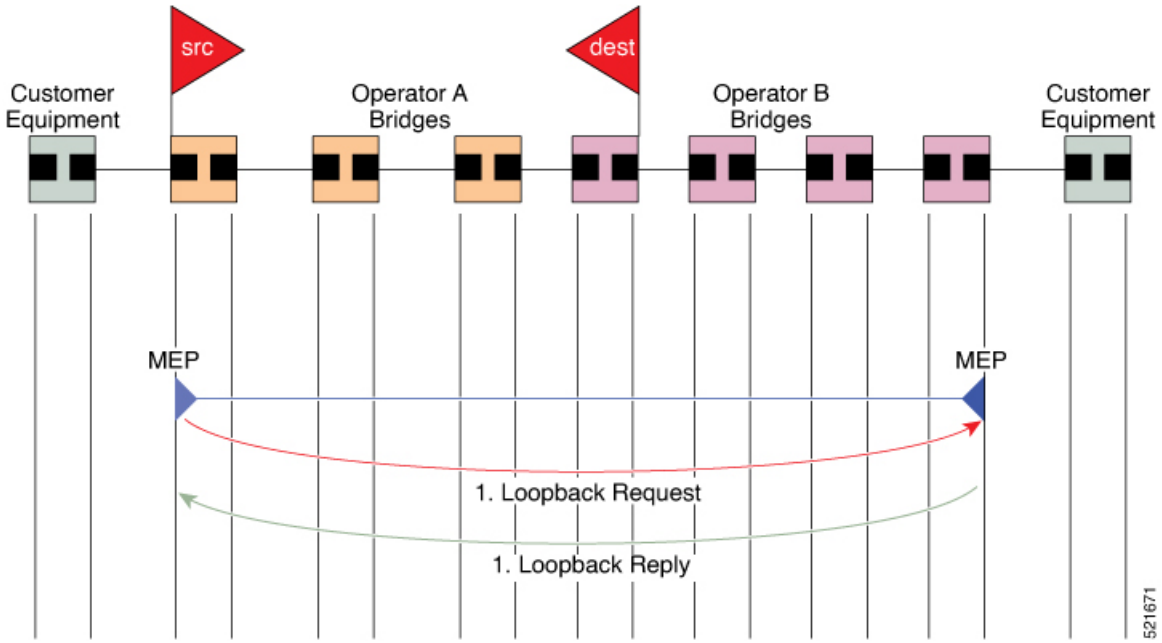
Summary

The following actors and components participate in this process:

- Originating MEP: Sends a unicast LBM to the target maintenance point.
- Target maintenance point: Receives the LBM and sends an LBR back to the originating MEP.
- Intermediate bridges: Forward the LBM like normal data traffic while respecting maintenance levels.
- Bridge forwarding database: Determines the outgoing interface for the loopback message at each device.

Workflow

Figure 7: Loopback Messages



These stages describe how a CFM loopback message flows between a local MEP and the target maintenance point and enables administrators to verify connectivity between a local MEP and a remote maintenance point efficiently without detailed path analysis.

1. On receiving each LBM, the target maintenance point sends an LBR back to the originating MEP.
2. Loopback messages function similarly to an ICMP Echo (ping) and indicate destination reachability but do not allow hop-by-hop path discovery.
3. Since loopback messages are destined for unicast addresses, they are forwarded like normal data traffic while respecting maintenance levels.
4. At each device the loopback message reaches, if the outgoing interface is known in the bridge's forwarding database, the frame is sent out on that interface.
5. If the outgoing interface is not known, the loopback message is flooded on all interfaces within the domain.

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

This topic describes how Linktrace Messages (LTM) and Linktrace Replies (LTR) track the path hop-by-hop to a unicast destination MAC address.

Linktrace Messages (LTM) and Linktrace Replies (LTR) are CFM protocol messages that are used to track the path (hop-by-hop) to a unicast destination MAC address.

The linktrace mechanism is designed to provide useful information even after a network failure. This allows it to be used to locate failures, for example after a loss of continuity is detected. To achieve this, each MP maintains a CCM Learning Database. This maps the source MAC address for each received CCM to the interface through which the CCM was

received. It is similar to a typical bridge MAC learning database, except that it is based only on CCMs and it times out much more slowly—on the order of days rather than minutes.

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

Table 18: Feature History Table

Feature Name	Release Information	Feature Description
Linktrace (IEEE 802.1ag and ITU-T Y.1731)	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Linktrace (IEEE 802.1ag and ITU-T Y.1731)	Release 25.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) This feature is now supported on: • 8011-4G24Y4H-I • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Linktrace (IEEE 802.1ag and ITU-T Y.1731)	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) You can trace the path to a specific unicast destination MAC address on a network using the Linktrace Messages (LTM) and Linktrace Replies (LTR) diagnostic tools. These tools perform a detailed hop-by-hop path analysis to identify connectivity problems. This feature is now supported on: • 8011-4G24Y4H-I

How linktrace works

This topic describes how CFM linktrace works.

CFM linktrace is similar in concept to IP traceroute, although the mechanism is different. In IP traceroute, successive probes are sent, whereas CFM linktrace uses a single LTM that is forwarded by each MP in the path. LTMs are multicast and carry the unicast target MAC address as data within the frame. They are intercepted at each hop where there is a maintenance point, and either retransmitted or dropped to discover the unicast path to the target MAC address.

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

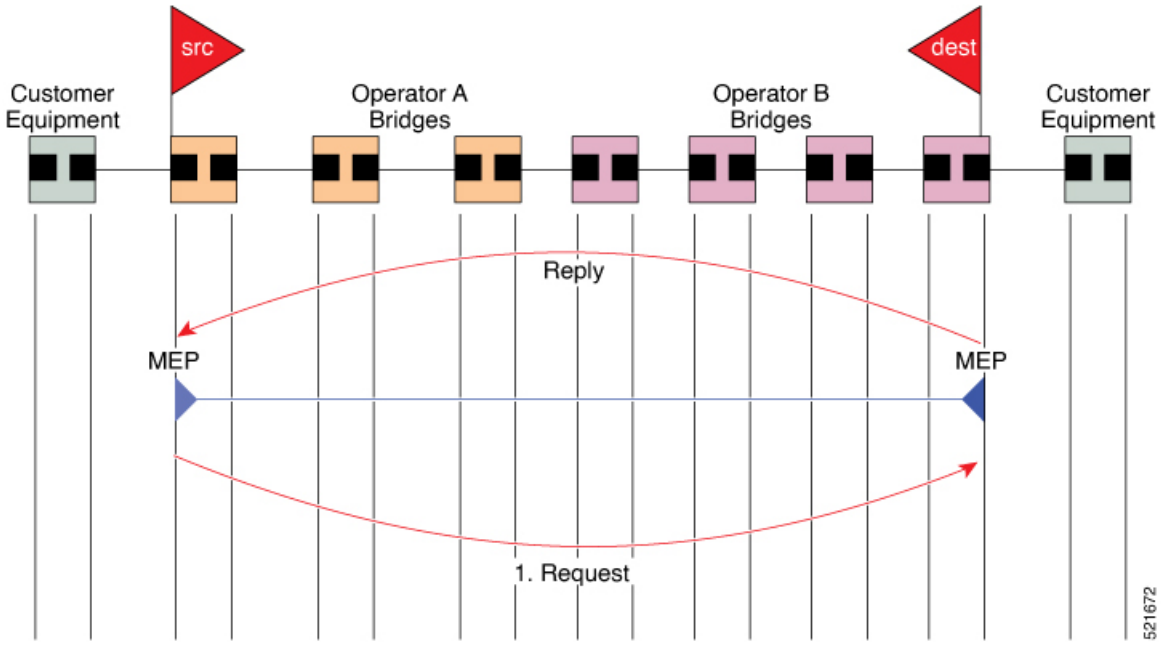
Summary

The following actors and components participate in this process:

- Operator: Requests a linktrace operation from the local MEP.
- Originating MEP: Sends a multicast LTM that carries the unicast target MAC address, and collects the LTRs returned by maintenance points along the path.
- Intermediate maintenance points (MPs): Intercept the LTM, consult their forwarding tables to decide whether to reply, and either send an LTR back to the originating MEP or drop the LTM.
- Bridge MAC learning table: The first table an MP consults to look up the target MAC address.
- CCM learning database: The second table an MP consults when the target MAC address is not in the bridge MAC learning table.

Workflow

Figure 8: Linktrace Message Flow



These stages describe how a CFM linktrace operation discovers the path to a target MAC address:

1. Stage 1 – The operator triggers the linktrace. At the request of the operator, the local MEP sends an LTM. The LTM is multicast and carries the unicast target MAC address as data within the frame.
2. Stage 2 – Each MP intercepts the LTM and decides whether to reply. In IEEE 802.1ag, when an MP receives an LTM message, it determines whether to send a reply using these steps:

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

3. Stage 3 – The originating MEP collects the replies. Each MP that finds the target MAC address sends an LTR back to the originating MEP, which enables the administrator to discover connectivity data about the path. If the target MAC has never been seen previously in the network, the linktrace operation does not produce any results.

Configurable logging

This topic describes how Connectivity Fault Management (CFM) supports logging of various conditions to syslog.

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

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

CFM hardware offload

This topic describes the hardware offload feature for connectivity fault management (CFM), which categorizes offload based on where continuity check messages (CCMs) are processed.

Depending on where the continuity check messages (CCMs) are processed, offload is categorized into these types:

- Software offload–When CCMs are processed by the line card CPU, offload type is known as software offload. Software offload is supported only on bundle interface.
- Hardware offload–When CCMs are processed by network processing unit (NPU), offload type is known as hardware offload.
- Non-offload–When CCMs are processed by route processor (RP), offload type is known as non-offload.

Table 19: Feature History Table

Feature Name	Release Information	Feature Description
CFM hardware offload	Release 26.2.1	Introduced in this release on: Modular Systems (8400 [ASIC: K100])(select variants only*) This feature enables faster fault detection in Connectivity Fault Management (CFM) sessions. It introduces hardware offload capabilities for 1 ms CCM interval, where the NPU now directly processes these packets. *This feature is supported on Cisco 8404-SYS-D.

Feature Name	Release Information	Feature Description
CFM hardware offload	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100])(select variants only*) *This feature is supported on: • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
CFM hardware offload	Release 25.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) This feature enables faster fault detection in Connectivity Fault Management (CFM) sessions. It introduces hardware offload capabilities for 3.3 ms, 10 ms, and 100 ms CCM intervals, where the NPU now directly processes these packets. *This feature is supported on 8011-4G24Y4H-I.

CCM intervals are the intervals in which CCMs are sent and received. If the CCMs are not received within the configured interval, the CFM MEP goes down. The following table shows the supported CCM timers for the offload types:

Interface Type	Offload Type	Supported CCM Timers
Physical interfaces and subinterfaces	Non-offload	• 1 sec • 10 sec • 1 min • 10 min
	Hardware offload	• 1 sec • 3.3 ms • 10 ms • 100 ms
Bundle members	Non-offload	• 1 sec • 10 sec • 1 min • 10 min
	Hardware offload	• 1 sec • 3.3 ms • 10 ms • 100 ms

Interface Type	Offload Type	Supported CCM Timers
Bundle interfaces and subinterfaces	Non-offload	• 1 sec • 10 sec • 1 min • 10 min
	Hardware offload	• 1 sec • 3.3 ms • 10 ms • 100 ms

Configure AIS in a CFM domain service

This topic describes how to configure Alarm Indication Signal (AIS) transmission for a Connectivity Fault Management (CFM) domain service, and configure AIS logging for the CFM domain service to indicate when AIS or LCK packets are received.

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

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter Ethernet CFM global configuration mode.

   Example

   ```
   Router(config)# ethernet cfm
   ```

3. Specifiy the domain and domain level.

   Example

   ```
   router(config-cfm)# domain D1 level 1
   ```

4. Use the **service** *name* **xconnect group** *xconnect-group-name* **p2p** *xconnect-name* command to specify the service and cross-connect group and name.

Example

```
Router(config-cfm-dmn)# service S1 xconnect group XG1 p2p X2
```

5. Use the **ais transmission** [**interval** {**1s**|**1m**}][**cos** *cos*] command to configure Alarm Indication Signal (AIS) transmission for a Connectivity Fault Management (CFM) domain service.

   Example

   ```
   Router(config-cfm-dmn-svc)# ais transmission interval 1m cos 7
   ```

6. Configure AIS logging for a Connectivity Fault Management (CFM) domain service to indicate when AIS or LCK packets are received.

   Example

   ```
   Router(config-cfm-dmn-svc)# log ais
   ```

7. Save the configuration changes to the running configuration file and remain within the configuration session.

   Example

   ```
   Router(config-sla-prof-stat-cfg)# commit
   ```

8. End the configuration session and exit to the EXEC mode.

   Example

   ```
   Router(config-sla-prof-stat-cfg)# end
   ```

Configure AIS on a CFM interface

This topic describes how to configure Alarm Indication Signal (AIS) transmission on a Connectivity Fault Management (CFM) interface by entering interface configuration mode, enabling Ethernet CFM, and specifying the AIS interval and class of service.

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

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter interface configuration mode.

Example

```
Router# interface TenGigE 0/0/0/2
```

3. Enter Ethernet CFM interface configuration mode.

   Example

   ```
   Router(config)# ethernet cfm
   ```

4. Use the **ais transmission up interval 1m cos** *cos* command to configure Alarm Indication Signal (AIS) transmission on a Connectivity Fault Management (CFM) interface.

   Example

   ```
   Router(config-if-cfm)# ais transmission up interval 1m cos 7
   ```

5. Save the configuration changes to the running configuration file and remain within the configuration session.

   Example

   ```
   Router(config-sla-prof-stat-cfg)# commit
   ```

6. End the configuration session and exits to the EXEC mode.

   Example

   ```
   Router(config-sla-prof-stat-cfg)# end
   ```

Verify the CFM configuration

This topic describes the show commands used to verify the CFM configuration, including commands to display configuration errors and a list of local maintenance points.

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

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

Note

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

CFM over bundles

This topic describes CFM support over bundle interfaces, bundle sub-interfaces, and bundle member interfaces, where the bundle member ports must span across line cards.

CFM over bundle supports the following:

- CFM Maintenance Points – UP MEP, Down MEP, which only includes L2 bundle main and sub-interfaces.
- CCM interval of 100 ms, 1 sec, 10 sec, 1 min, and 10 mins.

  CCM interval of 3.3 ms, 10 ms, 100 ms, 1 sec, 10 sec, 1 min, and 10 mins.
- RP OIR/VM reload without impacting learnt CFM peer MEPs.
- Process restart without impacting CFM sessions.
- Static MEPs.

Restrictions for configuration of CFM on bundles

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

- Only Layer 2 bundle Ethernet interfaces and sub-interfaces are supported, which are part of a L2VPN cross-connect.
- No support for 3.3 ms and 10 ms CCM interval.
- Supports 5000 pps rates of CCM traffic for bundle interfaces.
- Ethernet Connectivity Fault Management (CFM) is not supported with Maintenance association End Points (MEPs) that are configured on default and untagged encapsulated sub-interfaces that are part of a single physical interface.
- Multiple MEPs of different directions are not supported on the same interface or Xconnect.
- CFM does not support fast failover, which may result in session flaps on bundle interfaces. Use offload for virtual interfaces to avoid flaps on faster CCM intervals.

9 Configure Ethernet Performance Monitoring

Topics:

This chapter describes the Ethernet performance monitoring chapter, which covers Ethernet SLA statistics measurement in a profile, Ethernet frame delay measurement for L2VPN services, and minimum delay bin support, and includes tasks to configure SLA statistics, frame delay measurement, and minimum delay bin support.

Ethernet SLA statistics measurement in a profile

This topic describes how the Ethernet SLA feature supports measurement of one-way and two-way delay and jitter statistics and one-way FLR statistics.

The Ethernet SLA is a network performance measurement feature that

- supports measurement of one-way and two-way delay and jitter statistics
- calculates one-way frame loss ratio statistics, and
- evaluates network performance by sending packets and storing data metrics.

Table 20: Feature History Table

Feature Name	Release Information	Feature Description
Enhancement to Ethernet SLA Statistics Measurement	Release 7.7.1	You can now configure the size of bins for the delay and jitter measurement in Ethernet SLA statistics with a **width** value ranging from 1 to 10000000 microseconds. This enhancement provides granularity to store more accurate results of SLA statistics in the aggregate bins. In earlier releases, you could only configure the **width** value for the delay and jitter measurement in milliseconds. This feature introduces the **usec** keyword in the **aggregate** command.

The Ethernet SLA feature tracks various metrics to assess network performance:

- Round-trip delay time–The time for a packet to travel from source to destination and back to source again.
- Round-trip jitter–The variance in round-trip delay time (latency).
- One-way delay and jitter–The router also supports measurement of one-way delay or jitter from source to destination, or from destination to source.
- One-way frame loss–The router also supports measurement of one-way frame loss from source to destination, or from destination to source.
- Packet loss count records the overall number of measurement packets lost in either direction.
- Packet corruption events track data integrity issues.
- Out-of-order events monitor the sequence of received packets.
- Frame Loss Ratio (FLR) provides the ratio of lost packets to sent packets expressed as a percentage.

Performance metrics

The system maintains counters for packet loss, corruption, and out-of-order packets for each bucket and reports these metrics as a percentage of the total samples. It calculates the minimum, maximum, mean, and standard deviation for delay, jitter, and loss statistics within each bucket. The system also records individual samples or aggregated bins for these statistics. For synthetic loss measurement statistics, the system reports the overall Frame Loss Ratio (FLR) for the

bucket alongside individual measurements or aggregated bins. Finally, the system tracks the packet loss count as the total number of packets lost in both directions, while it measures one-way FLR separately for each direction.

Buckets

The bucket is a time interval that captures collected statistics, records results received during a defined period, and organizes measurements into bins and counters when aggregation is enabled.

The system records all results received during a specific time period in the corresponding bucket. When the system enables aggregation, it assigns a unique set of bins and counters to each bucket. These counters include only the results for measurements initiated during the time period that the bucket represents.

Frame Loss Ratios and availability

The Frame Loss Ratio (FLR) is a performance metric that calculates the ratio of lost packets to sent packets, expresses the result as a percentage value, and measures loss separately in each direction.

The system evaluates availability through the following methods:

- Availability measures network performance over long periods such as weeks or months.
- The system calculates the proportion of time during which the network experiences prolonged high loss.

Aggregations

The aggregation is a data management process that organizes samples into range-based bins, reduces memory consumption, and provides efficient statistical storage.

The system manages data storage through two primary methods:

- When the user enables aggregation using the **aggregate** command, the system creates bins to store counts of samples within specific value ranges defined by the **width** keyword.
- This process stores only a counter for each bin, which minimizes memory usage compared to storing individual results.
- When the user disables aggregation, the system stores each sample separately to provide more accurate statistical analysis at the cost of higher memory consumption.

One-way delay and jitter measurements

The one-way delay and jitter measurements are network performance configuration processes that - require profile configuration using the **type cfm-delay-measurement** command, mandate time synchronization between local and remote devices, and utilize specific frequency and time-of-day sources for accuracy.

The administrator follows these requirements to ensure accurate measurement results:

- The administrator configures the SLA profile using the **type cfm-delay-measurement** command.
- The system requires both local and remote devices to maintain time synchronization.
- The administrator selects frequency sources such as BITS, GPS, SyncE, or PTP.
- The administrator selects Time-of-Day (ToD) sources such as PTP or DTI, as NTP does not provide sufficient accuracy for these measurements.

Configuration guidelines and restrictions for Ethernet SLA

This topic describes the configuration guidelines and restrictions for Ethernet SLA statistics measurement.

Ethernet SLA memory usage considerations

Before you configure Ethernet SLA, consider how your SLA configuration affects router memory.

> **Caution**
>
> Certain SLA configurations can use a large amount of memory that can affect the performance of other features on the router.

- Aggregation–Use of the **aggregate none** command significantly increases the amount of memory required because each individual measurement is recorded, rather than just counts for each aggregation bin. When you configure aggregation, consider that more bins will require more memory.
- Buckets archive–When you configure the **buckets archive** command, consider that the more history that is kept, the more memory will be used.
- Measuring two statistics (such as both delay and jitter) will use approximately twice as much memory as measuring one.
- Separate statistics are stored for one-way source-to-destination and destination-to-source measurements, which consumes twice as much memory as storing a single set of round-trip statistics.

Define the schedule before probe parameters

You must define the schedule before you configure SLA probe parameters to send probes for a particular profile. It is recommended to set up the profile–probe, statistics, and schedule before any commit.

cfm-loopback profile restrictions

One-way delay and jitter measurements are not supported by cfm-loopback profile types.

Configure Ethernet SLA statistics measurement in a profile

This topic describes how to configure SLA statistics measurement in a profile by creating an SLA operation profile, enabling collection of SLA statistics, configuring aggregation bins, and configuring bucket size and archive.

To configure SLA statistics measurement in a profile, perform these steps:

Procedure

1. Enter the Ethernet SLA configuration mode, using the **ethernet sla** command in Global Configuration mode.

   Example

   ```
   Router(config)# ethernet sla
   ```

2. Create an SLA operation profile with the **profile** *profile-name* **type cfm-delay-measurement** command.

   Example

   ```
   Router(config-sla)# profile test type cfm-delay-measurement
   ```

3. Enable the collection of SLA statistics using the **statistics measure** {**one-way-delay-ds** | **one-way-delay-sd** | **one-way-jitter-ds** | **one-way-jitter-sd** | **round-trip-delay** | **round-trip-jitter** | **one-way-loss-ds** | **one-way-loss-sd**} command.

Example

```
Router(config-sla-prof)# statistics measure round-trip-delay
```

4. Configure the size and number of bins into which to aggregate the results of statistics collection. For delay measurements and data loss measurements, the default is that all values are aggregated into 1 bin. For synthetic loss measurements, by default the aggregation is disabled. Use the **aggregate** {**bins** *count* **width** [**usec**] *width* | **none**} command to configure the bins.

   Example

   ```
   Router(config-sla-prof-stat-cfg)# aggregate bins 5 width usec 123
   ```

   - For delay and jitter measurements, you can configure a width value from 1 to 10000 milliseconds, if the number of bins is at least 2. To configure the width value in microseconds, use the **usec** option. You can configure the width value from 1 to 10000000 microseconds.
   - For data loss and synthetic loss measurements, you can configure a width value from 1 to 100 percentage points, if the number of bins is at least 2.

5. Configure the size of the buckets in which statistics are collected, using the **buckets size** *number* **probes** command.

   Example

   ```
   Router(config-sla-prof-stat-cfg)# buckets size 1 probes
   ```

6. Configure the number of buckets to store in memory using the **buckets archive** *number* command.

   Example

   ```
   Router(config-sla-prof-stat-cfg)# buckets archive 50
   ```

7. Save the configuration changes using the **end** or **commit** command.

   Example

   ```
   Router(config-sla-prof-stat-cfg)# commit
   ```

8. Verify the configuration.

   Example

   ```
   /* This example displays aggregate bins configured with a range of 123
   microseconds */
   Router# show ethernet sla statistics detail
   Tue Sep 28 07:59:22.340 PDT
   Source: Interface GigabitEthernet0/0/0/2, Domain dom1
   Destination: Target MAC Address 0012.0034.0056
   ------------------------------------------------------------------------------
   Profile 'test', packet type 'cfm-delay-measurement'
   Scheduled to run every 1min first at 00:00:31 UTC for 10s
   Round Trip Delay
   ~~~~~~~~~~~~~~~~
   1 probes per bucket
   No stateful thresholds.
   ```

```
Bucket started at 07:56:31 PDT Tue 28 September 2021 lasting 10s
Pkts sent: 10; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
Misordered: 0 (0.0%); Duplicates: 0 (0.0%)
Result count: 10
Min: 0.000ms, occurred at 07:56:32 PDT Tue 28 September 2021
Max: 1.000ms, occurred at 07:56:31 PDT Tue 28 September 2021
Mean: 0.100ms; StdDev: 0.300ms
Bins:
Range Samples Cum. Count Mean
---------------- ---------- ----------- -------
0 to 0.123 ms 9 (90.0%) 9 (90.0%) 0.000ms
0.123 to 0.246 ms 0 (0.0%) 9 (90.0%) -
0.246 to 0.369 ms 0 (0.0%) 9 (90.0%) -
0.369 to 0.492 ms 0 (0.0%) 9 (90.0%) -
> 0.492 ms 1 (10.0%) 10 (100.0%) 1.000ms
```

```
/* This example displays aggregate bins configured with a range of 10
milliseconds */
Router# show ethernet sla statistics detail
Tue Sep 28 08:00:57.527 PDT
Source: Interface GigabitEthernet0/0/0/2, Domain dom1
Destination: Target MAC Address 0012.0034.0056
================================================================================
Profile 'test', packet type 'cfm-delay-measurement'
Scheduled to run every 1min first at 00:00:31 UTC for 10s
Round Trip Delay
~~~~~~~~~~~~~~~~
1 probes per bucket
No stateful thresholds.
Bucket started at 08:00:32 PDT Tue 28 September 2021 lasting 10s
Pkts sent: 9; Lost: 0 (0.0%); Corrupt: 0 (0.0%);
Misordered: 1 (11.1%); Duplicates: 0 (0.0%)
Result count: 9
Min: 0.000ms, occurred at 08:00:32 PDT Tue 28 September 2021
Max: 0.000ms, occurred at 08:00:32 PDT Tue 28 September 2021
Mean: 0.000ms; StdDev: 0.000ms
Results suspect due to a probe starting mid-way through a bucket
Bins:
Range Samples Cum. Count Mean
----------- ---------- ---------- -------
0 to 10 ms 9 (100.0%) 9 (100.0%) 0.000ms
10 to 20 ms 0 (0.0%) 9 (100.0%) -
20 to 30 ms 0 (0.0%) 9 (100.0%) -
30 to 40 ms 0 (0.0%) 9 (100.0%) -
> 40 ms 0 (0.0%) 9 (100.0%) -
```

Ethernet frame delay measurement for L2VPN services

This topic describes how Ethernet frame delay measurement complies with the ITU-T Y.1731 standard, using Delay Measurement Message (DMM) and Delay Measurement Reply (DMR) to periodically measure one-way or two-way frame delay.

The Ethernet frame delay measurement is a performance monitoring process that

- complies with the ITU-T Y.1731 standard, which provides comprehensive fault management and performance monitoring recommendations
- utilizes DMM and DMR protocol data units, and

- measures frame delay and delay variation between point-to-point Maintenance End Points (MEPs).

Table 21: Feature History Table

Feature Name	Release Information	Feature Description
Ethernet frame delay measurement for L2VPN services	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8711-48Z-M
Ethernet frame delay measurement for L2VPN services	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-36EH+A8:B12 • 88-LC1-52Y8H-EM
Ethernet frame delay measurement for L2VPN services	Release 7.5.3	You can now monitor L2VPN networks and avoid impact to your customers' operations by accurately measuring frame round-trip delays and jitters between two maintenance endpoints (MEPs). This feature lets you detect end-to-end connectivity, loopback, and link trace on MEPs. It reports service performance to your end customers, helping improve technical and operational tasks such as troubleshooting and billing. This feature introduces the **cfm-delay-measurement probe** command.

You can measure frame delay in the Layer 2 networks to detect end-to-end connectivity, loopback, and link trace on Maintenance End Points (MEPs) and also report service performance that helps to improve technical and operational tasks

such as troubleshooting, billing, and so on. Frame delay is the duration between the time the source node transmits the first bit of a frame and the time the same source node receives the last bit of the frame.

The frame delay measurement uses these protocol data units (PDUs):

- Delay Measurement Message (DMM)—DMM is used to measure frame delay and frame delay variation between a pair of point-to-point Maintenance End Points (MEPs).
- Delay Measurement Response (DMR)—DMR is the delay measurement response sent by the destination MEP. When an MEP receives a DMM frame, the responder MEP responds with a DMR frame. The DMR frame carries a reply information and a copy of the timestamp contained in the DMM frame.

Note

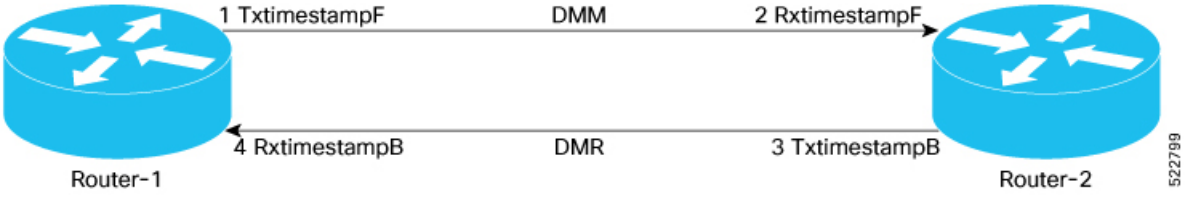
DMM sessions (using CFM) are not supported with MACsec enabled on the core interface, as this requires pre-encryption timestamping in the interface group.

We support one-way and two-way frame delay measurement.

Frame Delay Measurement	Description
One-way frame delay measurement (1DM)	• Measures the frame delay on a unidirectional link between the MEPs. • 1DM requires that clocks at both the transmitting MEP and the receiving MEPs are synchronized. • Measuring frame-delay variation does not require clock synchronization and the variation can be measured using 1DM and DMR frame combination.
Two-way frame delay measurement	• Measures the frame delay on a bidirectional link between the MEPs. • Two-way delay measurement does not require the clocks at both the transmitting MEP and the receiving MEPs to be synchronized. • The two-way frame delay is measured using only DMM and DMR frames.

Topology

Let's see how a round-trip frame delay is measured with this sample topology.



- The sender MEP (Router-1) transmits a frame containing delay measurement request information and the timestamp at the which router sends the DMM.

- When packets pass through each interface, timestamps are written into DMMs and DMRs at both local and peer MEPs.
- When the DMM leaves the local interface, the TX timestamp is added to the packet.
- When the receiver MEP (Router-2) receives the frame, records the timestamp at which the receiver MEP receives the frame with the delay measurement request information and the remote MEP (Router-2) responds with an DMR adding the remote TX timestamp to the packet as it leaves the remote interface.

To measure a round-trip delay for a traffic exchange between Router-1 and Router-2, four timestamps get populated as the packet moves through the network.

- Router-1 adds the TxTimestampF when DMM packet is transmitted.
- Router-2 adds RxTimestampF when DMM packet is received by it.
- Router-2 adds TxTimestampB when DMR packet it transmitted.
- Router-1 adds RxTimestampB when DMR is received by it

The round-trip delay is calculated using the following formula:

```
Delay = (RxTimestampB – TxTimestampF) - (TxTimestampB – RxTimestampF)
    = RxTimestampB – TxTimestampF - TxTimestampB + RxTimestampF
    = (RxTimestampF – TxTimestampF) - (TxTimestampB – RxTimestampB)
```

Configure Ethernet frame delay measurement for L2VPN services

This topic describes how to configure Ethernet frame delay measurement for L2VPN services.

Use this procedure to configure Ethernet Frame Delay Measurement for L2VPN Services:

Procedure

1. Configure L2VPN service.

   Example

   ```
   Router# configure
   Router(config)# l2vpn
   Router(config-l2vpn)# xconnect group evpn_vpws_203
   Router(config-l2vpn-xc)# p2p evpn_vpws_phy-100
   Router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/2.100
   Router(config-l2vpn-xc-p2p)# neighbor evpn evi 30001 target 30001 source 50001
   Router(config-l2vpn-xc-p2p)# commit
   ```

2. Enable CFM service continuity check.

   Example

   ```
   Router# configure
   Router(config)# l2vpn
   Router(config-l2vpn)# xconnect group evpn_vpws_203
   Router(config-l2vpn-xc)# p2p evpn_vpws_phy-100
   Router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/2.100
   Router(config-l2vpn-xc-p2p)# neighbor evpn evi 30001 target 30001 source 50001
   Router(config-l2vpn-xc-p2p)# commit
   ```

3. Enable CFM on the interface.

Example

```
Router(config)# interface GigabitEthernet0/0/0/2.100 l2transport
Router(config-subif)# encapsulation dot1q 100
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# mtu 9100
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain bd-domain service bd-service mep-id 4001
Router(config-if-cfm-mep)# sla operation profile test-profile1 target mep-id
 1112
Router(config-if-cfm-mep)# commit
```

4. Configure Ethernet frame delay measurement.

Example

```
Router(config)# ethernet sla
Router(config-sla)# profile EVC-1 type cfm-delay-measurement
Router(config-sla-prof)# probe
Router(config-sla-prof-pb)# send packet every 1 seconds
Router(config-sla-prof-pb)# schedule
Router(config-sla-prof-schedule)# every 3 minutes for 120 seconds
Router(config-sla-prof-schedule)# statistics
Router(config-sla-prof-stat)# measure round-trip-delay
Router(config-sla-prof-stat-cfg)# buckets size 1 probes
Router(config-sla-prof-stat-cfg)# buckets archive 5
Router(config-sla-prof-stat-cfg)# commit
```

5. Verify the configuration.

Example

```
Router# show ethernet cfm local meps interface GigabitEthernet0/0/0/2.100
verbose
Up MEP on GigabitEthernet0/0/0/2.100 MEP-ID 4001
================================================================================
Interface state: Up MAC address: 0c11.6752.3af8
Peer MEPs: 1 up, 0 with errors, 0 timed out (archived)
CCM generation enabled: Yes, 10s (Remote Defect detected: No)
AIS generation enabled: No
Sending AIS: No
Receiving AIS: No
Sending CSF: No
Receiving CSF: No
Packet Sent Received
------ ---------- ------------------------------------------------------------
CCM 19 9 (out of seq: 0)
DMM 473 0
DMR 0 473
```

In this example, observe that the sent and received DMM and DMR packets are same. So there is no delay in frame transimission.

Minimum delay bin

This topic describes how Y.1731 Ethernet SLA collects and optionally aggregates performance metrics over time, and how the minimum-delay bin feature lets you configure a distinct width for the first bin to prevent wastage of bins that might otherwise be empty due to inherent speed of light delays.

The Y.1731 Ethernet SLA is a performance monitoring feature that

- collects and aggregates performance metrics over time
- organizes data into buckets, and
- utilizes bins to reduce memory consumption.

The system manages performance data through these mechanisms:

- The system organizes data collected during specific timeframes into buckets.
- The system utilizes bins to store aggregated data, which reduces overall memory consumption.
- The minimum-delay bin feature allows the administrator to configure a distinct width for the first bin to prevent the wastage of empty bins.
- The remaining bins capture variations in observed delays.

Table 22: Feature History Table

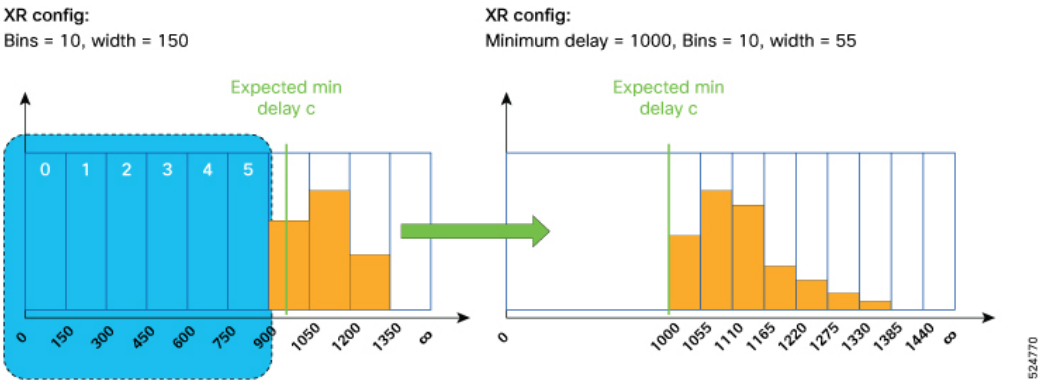
Feature Name	Release Information	Feature Description
Minimum-delay bin	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100]) (select variants only*) *This feature is supported on Cisco 8404-SYS-D router.
Minimum-delay bin	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]). For statistics aggregation, you can now configure a distinct width for the first bin to adjust for large propagation delay. By using this feature, you can avoid wasting several bins that would be empty in some unavoidable situations such as delay due to speed of light limitations. The feature introduces these changes: CLI: - The **minimum-delay** keyword is introduced in the **aggregate** and **cfm-delay-measurement probe** commands. YANG Data Models: New XPaths for - `Cisco-IOS-XR-infra-sla-cfg.yang` - `Cisco-IOS-XR-um-ethernet-sla-cfg.yang` - `Cisco-IOS-XR-infra-sla-oper.yang` (see GitHub, YANG Data Models Navigator)

Information about minimum-delay bin

In situations where a delay is expected in first few iterations, you can specify the width of the first bin independently using the **minimum-delay** keyword in statistics aggregation configuration. See below examples to configure aggregation with and without minimum-delay support.

The below figure shows the comparision between statistics aggregation before and after minimum-delay configuration.

Figure 9: Comparision diagram before and after minimum-delay configuration



Example 1: Configuring aggregation without minimum-delay

```
statistics measure round-trip-delay aggregate bins 10 width 150
```

This configuration has 10 bins and the results being aggregated into the ranges 0-150, 150-300, 300-450 and so on, until 1350+, where the last bin has infinite width to hold all values greater than 1350.

Example 2: Configuring aggregation with minimum-delay

```
statistics measure round-trip-delay aggregate bins 10 width 55 minimum-delay
1000
```

Here, the width of the first bin is 1000ms and not 150ms. The width of the other nine bins are aggregated into 1000-1055, 1055-1110, 1110-1165, and so on. This leads to increased resolution with the same number of bins, as all the bins are utilized.

> **Note**
>
> To specify the values of width and minimum-delay in microseconds instead of milliseconds, you must use the **usec** keyword. For more information, see **aggregate** and **cfm-delay-measurement probe** commands.

Configure minimum delay bin support

This topic describes how to configure an SLA operation profile with aggregation, and then set the width of the first bin using the minimum-delay keyword to adjust for large propagation delays.

Before you begin

Make sure that the number of bins for aggregate configuration is at least two.

Procedure

1. Configure an SLA Operation profile and statistics measurement for the Profile.

Example

Configure Ethernet Frame Delay Measurement for L2VPN Services.

```
Router(config)# ethernet sla
Router(config-sla)# profile EVC-1 type cfm-delay-measurement
Router(config-sla-prof)# probe
Router(config-sla-prof-pb)# send packet every 1 seconds
Router(config-sla-prof-pb)# schedule
Router(config-sla-prof-schedule)# every 3 minutes for 120 seconds
Router(config-sla-prof-schedule)# statistics
Router(config-sla-prof-stat)# measure round-trip-delay
Router(config-sla-prof-stat-cfg)# buckets size 1 probes
Router(config-sla-prof-stat-cfg)# buckets archive 5
Router(config-sla-prof-stat-cfg)# commit
```

2. Configure aggregation for the SLA profile and then configure the width of the first bin by using the **minimum-delay** keyword.

Example

Configure aggregation with bin count of 10 and the width of the first bin as 5 ms.

```
Router(config-sla-prof-schedule)# statistics
Router(config-sla-prof-stat)# measure round-trip-delay
Router(config-sla-prof-stat-cfg)# aggregate bins 5 width 10 minimum-delay 30
Router(config-sla-prof-stat-cfg)# buckets size 1 probes
Router(config-sla-prof-stat-cfg)# buckets archive 5
Router(config-sla-prof-stat-cfg)# commit
```

3. View the running configuration using the **show running-config** command.

Example

```
ethernet sla
 profile EVC-1 type cfm-delay-measurement
  probe
   send packet every 1 seconds
  !
  schedule
   every 3 minutes for 120 seconds
  !
  statistics
   measure round-trip-delay
    aggregate bins 10 width 2 minimum-delay 5
    buckets size 1 probes
    buckets archive 5
   !
```

4. Verify the output by using these show commands.
   - **show** *protocol* **sla operations detail**
   - **show** *protocol* **sla probes**
   - **show** *protocol* **sla statistics**

Example

```
router# show ethernet sla statistics history detail on-demand
Bucket started at 15:38 on Tue 02 Jul 2024, lasting 1 hour:
    Pkts sent: 1200; …
    Result count: 30
    Min: 13ms; Max: 154ms; Mean: 28ms; StdDev: 11ms
    Bins:
    Range               Samples    Cum. Count      Mean
    -------------  ------------  ------------  --------
     0 to 30 ms        20 (2%)       20 (2%)       37ms
    30 to 35 ms       909 (61%)     929 (77%)      67ms
    35 to 40 ms       212 (18%)    1141 (95%)      75ms
    40 to 45 ms        98 (11%)    1141 (95%)      75ms
     > 45 ms           55  (5%)    1196 (85%)      90ms
```

10 Configure Unidirectional Link Detection

Topics:

This chapter describes the Unidirectional Link Detection (UDLD) chapter, which covers UDLD operation, types of fault detection, modes of operation, aging mechanism, state machines, and limitations, and includes the task to configure UDLD on an Ethernet interface on the router.

Unidirectional Link Detection protocol

This topic describes Unidirectional Link Detection (UDLD), a Cisco-proprietary single-hop physical link protocol that monitors ethernet links, including point-to-point and shared media links, and detects link error conditions such as miswiring or unidirectional link failure that are not detected at the physical link layer.

The Unidirectional Link Detection (UDLD) is a Cisco-proprietary Ethernet OAM protocol that

- monitors Ethernet links including point-to-point and shared media links
- detects link error conditions such as miswiring or unidirectional link failure that the physical link layer misses, and
- identifies potential wiring errors in unbundled fiber links that cause mismatches between port transmitting and receiving connections.

Table 23: Feature History Table

Feature Name	Release	Description
Unidirectional link detection protocol support on physical Ethernet interfaces	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100]) (select variants only*) *This feature is supported on Cisco 8404-SYS-D router.
Unidirectional link detection protocol support on physical Ethernet interfaces	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: - 8011-32Y8L2H2FH - 8011-12G12X4Y-A/D
Unidirectional link detection protocol support on physical Ethernet interfaces	Release 25.2.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100]))(select variants only*) This release supports: - Unidirectional Link Detection Protocol (UDLD) on the physical ethernet interfaces. This feature helps detect faults and miswiring conditions with unbundled fiber links and enables each device to understand its own connections as well as those of its neighbors. *This feature is now supported on Cisco 8712-MOD-M routers.
Unidirectional link detection protocol support on physical Ethernet interfaces	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) * This feature is supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release	Description
Unidirectional link detection protocol support on physical Ethernet interfaces	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700)(select variants only*); Modular Systems (8800 [LC ASIC: P100]). The Unidirectional Link Detection Protocol (UDLD) is now supported on the Physical Ethernet interfaces on the Cisco Silicon One P100 ASIC-based Systems. This feature helps detect faults and miswiring conditions with unbundled fiber links and enables each device to understand its own connections as well as those of its neighbors. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M This feature introduces these changes: **CLI:** • **clear ethernet udld statistics** • **ethernet udld reset interface** • **show ethernet udld interfaces** • **show ethernet udld statistics**

The UDLD protocol performs these functions:

- It monitors Ethernet links including point-to-point and shared media links.
- It detects link error conditions such as miswiring or unidirectional link failure that the physical link layer misses.
- It identifies potential wiring errors in unbundled fiber links that cause mismatches between port transmitting and receiving connections.

UDLD operation

This topic describes how Unidirectional Link Detection (UDLD) exchanges PROBE, ECHO, and FLUSH protocol packets between neighboring devices to identify sender and neighbor ports, detect faults and miswiring conditions, and disable affected interfaces when a problem is found.

UDLD exchanges protocol packets between the neighboring devices. UDLD works if both devices on the link support UDLD and have it enabled on respective ports.

UDLD sends an initial PROBE message on the ports where it is configured. When it receives a PROBE message, UDLD sends periodic ECHO (hello) messages. Both the messages help identify the sender and its port, and also contain some information about the operating parameters of the protocol on that port. The messages also contain the device and port identifiers on the port for any neighbor devices that the local device has connected with on the port. Similarly, each device

gets to know where it is connected and where its neighbors are connected. This helps in detecting faults and miswiring conditions.

The protocol employs a mechanism where information from neighbors that is not periodically refreshed is eventually timed out for fault detection.

The protocol uses a FLUSH message to indicate when UDLD is disabled on a port. This causes the peers to remove the local device from their neighbor cache to prevent a time out.

If a problem is detected, UDLD disables the affected interface and notifies the user to avoid further network problems beyond traffic loss. Example: Loops which are not detected or prevented by Spanning Tree Protocol (STP).

Types of fault detection

This topic describes the transmit, miswiring, loopback, and receive faults that Unidirectional Link Detection (UDLD) can detect on a link, and the conditions under which each type of fault arises.

UDLD can detect these types of faults:

- Transmit faults – These are transmission failures from the local port to the peer device which also includes the faults caused by physical link failure or packet path issues on the local or peer device. These failures can lead to serious network issues such as loops which occur specifically when a link is unidirectional.
- Miswiring faults – These are instances that occur when using unbundled fibers to connect fiber optic ports. In such instances, the receiving and transmitting sides of a port on the local device are connected to different peer ports (on the same device or on different devices).
- Loopback faults – In these instances, the receiving and transmitting sides of a port are connected to each other, creating a loopback condition. This can be an intentional mode of operation, for certain types of testing, but UDLD must not be used in these cases.
- Receive faults – The protocol uses a heartbeat signal that is transmitted at a negotiated periodic interval to the peer device. Missed heartbeats can therefore be used to detect failures on the receiving side of the link (where they do not result in interface state changes). These could be caused by a unidirectional link with a failure only affecting the receiving side, or by a link which has developed a bidirectional fault. This detection depends on reliable, regular packet transmission by the peer device. For this reason, the UDLD protocol has two configurable modes of operation namely **Normal** mode and **Agressive** mode, which determine the behavior on a heartbeat timeout. For more information about these modes, see UDLD modes of operation on page 170.

UDLD modes of operation

This topic describes the two Unidirectional Link Detection (UDLD) modes of operation: normal mode, in which the user is informed when a receive fault is detected; and aggressive mode, in which the affected port is also disabled.

UDLD can operate in these modes:

- **Normal mode**: In this mode, if a `Receive Fault` is detected, the user is informed and no further action is taken.
- **Aggressive mode**: In this mode, if a `Receive Fault` is detected, the user is informed and the affected port is disabled.

UDLD aging mechanism

This topic describes how Unidirectional Link Detection (UDLD) ages out neighbor information when packets are not received within the hold time, which is set to three times the remote port's message interval in Cisco IOS XR Software.

Aging of UDLD information occurs in a `Receive Fault` condition when the port that runs UDLD does not receive UDLD packets from the neighbor port for a duration of the hold time. The hold time for the port is dictated by the remote port and is dependent on the message interval at the remote end. The shorter the message interval, the shorter is the hold time and faster the detection of the fault. The hold time is three times the message interval in Cisco IOS XR Software.

UDLD information can age out due to the high error rate on the port caused by a physical issue or duplex mismatch. Packet drops due to age out does not mean that the link is unidirectional. UDLD in normal mode does not disable such link.

It is important to choose the right message interval to ensure proper detection time. The message interval should be fast enough to detect the unidirectional link before the forwarding loop is created. The default message interval is 60 seconds. The detection time is approximately equal to three times the message interval. Therefore, when using default UDLD timers, UDLD does not timeout the link faster than the STP aging time.

UDLD state machines

This topic describes the two Unidirectional Link Detection (UDLD) finite state machines: the Main FSM that handles all phases of protocol operation, and the Detection FSM that determines the status of a port.

UDLD uses two types of finite state machines (FSMs), generally referred as state machines. The Main FSM deals with all the phases of operation of the protocol while the Detection FSM handles only the phases that determine the status of a port.

Main FSM

The Main FSM can be in one of these states:

- **Init**: Protocol is initializing.
- **UDLD inactive**: Port is down or UDLD is disabled.
- **Linkup**: Port is up and running, and UDLD is in the process of detecting a neighbor.
- **Detection**: A hello message from a new neighbor has been received and the Detection FSM is running to determine the status of the port.
- **Advertisement**: The Detection FSM has run and concluded that the port is operating correctly, periodic hello messages are being sent and the hello messages from neighbors are monitored.
- **Port shutdown**: The Detection FSM detected a fault, or all neighbors were timed out in Aggressive mode, and the port has been disabled as a result.

Detection FSM

The Detection FSM can be in one of these states:

- **Unknown**: Detection has not yet been performed or UDLD has been disabled.
- **Unidirectional detected**: A unidirectional link condition has been detected because a neighbor does not see the local device, the port will be disabled.
- **Tx/Rx loop**: A loopback condition has been detected by receiving a type, length, and value (TLV) message with the ports own identifiers, the port will be disabled.
- **Neighbor mismatch**: A miswiring condition has been detected in which a neighbor can identify other devices than the devices the local device can see and the port will be disabled.
- **Bidirectional detected**: UDLD hello messages are exchanged successfully in both the directions, the port is operating correctly.

Limitations

This topic describes the Unidirectional Link Detection (UDLD) protocol limitations on Cisco 8000 Series Routers, including peer MAC address requirements, L2VPN tunneling behavior, and Switched Port Analyzer (SPAN), subinterface, and bundle interface support.

- UDLD on Cisco 8000 Series Routers does not work if the peer UDLD configuration has custom MAC address; Peer must have either Cisco MAC address or IEEE Slow Proto MAC address.
- Use only these MAC Addresses to establish a successful connection and communication with the Cisco 8000 Series Routers.
  - cisco-l2cp (0x01000ccccccc) - Cisco proprietary MAC Address which can also be used by all other Cisco protocols.
  - ieee-slow-protocols (0x0180c2000002) - IEEE Slow Protocol MAC Address.
- UDLD is not tunneled through L2VPN like other slow protocols.
- UDLD must not be enabled on a Switched Port Analyzer (SPAN) source or a destination port.
- The UDLD protocol is not supported on the subinterfaces and bundle interfaces.

Configure UDLD

This topic describes how to configure the Unidirectional Link Detection (UDLD) protocol on a physical ethernet interface, and optionally set the mode of operation, message time, and syslog message suppression for the interface.

UDLD is configured for each interface. The interface must be a physical ethernet interface.

Perform these steps to configure UDLD protocol on an interface.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Use the **interface [GigabitEthernet | TenGigE]** *interface-path-id* command. Enter interface configuration mode and specify the Ethernet interface name and notation *rack/slot/module/port*.

   Example

   ```
   Router(config)# interface  TenGigE 0/1/0/0
   ```

   Note

   The example indicates an 8-port 10-Gigabit Ethernet interface in modular services card slot 1.

3. Enable ethernet UDLD functionality and enters interface Ethernet UDLD configuration mode.

   Example

   ```
   Router(config-if)# ethernet udld
   ```

4. Optional: Use the **mode normal** |**aggressive** command to specify the mode of operation for UDLD. The options are normal and aggressive.

   Example

   ```
   Router(config-if-udld)# mode normal
   ```

5. Optional: Use the **message-time** *[7-90]* command to specify the message time (in seconds) to use for the UDLD protocol. The value ranges from 7 to 90 seconds.

   Example

   ```
   Router(config-if-udld)# message-time 70
   ```

6. Optional: Suppress the operational UDLD syslog messages.

   Example

   ```
   Router(config-if-udld)# logging disable
   ```

7. End the configuration session and exits to the EXEC mode.

   Example

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

11 Configure Advanced Fault Propagation

Topics:

This chapter describes the Advanced Fault Propagation chapter, which covers Link Loss Forwarding (LLF), link state monitor and propagation by CFM, remote link state propagation, and restrictions for LLF, and includes tasks to configure LLF for CFM and for Layer 2 transport on the router.

Link Loss Forwarding

This topic describes Link Loss Forwarding (LLF), a mechanism used in networking to propagate the status of a network link to other connected devices.

Link Loss Forwarding (LLF) is a networking mechanism that

- propagates the status of a network link to other connected devices
- ensures that link failure information is forwarded to other network devices, and
- takes appropriate actions to maintain network stability and performance.

LLF uses Connectivity Fault Management (CFM) to transmit notification of a signal loss or fault across the network. When there is a fault on a link to a device on one side of the network, the connection to the port on the other side needs to be shutdown so that the device re-routes the traffic.

You can enable LLF on a network by one of these methods:

- Link State Monitor and Propagation by CFM: LLF uses Connectivity Fault Management (CFM) to transmit notification of a signal loss or fault across the network. When there is a fault on a link to a device on one side of the network, the connection to the port on the other side needs to be shutdown so that the device re-routes the traffic.
- Remote Link State Propagation: LLF uses this method for Layer 2 transport events to propagate link failures to remote endpoints. When a link failure occurs, LLF ensures that the failure is communicated to other devices in the network. This enables the other devices to take appropriate action, such as rerouting traffic or triggering failover mechanisms.

Restrictions for link loss forwarding for CFM

This topic describes the restrictions that apply when you use Link Loss Forwarding (LLF) for Connectivity Fault Management (CFM), including the supported interface and MEP types, the damping timer behavior for interface state transitions, and platform-specific limitations.

Supported interface and MEP types for LLF

Configure Link Loss Forwarding only on the interface and MEP types that support it.

- Link loss forwarding is not permitted on subinterfaces.
- Link loss forwarding is permitted only on UP MEPs. The UP MEPs send the frames into the bridge relay function and not through the wire connected to the port where the MEP is configured. For more information on UP MEPs, see Maintenance End Points and Connectivity Fault Management processes on page 131.
- Link loss forwarding does not work on bundle interfaces configured with LACP.

Damping timer behavior for interface state transitions

A damping or restore timer governs transitions of an interface from TX-disabled state to TX-enabled state.

- By default, the period of the damping timer is three times the configured CCM interval. You cannot configure the damping timer.
- The damping timer is not provided for transitions of an interface from TX-enabled state to TX-disabled state.

TX_LOS alarm not displayed on Cisco 8404-SYS-D with CFM LLF enabled

In the Cisco 8404-SYS-D router, with CFM LLF feature enabled, and when there is a remote fault, and on the local side, the Tx-disable error message is triggered. The **show controller optics** command does not display the TX_LOS alarm for XR SFP+ 10G module optics.

Link state monitor and propagation by CFM

This topic describes how Link Loss Forwarding (LLF) uses Connectivity Fault Management (CFM) to transmit notification of a signal loss or fault across the network, and how the receiving interface is transitioned to TX-disable state and then restored based on a restore or damping timer.

Link State Monitoring is a network management mechanism that

- tracks the status of network links to ensure they are operational and performing as expected
- propagates link state changes throughout the network so other devices can adjust routing and operations accordingly, and
- manages interface transitions such as TX-disable to maintain network stability.

When there is a fault on a link to a device on one side of the network, the connection to the port on the other side needs to be shutdown so that the device re-routes the traffic. This requires the interface to be TX-disabled.

Link Loss Forwarding (LLF) uses Connectivity Fault Management (CFM) to transmit notification of a signal loss or fault across the network. If a local attachment circuit (AC) on a bridged interface fails, one of the following signals or packet types are sent to the neighboring device:

- Continuity Check Message (CCM) – The CCMs are heartbeat messages exchanged periodically between all the Maintenance End Points (MEPs) in a service. MEPs are members of a particular service within a domain and are responsible for sourcing and sinking CFM frames. Each MEP sends out multicast CCMs, and receives CCMs from all the other MEPs in the service. This allows each MEP to discover its peer MEPs, and to verify that there is connectivity between them.
- Alarm Indication Signal (AIS) – These are messages sent periodically by MEPs that have detected a fault, to the MEPs in the next highest maintenance domain level.
- Client Signal Fail (CSF) – A mechanism for error detection. When a MEP detects an issue, the MEP sends CSF packets to its peer MEPs. For more information on MEPs, see Maintenance points on page 130.

How interface state transition works

This topic describes how the Connectivity Fault Management Daemon (CFMD) and Ether-MA coordinate interface state transitions.

The Connectivity Fault Management Daemon (CFMD) and Ether-MA are processes that run on the router's control plane, coordinating interface state transitions based on fault notifications. Ether-MA manages owner channel communication and resynchronization from CFMD, L2VPN, and other Ether-MA processes. It handles TX-disable and TX-enable events triggered by CFMD notifications.

Summary

These actors and components participate in the process:

- Connectivity Fault Management Daemon (CFMD): Receives Continuity Check Message (CCM), Alarm Indication Signal (AIS), or Client Signal Fail (CSF) fault indications and notifies Ether-MA.
- Ether-MA: Handles TX-disable and TX-enable events on the interface based on notifications from CFMD.
- Restore or damping timer: Runs for 3.5 times the packet interval duration and determines when the interface returns to the TX-enable state.
- Interface: Transitions to the TX-disable state on a fault and back to the TX-enable state when the restore timer expires without additional faults.

Workflow

This coordinated process ensures rapid response to link faults and controlled restoration of interface operation to maintain network stability and performance.

These stages describe how the router handles a fault notification and transitions the interface between the TX-disable and TX-enable states:

1. Stage 1 – CFMD receives a fault indication. When the system receives a Continuity Check Message (CCM) or Alarm Indication Signal (AIS) with a fault indication, or a Client Signal Fail (CSF) error packet, CFMD communicates with Ether-MA to transition the interface to the TX-disable state.
2. Stage 2 – The interface transitions to TX-disable. Upon receiving a fault notification, the interface is immediately transitioned to the TX-disable state.
3. Stage 3 – The restore or damping timer starts. A restore or damping timer with a 3.5 * packet interval duration is started.
4. Stage 4 – The interface transitions back to TX-enable. If no additional fault packets are received before the restore timer expires, the TX-disable state is cleared, and the interface transitions back to the TX-enable state.

Remote link state propagation

This topic describes how Remote Link State Propagation communicates link status to remote devices in Layer 2 transport networks, and how Link Loss Forwarding (LLF) uses it to propagate link failures across the pseudowire so that neighboring devices can reroute traffic and trigger failover.

Remote link state propagation is a LLF feature that

- allows the status of a link to be communicated to remote devices
- ensures that all relevant parts of the network are aware of link state changes, and
- maintains accurate link status information crucial for network performance and reliability, which is particularly useful in Layer 2 transport networks.

Table 24: Feature History Table

Feature Name	Release Information	Feature Description
Remote link state propagation	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100]) (select variants only*) *This feature is supported on Cisco 8404-SYS-D router.
Remote link state propagation	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) (select variants only*) *This feature is supported on: - 8711-48Z-M - 8011-32Y8L2H2FH - 8011-12G12X4Y-A/D

Feature Name	Release Information	Feature Description
Remote link state propagation	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Remote Link State Propagation	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *This feature is now supported on: • 8212-32FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E
Remote link state propagation	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) Remote Link State Propagation allows the status of a link to be communicated to remote devices, ensuring that all relevant parts of the network are aware of link state changes. Link Loss Forwarding (LLF) uses this feature to propagate link failures to remote endpoints. By enabling remote state propagation and LLF on an interface, you can ensure that the link state changes are communicated to remote devices, allowing for quick failover and rerouting of traffic. This feature introduces the **propagate remote-status** command.

Link Loss Forwarding for Layer 2 Transport

Link Loss Forwarding (LLF) uses Remote Link State Propagation to propagate link failures to remote endpoints. When a link failure occurs, LLF ensures that the failure is communicated to other devices in the network, allowing them to take appropriate action, such as rerouting traffic or triggering failover mechanisms. LLF helps avoid packet loss and triggers network convergence through alternate links. It works by sending signals across the pseudowire (PW) to the neighboring device, bringing the PW and far-end attachment circuit (AC) down if the local AC goes down.

You can configure LLF for the Layer 2 transport events, using the **propagate remote-status** command. You can enable LLF on the following interface types: 1G, 10G, 25G, 40G, 100G, and 400G.

Remote State Propagation with LLF for L2 Transport

When you configure LLF for L2 transport events, the following are the processes that happen:

1. Link State Detection: The network device detects a change in the link state, such as a link going down.
2. Remote State Propagation: When remote state propagation is enabled, the detected link state change is propagated to remote devices. This ensures that all relevant devices are aware of the link failure and can take appropriate action.
3. LLF Activation: LLF uses the propagated link state information to trigger failover mechanisms or reroute traffic. This helps maintain network performance and reliability by quickly responding to link failures.

When you enable remote state propagation and LLF on an interface, the link state changes are communicated to remote devices allowing for quick failover and rerouting of traffic.

Configure link loss forwarding for CFM

This topic describes how to configure LLF on a network by configuring a CFM domain and service, a MEP under the CFM domain and service, the continuity check message (CCM) interval on the MEP, Client Signal Fail (CSF) transmission and logging, and by enabling LLF on an interface.

Use this procedure to configure LLF on a network.

Procedure

1. Configure a Connectivity Fault Management (CFM) domain and service.

   Example

   ```
   Router# configure
   Router(config)# ethernet cfm
   Router(config-cfm)# domain dom1 level 1 service ser1 bridge group up-meps
   bridge-domain up-mep
   ```

   The CFM domain and service are established.
2. Configure a Maintenance End Point (MEP) under the CFM domain and service.

   Example

   ```
   Router(config)# interface GigabitEthernet0/2/0/0
   Router(config-if)# ethernet cfm
   Router(config-if-cfm)# mep domain dom1 service ser1 mep-id 1
   ```

   The MEP is defined under the specified CFM domain and service.
3. Configure continuity check message (CCM) interval on the MEP. By default, the restore timer for a CCM notification is calculated based on the configured CCM interval.

   Example

   ```
   Router(config-cfm-dmn-svc)# continuity-check interval 1m
   ```

   The CCM interval is set on the MEP.

4. Configure Client Signal Fail (CSF) transmission on the MEP, to enable CSF transmission.
5. Configure CSF logging on the MEP, to enable logging on receiving a CSF packet.

   Example

```
Router(config-cfm-dmn-svc)# csf interval 1m cos 4
Router(config-cfm-dmn-svc)# csf-logging
```

> **Note**
>
> The CSF configuration is optional and is not required when both the devices in CFM-protected network are running with IOS-XR. This configuration is required for inter-operation with certain client-end setups that contain devices from other clients.

   CSF transmission is enabled and logging is active upon receiving a CSF packet.
6. Enable LLF on an interface using the **propagate-remote-status** command. This command triggers the interface to be TX-disabled on fault detection.

   Example

```
Router(config-if-cfm-mep)# propagate-remote-status
Router(config-if-cfm-mep)# commit
```

7. Verify the LLF configuration and fault state for each interface

   Example

```
Router# show ethernet cfm interfaces llf location 0/RP0/CPU0
Defects (from at least one peer MEP):
A - AIS received I - Wrong interval
R - Remote Defect received V - Wrong Level
L - Loop (our MAC received) T - Timed out (archived)
C - Config (our ID received) M - Missing (cross-check)
X - Cross-connect (wrong MAID) U - Unexpected (cross-check)
P - Peer port down F - CSF received
GigabitEthernet0/1/0/0
MEP Defects Restore Timer
----- ---------------------------------- --------------
100 R Not running
101 None 10s remaining
102 RPF Not running
GigabitEthernet0/1/0/1
MEP Defects Restore Timer
----- ---------------------------------- --------------
110 None 3s remaining
GigabitEthernet0/1/0/2
MEP Defects Restore Timer
----- ---------------------------------- --------------
120 P 5s Not running
```

   The following output shows that the interface received a single CSF packet at 1 minute interval, so that the interface is TX-disabled with a damping timer of 3.5 minutes.

Example

```
Router# show ethernet cfm local meps detail
Domain dom1 (Level 1), Service ser1
UP MEP on GigabitEthernet0/1/0/0 MEP-ID 1
================================================================================
Interface state: UP MAC address: 0204.3dbe.c93b
Peer MEPs: 0 up, 0 with errors, 0 timed out (archived)
Cross-check errors: 0 missing, 0 unexpected
CCM generation enabled: No
AIS generation enabled: No
Sending AIS: No
Receiving AIS: No
Sending CSF: No
Receiving CSF: Yes (Interval: 1min, started 00:03:29 ago)
TX Disable triggered: Yes (restore timer not running)
```

The following output shows that the interface received a CCM notification that the peer MEP port is down, so that the interface is TX-disabled.

Example

```
Router# show ethernet cfm local meps detail
Domain dom1 (Level 1), Service ser1
UP MEP on GigabitEthernet0/1/0/0 MEP-ID 1
================================================================================
Interface state: UP MAC address: 0204.3dbe.c93b
Peer MEPs: 1 up, 1 with errors, 0 timed out (archived)
Cross-check errors: 0 missing, 0 unexpected
CCM generation enabled: Yes, 1min (Remote Defect detected: Yes)
CCM defects detected: P - peer port down
AIS generation enabled: No
Sending AIS: No
Receiving AIS: No
Sending CSF: No
Receiving CSF: No
TX Disable triggered: Yes (restore timer 5s not running)
```

The following output shows that the interface received CCM notification that the peer MEP port is up, and restore timer is started for the TX-disabled interface.

Example

```
Router# show ethernet cfm local meps detail
Domain dom1 (Level 1), Service ser1
UP MEP on GigabitEthernet0/1/0/0 MEP-ID 1
================================================================================
Interface state: UP MAC address: 0204.3dbe.c93b
Peer MEPs: 1 up, 0 with errors, 0 timed out (archived)
Cross-check errors: 0 missing, 0 unexpected
CCM generation enabled: Yes, 1min (Remote Defect detected: No)
AIS generation enabled: No
Sending AIS: No
Receiving AIS: No
Sending CSF: No
Receiving CSF: No
TX Disable triggered: Yes (restore timer running, 1183ms remaining)
```

The following output shows Ether-MA configured bundles and their members:

Example

```
Router# show ethernet infra internal ether-ma bundles
Bundle interface: Bundle-Ether1 (TX disabled)
Bundle members:
GigabitEthernet0/1/0/1
GigabitEthernet0/1/0/2
Bundle interface: Bundle-Ether2
Bundle members:
GigabitEthernet0/2/0/1
```

Configure link loss forwarding for Layer 2 transport

This topic describes how to configure LLF for Layer 2 transport events by entering interface mode, configuring l2transport, and enabling LLF using the propagate remote-status command.

Use this procedure to configure LLF for Layer 2 transport events.

Procedure

1. To enable link loss forwarding for Layer transport events, enter the interface mode, configure **l2transport**, and then enable LLF using the **propagate remote-status** command.

```
Router(config)# interface tenGigE 0/0/0/1
Router(config-if)# l2transport
Router(config-if-l2)# propagate remote-status
Router(config-if-l2)# commit
```

2. View the running configuration.

```
interface TenGigE 0/0/0/1
   l2transport
      propagate remote-status
     !
```

12 Configure Integrated Routing and Bridging

Topics:

This chapter describes the Integrated Routing and Bridging (IRB) chapter, which covers the Bridge-Group Virtual Interface (BVI), packet flows between bridged and routed domains, supported environments, and tasks to configure a BVI, Layer 2 attachment circuits, and bridge domains.

Integrated Routing and Bridging

This topic describes how Integrated Routing and Bridging (IRB) provides the ability to route between a bridge group and a routed interface.

The Integrated Routing and Bridging (IRB) is a networking feature that

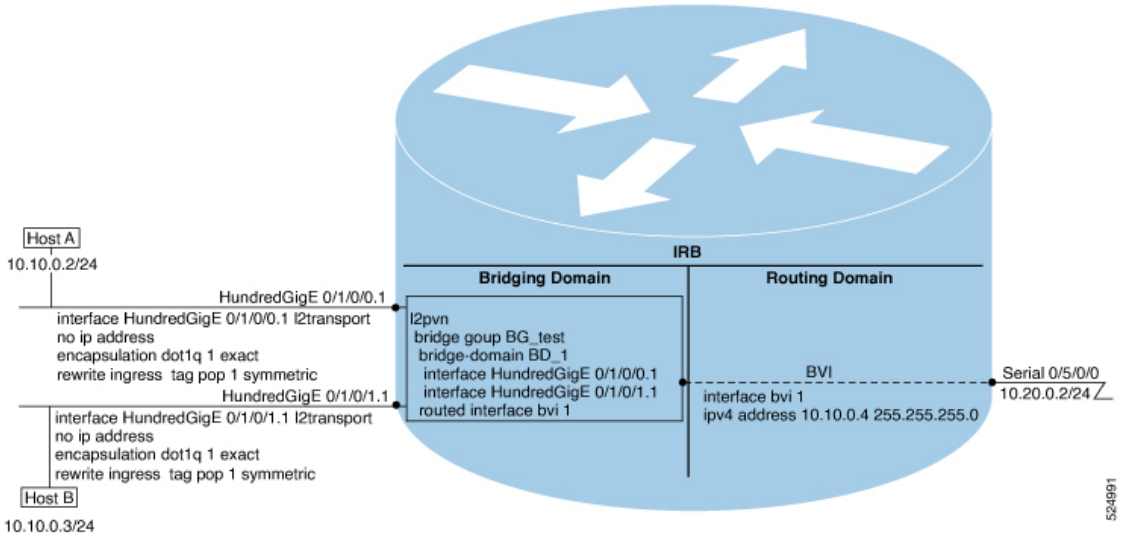
- routes traffic between bridge groups and routed interfaces
- utilizes Bridge-Group Virtual Interfaces (BVI) for connectivity, representing the link between the bridging and the routing domains on the router, and
- supports receipt of packets from a bridged interface that are destined to a routed interface by configuring BVI with the appropriate IP addresses and relevant Layer 3 attributes.

The IRB provides the ability to route traffic between a bridge group and a routed interface through a BVI. A BVI functions as a virtual interface within the router and acts like a standard routed interface. This interface links the bridging and routing domains by associating with a single bridge domain. To support the receipt of packets from a bridged interface that are destined for a routed interface, the system configures the BVI with the appropriate IP addresses and relevant Layer 3 attributes. While previous software releases required physical cabling to connect the egress Layer 2 bridge domain to a Layer 3 routing domain, Cisco IOS XR Release 4.0.1 and later versions accomplish this functionality using the BVI configuration.

The BVI configuration requires these implementation steps:

- The network administrator associates the BVI with a single bridge domain.
- The system assigns the appropriate IP addresses and Layer 3 attributes to the BVI.
- The router enables the BVI to act as the primary link between the bridging and routing domains.

Figure 10: IRB Functional View and Configuration Elements



Bridge-Group Virtual Interface

This topic describes the Bridge-Group Virtual Interface (BVI), which is a virtual interface that acts as a gateway for the corresponding bridge domain to a routed interface within the router.

The Bridge-Group Virtual Interface (BVI) is a virtual interface that

- acts as a gateway between a bridge domain and a routed interface

- supports Layer 3 attributes, and
- utilizes a configurable MAC address.

BVI functions as a virtual interface within the router and provides a gateway for the corresponding bridge domain to a routed interface. While it does not support bridging itself, it maintains specific operational characteristics:

- Uses a MAC address taken from the local chassis MAC address pool, unless overridden at the BVI interface.
- Is configured as an interface type using the **interface BVI** command and uses an IPv4 address that is in the same subnet as the hosts on the segments of the bridged domain. The BVI also supports secondary addresses.
- The BVI identifier is independent of the bridge-domain identifier. These identifiers do not need to correlate like they do in Cisco IOS software.
- Is associated to a bridge group using the **routed interface BVI** command.
- BVI interfaces support a number range of 1 to 4294967295.

Supported features on BVI

This topic lists the routing protocols, forwarding features, and interface commands supported on a Bridge-Group Virtual Interface (BVI) on the Cisco 8000 Series Router.

The following are the supported features on BVI:

- Border Gateway Protocol (BGP)
- Bidirectional Forwarding Detection (BFD)
- Open Shortest Path First (OSPF)
- Integrated Intermediate System-to-Intermediate System (IS-IS)
- Netflow
- Access Control Lists

These interface commands are supported on a BVI:

- **arp purge-delay**
- arp timeout
- **bandwidth** (The default is 10 Gbps and is used as the cost metric for routing protocols for the BVI)
- **ipv4**
- **ipv6**
- **mac-address**
- **mtu** (The default is 1500 bytes)
- **shutdown**
- The BVI supports IP helper addressing and secondary IP addressing.

BVI MAC address

This topic describes the default MAC address behavior for Bridge-Group Virtual Interfaces (BVIs) on the Cisco 8000 Series Router and how to override the default by configuring a unique MAC address at the BVI interface.

By default, the Cisco 8000 Series Router uses one MAC address for all BVI interfaces on the router. However, this means that the MAC address is not unique globally. If you want to override the default and specify a unique MAC address at the BVI, then you can configure it at the BVI interface.

BVI interface and line protocol states

This topic describes the interface and line protocol states of a Bridge-Group Virtual Interface (BVI), including the conditions under which each state transitions to Up or Down based on the bridge domain, bridge ports, and IP address configuration.

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

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

    Note

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

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

Packet flows using IRB

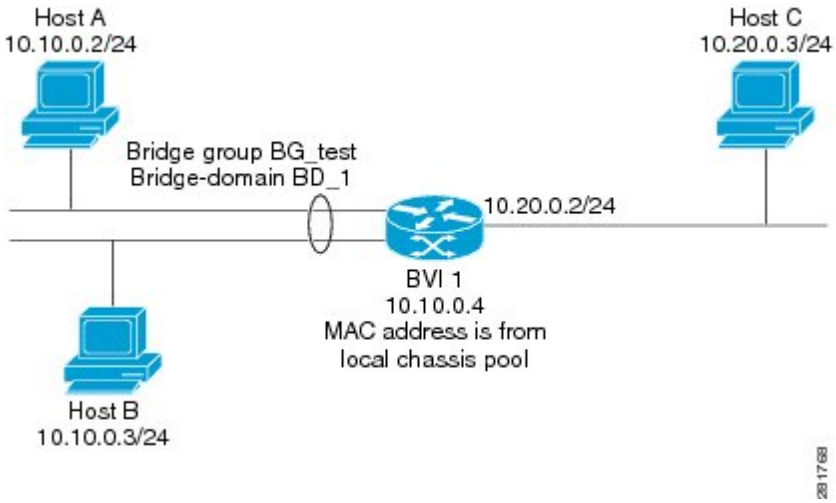
This topic describes the packet flows between hosts when Integrated Routing and Bridging (IRB) is configured on a router, using an example with three hosts to illustrate how packets are bridged or routed through the Bridge-Group Virtual Interface (BVI).

The IRB packet flow is a networking process that

- resolves ARP requests between hosts and the BVI
- directs traffic between bridged and routed interfaces, and
- manages packet forwarding through the bridge or routing engines.

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

Figure 11: IRB packet flows between hosts



The router performs these processing steps when IRB is configured:

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

Packet flows when host A sends to host B on the bridge domain

This topic describes the packet flow when host A sends data to host B on the same bridge domain and subnet, where no routing occurs and packets are bridged between the segment interfaces on the router.

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

Packet flows when host A sends to host C from the bridge domain to a routed interface

This topic describes the packet flow when host A sends data to host C from the IRB bridging domain to the routing domain, showing how the BVI receives the packet and forwards it through a routed interface with updated MAC and IP addressing.

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

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

    - IP destination address is the IP address of host C (10.20.0.3).

- Interface 10.20.0.2 sees receipt of a packet from the routed BVI 10.10.0.4. The packet is then routed through interface 10.20.0.2 to host C.

Packet flows when host C sends to host B from a routed interface to the bridge domain

This topic describes the packet flow when host C sends data to host B from the IRB routing domain to the bridging domain, showing how the routing engine captures the packet destined for the BVI and forwards it through the bridge domain.

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

- The packet comes into the routing domain with the following information:
    - MAC source address–MAC of host C.
    - MAC destination address–MAC of the 10.20.0.2 ingress interface.
    - IP source address–IP address of host C (10.20.0.3).
    - IP destination address–IP address of host B (10.10.0.3).
- When interface 10.20.0.2 receives the packet, it looks in the routing table and determines that the packet needs to be forwarded to the BVI at 10.10.0.4.
- The routing engine captures the packet that is destined for the BVI and forwards it to the BVI's corresponding bridge domain. The packet is then bridged through the appropriate interface if the destination MAC address for host B appears in the bridging table, or is flooded on all interfaces in the bridge group if the address is not in the bridging table.

Supported environments for IRB

This topic lists the environments and configuration elements supported with Integrated Routing and Bridging (IRB) on the Cisco 8000 Series Router, including routing protocols, redundancy protocols, DHCP, IP SLA, and load balancing.

These environments and configuration elements are supported with IRB on the Cisco 8000 Series Router:

- Configuration of one BVI per bridge domain.
- Virtual Private LAN Service (VPLS) virtual forwarding instance (VFI) configuration associated with a bridge domain configured with a BVI.
- BGP PIC edge for BVI-based prefixes.
- Traffic forwarding for the BVI using Open Shortest Path First (OSPF), Intermediate System-to-Intermediate System (IS-IS), and Border Gateway Protocol (BGP).
- Internet Group Management Protocol (IGMP) static groups.
- Dynamic Host Configuration Protocol (DHCP) relay agent. When DHCP relay is used from an aggregation node to obtain an IP address, the default gateway will be the IP address configured on the BVI. The BVI IP address should be in a common subnet as the DHCP pool that is being used by the aggregation node to assign IP addresses.
- Virtual Router Redundancy Protocol (VRRP) configuration and priority.
- Hot Standby Router Protocol (HSRP).
- Up to 255 VRRP or HSRP virtual MAC addresses can be configured on a physical or bundle interface, with each VRRP or HSRP ID assigned to only one BVI interface.
- Bridging of non-IP packets on a bridge domain configured with a BVI.
- Parity with stateful protocol support as currently supported on Layer 3 subinterfaces on the Cisco 8000 Series Router.

- IP SLA support as currently supported on Layer 3 subinterfaces on the Cisco 8000 Series Router.
- Load balancing of BVIs as ECMP paths (up to 32 paths).
- Interface-MIB.
- Packet counters for BVI interfaces.

Additional IPv4-specific environments supported for IRB

This topic lists the additional IPv4-specific environments supported with Integrated Routing and Bridging (IRB), including Layer 3 IP multicast bridging to Layer 2 subinterfaces and per-VPN label VRFs for IPv4.

- Layer 3 IP multicast, with ability to take ingress IP multicast traffic and bridge it to multiple Layer 2 subinterfaces (Ethernet flow points) on a bridge domain that are part of multicast groups.
- VRFs for IPv4 (Per-VPN label VRFs only—not per prefix).

Additional IPv6-specific environments supported for IRB

This topic describes the additional IPv6-specific environments supported with Integrated Routing and Bridging (IRB), including Cisco IPv6 Provider Edge Router over MPLS (6PE) and IPv6 VPN Provider Edge (6VPE) support with BVI interfaces at the customer-edge side.

Cisco IPv6 Provider Edge Router over MPLS (6PE) and IPv6 VPN Provider Edge (6VPE) support with BVI interfaces at the customer edge (CE)-facing side of the Cisco 8000 Series Router as the PE device with the following restriction:

Only per-VRF label allocation is supported (using the **label-allocation-mode per-vrf** command). For a configuration example, see the 6PE or 6VPE with BVI configuration: example.

Configuration guidelines and restrictions for IRB

This topic describes the guidelines and restrictions to configure Integrated Routing and Bridging (IRB), including BVI-to-bridge-domain limits, unsupported features on the BVI, supported bridge-domain characteristics, subinterface encapsulation rules, and QoS restrictions.

BVI-to-bridge-domain relationship

Before configuring IRB, consider these restrictions:

- Only one BVI can be configured in any bridge domain.
- The same BVI can not be configured in multiple bridge domains.
- The BVI must be assigned to an IPv4 or IPv6 address that is in the same subnet as the hosts in the bridged segments.
- If the bridged network has multiple IP networks, then the BVI must be assigned secondary IP addresses for each network.

Features not supported on the BVI

The following features are not supported on the BVI:

- IP fast reroute (FRR)
- MoFRR
- Traffic mirroring
- Unnumbered interface for BVI

Supported features and protocols with IRB

The following features and protocols are supported when configuring IRB:

- Multi protocol Label Switching (MPLS) on BVI is supported.
- IRB with 802.1ah (BVI and Provider Backbone Bridge (PBB) should not be configured in the same bridge domain).
- PIM snooping. (Need to use selective flood.)
- VRF-aware DHCP relay is supported.

Supported bridge-domain characteristics for BVIs

BVIs are supported only on bridge domains with the following characteristics:

- The bridge domain supports single and double-tagged dot1q- and dot1ad-encapsulated EFPs with non-ambiguous or "exact match" EFP encapsulations. Single and double-tagged encapsulation can be specified as long as the **rewrite ingress tag pop symmetric** command is configured.
- All Layer 2 tags must be removed. VLAN ranges are not supported.
- Untagged EFPs are supported.

Subinterface encapsulation combinations on the same physical interface

To use the sub-interface configurations **encapsulation dot1ad** (or **encapsulation dot1q**) and **encapsulation dot1ad second-dot1q any** (or **encapsulation dot1q second-dot1q any**) together on the same physical interface, use the **exact** keyword as shown below. Else, it results in traffic loss.

- ```
  Router(config)# interface hundredGigE 0/0/0/0.0
  Router(config-subif)# encapsulation dot1ad 200 exact
  ```

- ```
  Router(config)# interface hundredGigE 0/0/0/0.1
  Router(config-subif)# encapsulation dot1ad 200 dot1q any
  Router(config-subif)# commit
  ```

QoS on BVI restriction

QoS on BVI is not supported on Cisco 8000 routers. Any QoS policy configuration on BVI will be rejected.

Prerequisites to configure IRB

This topic describes the prerequisites to configure Integrated Routing and Bridging (IRB), including user group and task ID requirements, line card selection, Layer 3 information for the Bridge-Group Virtual Interface (BVI), MAC address planning, and routing advertisement.

User group and task ID requirements

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

Prerequisites to complete before configuring IRB

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

- Confirm that you are configuring the required line cards where you plan to support IRB in support of both Layer 3 to Layer 2 traffic flows and Layer 2 to Layer 3 traffic flows
- Know the IP addressing and other Layer 3 information to be configured on the bridge virtual interface (BVI). For more information, see the Guidelines and restrictions to configure IRB.

- Complete MAC address planning if you decide to override the common global MAC address for all BVIs.
- Be sure that the BVI network address is being advertised by running static or dynamic routing on the BVI interface.

Configure BVI

This topic describes how to configure a Bridge-Group Virtual Interface (BVI) by creating the interface, assigning IP addresses, and optionally setting ARP timers, bandwidth, MAC address, and MTU for the interface.

To configure a BVI, complete the following steps.

Consider the following guidelines when configuring the BVI:

- The BVI must be assigned to an IPv4 or IPv6 address that is in the same subnet as the hosts in the bridged segments.
- If the bridged network has multiple IP networks, then the BVI must be assigned secondary IP addresses for each network.

Procedure

1. Run the **configure** command to enter the global configuration mode.

   Example

   ```
   Router#configure
   ```

2. Run the **interface bvi** *identifier* command to specify and create a BVI, where *identifier* is a number from 1-65535.

   Example

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

3. Specify a primary or secondary IPv4 address or an IPv6 address for the interface.

   Example

   ```
   Router(config-if)#ipv4 address 10.10.0.4 255.255.255.0
   ```

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

   Example

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

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

5. (Optional) Run the **arp timeout** *seconds* command to configure the duration that the dynamic entries learned on the interface remain in the ARP cache.

Example

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

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

6. (Optional) Specify the amount of bandwidth (in kilobits per second) to be allocated on the interface.

   Example

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

   This bandwidth number is used as the cost metric in routing protocols for the BVI. The range is 0-4294967295. The default value is 10000000 (10 Gbps).

7. (Optional) Specify the 48-bit MAC address for the BVI as three dotted-hexadecimal values, which override the use of the default MAC address.

   Example

   ```
   Router(config-if)#mac-address 1111.2222.3333
   ```

   The range for each value is 0000 to ffff. A MAC address of all 0s is not supported.

8. (Optional) Run the **mtu** *bytes* command to configure the maximum transmission unit (MTU) size for packets on the interface.

   Example

   ```
   Router(config-if)# mtu 2000
   ```

   The range is 64-65535. The default value is 1514.

9. Run the **end** or **commit** commands to save the configuration changes.

   Example

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

   or

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

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

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

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

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

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

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

Configure Layer 2 AC interfaces

This topic describes how to configure the Layer 2 attachment circuit (AC) interfaces for routing by a Bridge-Group Virtual Interface (BVI), including enabling Layer 2 transport mode, applying IEEE 802.1Q encapsulation, and rewriting ingress tags.

Before you begin

The interfaces to be configured as Layer 2 ACs in the bridge domain and routed by a BVI must be located on the Cisco 8000 Series Router.

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

Procedure

1. Run the **configure** command to enter the global configuration mode.

   Example

   ```
   Router#configure
   ```

2. Run the **interface** [**HundredGigE**] *interface-path-id*[**.***subinterface*] **l2transport** command to enable Layer 2 transport mode on a HundredGigE interface or subinterface and enter the interface or subinterface configuration mode.

   Example

   ```
   Router(config)#interface HundredGigE 0/1/0/0.1 l2transport
   ```

   The *interface-path-id* is specified as the *rack/slot/module/port* location of the interface and **.***subinterface* is the optional subinterface number.

3. (Optional) Run the **encapsulation dot1q** *vlan-id* [**exact**]or**encapsulation dot1ad** *vlan-id* **dot1q** *vlan-id* command to configure the IEEE 802.1Q encapsulation on the specified VLAN only.

   Example

   ```
   Router(config-if)#encapsulation dot1q 1 exact
   ```

4. (Required if VLAN tagging configured) Specify the one or two tags (depending on the network configuration) that should be removed from frames arriving at the ingress interface to the bridge domain.

   Example

   ```
   Router(config-if)#rewrite ingress tag pop 1 symmetric
   ```

> **Note**
>
> When you configure double tags using dot1ad and dot1q encapsulation, you must use the **rewrite ingress tag pop 2 symmetric** command.

5. Run the **end** or **commit** commands to save the configuration changes.

Example

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

or

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

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

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

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

Configure a bridge group and assign interfaces to a bridge domain

This topic describes how to configure an L2VPN bridge group and bridge domain and associate a HundredGigE interface with the bridge domain so that the interfaces can be routed by a Bridge-Group Virtual Interface (BVI).

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

Procedure

1. In the global configuration mode, run the **l2vpn** command to enter the L2VPN configuration mode.

Example

```
Router(config)#l2vpn
```

2. Run the **bridge group** *bridge-group-name* command to create a bridge group and enter the L2VPN bridge group configuration mode.

   Example

   ```
   Router(config-l2vpn)#bridge group 10
   ```

3. Run the **bridge-domain** *bridge-domain-name* command to create a bridge domain and enter L2VPN bridge group bridge domain configuration mode.

   Example

   ```
   Router(config-l2vpn-bg)#bridge-domain BD_1
   ```

4. Associate a HundredGigE interface with the specified bridge domain and enter the L2VPN bridge group bridge domain attachment circuit configuration mode.

   Example

   ```
   Router(config-l2vpn-bg-bd)#interface HundredGigE 0/1/0/0.1
   ```

   The *interface-path-id* is specified as the *rack/slot/module/port* location of the interface and **.***subinterface* is the optional subinterface number.

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

5. Run the **end** or **commit** commands to save the configuration changes.

   Example

   ```
   Router(config-l2vpn-bg-bd-ac)#end
   ```

   or

   ```
   Router(config-l2vpn-bg-bd-ac)#commit
   ```

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

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

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

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

Associate BVI as a routed Interface on a bridge domain

This topic describes how to associate a Bridge-Group Virtual Interface (BVI) as the routed interface on a bridge domain by creating the bridge group and bridge domain and applying the routed interface BVI configuration.

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

Procedure

1. In the global configuration mode, run the **l2vpn** command to enter the L2VPN configuration mode.

   Example

   ```
   Router(config)#l2vpn
   ```

2. Create a bridge group and enter the L2VPN bridge group configuration mode.

   Example

   ```
   Router(config-l2vpn)#bridge group BG_test
   ```

3. Run the **bridge-domain** *bridge-domain-name* command to create a bridge domain and enter the L2VPN bridge group bridge domain configuration mode.

   Example

   ```
   Router(config-l2vpn-bg)#bridge-domain 1
   ```

4. Associate the specified BVI as the routed interface for the interfaces that are assigned to the bridge domain.

   Example

   ```
   Router(config-l2vpn-bg-bd)# routed interface bvi 1
   ```

5. Run the **end** or **commit** commands to save the configuration.

   Example

   ```
   Router(config-l2vpn-bg-bd)#end
   ```

   or

   ```
   Router(config-l2vpn-bg-bd)#commit
   ```

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

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

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

Display information about BVI

This topic lists the show commands you can use to display information about a Bridge-Group Virtual Interface (BVI).

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

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

IRB configuration examples

This topic covers various configuration examples for IRB.

Basic IRB configuration

This topic shows an example of the most basic Integrated Routing and Bridging (IRB) configuration, including creating a BVI with an IPv4 address, configuring a Layer 2 AC interface, and associating the BVI as the routed interface for the bridge domain.

This example shows how to perform the most basic IRB configuration:

```
! Configure the BVI and its IPv4 address
!
Router#configure
Router(config)#interface bvi 1
Router(config-if))#ipv4 address 10.10.0.4 255.255.255.0
Router(config-if))#exit
```

```
!
! Configure the Layer 2 AC interface
!
Router(config)#interface HundregGigE 0/1/0/0 l2transport
Router(config-if))#exit
!
! Configure the L2VPN bridge group and bridge domain and assign interfaces
!
Router(config)#l2vpn
Router(config-l2vpn)#bridge group 10
Router(config-l2vpn-bg)#bridge-domain 1
Router(config-l2vpn-bg-bd)#interface HundregGigE 0/1/0/0
Router(config-l2vpn-bg-bd-if)#exit
!
! Associate a BVI to the bridge domain
!
Router(config-l2vpn-bg-bd)#routed interface bvi 1
Router(config-l2vpn-bg-bd)#commit
```

IRB using ACs with VLANs

This topic shows an example of configuring Integrated Routing and Bridging (IRB) on a bridge domain with Layer 2 attachment circuits (ACs) that use 802.1Q-encapsulated VLANs on HundredGigE subinterfaces.

This example shows how to configure IRB on a bridge domain with Layer 2 ACs using 802.1Q-encapsulated VLANs:

```
! Configure the BVI and its IPv4 address
!
Router#configure
Router(config)#interface bvi 1
Router(config-if))#ipv4 address 10.10.0.4 255.255.255.0
Router(config-if))#exit
!
! Configure the Layer 2 AC interfaces using dot1q encapsulation on a VLAN
!
Router(config)#interface HundredGigE 0/1/0/0.1 l2transport
Router(config-if))#no ip address
Router(config-if))#encapsulation dot1q 1 exact
Router(config-if))#rewrite ingress tag pop 1 symmetric
Router(config-if))#exit
Router(config)#interface HundredGigE 0/1/0/1.1 l2transport
Router(config-if))#no ip address
Router(config-if))#encapsulation dot1q 1 exact
Router(config-if))#rewrite ingress tag pop 1 symmetric
Router(config-if))#exit
!
! Configure the L2VPN bridge group and bridge domain and assign interfaces
!

Router(config)#l2vpn
Router(config-l2vpn)#bridge group 10
Router(config-l2vpn-bg)#bridge-domain 1
Router(config-l2vpn-bg-bd)#interface HundredGigE 0/1/0/0.1
Router(config-l2vpn-bg-bd)#interface HundredGigE 0/1/0/1.1
Router(config-l2vpn-bg-bd-if)#exit
!
! Associate a BVI to the bridge domain
!
```

```
Router(config-l2vpn-bg-bd)#routed interface bvi 1
Router(config-l2vpn-bg-bd)#commit
```

IPv4 addressing on a BVI supporting multiple IP networks

This topic shows an example of configuring secondary IPv4 addresses on a Bridge-Group Virtual Interface (BVI) that supports bridge domains for multiple IP networks, with the BVI having an address on each of the bridge domain networks.

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

```
Router#configure
Router(config)#interface bvi 1
Router(config-if))#ipv4 address 10.10.10.4 255.255.255.0
Router(config-if))#ipv4 address 10.20.20.4 255.255.255.0 secondary
Router(config-if))#ipv4 address 10.30.30.4 255.255.255.0 secondary
Router(config-if))#commit
```

Comprehensive IRB configuration with BVI bundle interfaces and multicast configuration

This topic shows a comprehensive router configuration for Integrated Routing and Bridging (IRB) that includes BVI bundle interfaces, dot1q-encapsulated Layer 2 subinterfaces, OSPF routing, and IGMP snooping with multicast routing support.

This example shows a more comprehensive router configuration with IRB and BVI multicast support:

```
interface Bundle-Ether25
 ipv4 address 10.21.0.2 255.255.255.0
!
interface Loopback0
 ipv4 address 10.5.5.5 255.255.255.255
!
interface HundredGigE0/0/0/1
 negotiation auto
!
interface HundredGigE0/0/0/1.1 l2transport
 encapsulation dot1q 1
 rewrite ingress tag pop 1 symmetric
!
interface HundredGigE0/0/0/1.2 l2transport
 encapsulation dot1q 2
 rewrite ingress tag pop 1 symmetric
!

interface HundredGigE0/0/0/9
 bundle id 25 mode active
!
interface HundredGigE0/0/0/19
 bundle id 25 mode active
!
interface HundredGigE0/0/0/29
 bundle id 25 mode active
!

interface HundredGigE0/0/0/39
 bundle id 25 mode active
```

```
interface BVI1
 ipv4 address 10.1.1.1 255.255.255.0
!
interface BVI2
 ipv4 address 10.1.2.1 255.255.255.0

router ospf 100
 router-id 10.5.5.5
 area 0
  interface Bundle-Ether25
    interface Loopback0
    interface BVI1
   interface BVI2
  !
l2vpn
 bridge group IRB
  bridge-domain IRB1
   igmp snooping profile IRB_SNOOP
   interface HundredGigE0/0/0/1.1
   !
   routed interface BVI1
  !
  bridge-domain IRB2
   igmp snooping profile IRB_SNOOP
   interface HundredGigE0/0/0/1.2
   !
   routed interface BVI2

multicast-routing
 address-family ipv4
  interface all enable
igmp snooping profile IRB_SNOOP
 report-suppression disable
!
router pim
 address-family ipv4
  rp-address 10.10.10.10
```

IRB with BVI and VRRP configuration

This topic shows a partial router configuration for Integrated Routing and Bridging (IRB) support of a Bridge-Group Virtual Interface (BVI) and Virtual Router Redundancy Protocol (VRRP), including the L2VPN bridge domain and VRRP group configuration on the BVI.

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

> **Note**
>
> VRRPv6 is also supported.

```
l2vpn
 bridge group IRB
  bridge-domain IRB-EDGE
   interface HundredGigE0/0/0/8
!
```

```
   routed interface BVI 100
!
interface HundredGigE0/0/0/8
  l2transport
!
interface BVI 100
 ipv4 address 10.21.1.1 255.255.255.0
!
router vrrp
 interface BVI 100
  vrrp 1 ipv4 10.21.1.100
  vrrp 1 priority 100
 !
```

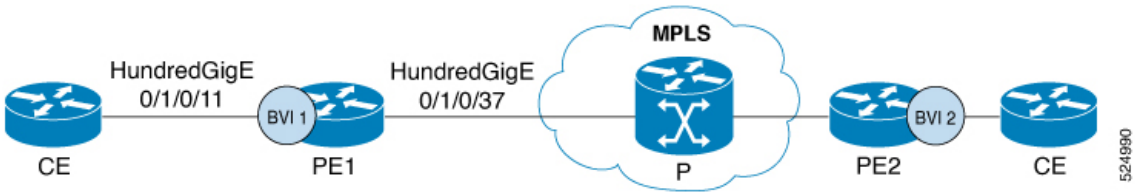
6PE or 6VPE with BVI configuration

This topic shows an example of configuring an MPLS 6PE or 6VPE environment using Bridge-Group Virtual Interfaces (BVIs) at the customer-edge-facing sides of Cisco 8000 Series Routers acting as the provider-edge devices, with per-VRF label allocation.

This example shows how to configure an MPLS 6PE/6VPE environment using BVIs at the CE-facing sides of the Cisco 8000 Series Router as the PE devices.

This figure shows the location of the BVI interfaces (green icons) on the Cisco 8000 Series Routers as the PE1 and PE2 devices.

Figure 12: BVI interfaces on the CE-facing sides in an MPLS 6PE or 6VPE network



This is a sample configuration only for the Cisco 8000 Series Router (PE1) device with a BVI interface numbered 1 on the CE-facing side, and a non-BVI interface (HundredGigE 0/1/0/37) on the core-facing side. A similar configuration would apply to the PE2 device:

```
! Be sure to configure IPv6 unicast address families
!
vrf 1
address-family ipv6 unicast
  import route-target
   100:2
  export route-target
   100:2

interface Loopback0
 ipv4 address 10.11.11.11/32
!
! Configure the BVI interface to participate in the VRF
! and with an IPv6 address.
!
interface BVI1
 vrf 1
 ipv6 address 2001:DB8:1/32
!
! Assign the Ethernet CE-facing interface to the
```

```
! L2VPn bridge domain where the routed BVI interface is also associated.
!
l2vpn
 bridge group 1
  bridge-domain 1
   interface HundredGigE 0/1/0/11
 routed interface BVI1
!
! Configure OSPF routing for the BVI interface for
! advertisement of its IPv6 address.
!
router ospfv3 1
 graceful-restart
 redistribute bgp 1
 area 1
  interface BVI1
  interface Loopback0
!
! Configure BGP routing and be sure to specify the
! IPv6 unicast address family.
! Note that the per-VRF label allocation mode is required
! and is the only supported label allocation mode.
!
router bgp 1
 bgp router-id 10.11.11.11
 bgp redistribute-internal
 bgp graceful-restart

 address-family ipv6 unicast
  redistribute ospfv3 1 match internal external
  label-allocation-mode per-vrf
  allocate-label all
 !
 address-family vpnv6 unicast
 !
 neighbor 10.11.12.12
  remote-as 1
  update-source Loopback0
  address-family ipv6 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
  address-family ipv6 labeled-unicast
  !
  address-family vpnv6 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 vrf 1
  rd 100:2
 label-allocation-mode per-vrf
  address-family ipv6 unicast
   redistribute connected

mpls ldp
 router-id 10.11.11.11
 graceful-restart
 interface HundredGigE 0/1/0/37
```

13 Configure IP event dampening

Topics:

This chapter describes the IP Event Dampening feature, interface state change events and thresholds, affected components and supported protocols, guidelines and limitations, and how to enable and verify IP event dampening on the Cisco 8000 Series Router.

IP event dampening

This topic describes how the IP Event Dampening feature uses a configurable exponential decay mechanism to suppress the effects of excessive interface flapping on routing protocols and routing tables, so that the network stays stable and converges faster.

The IP Event Dampening is a networking feature that

- suppresses the effects of excessive interface flapping on routing protocols
- automatically identifies flapping interfaces, and
- isolates unstable interfaces to improve network stability.

Interface state changes occur when operators administratively bring interfaces up or down or when an interface changes state. These changes notify routing protocols of affected routes. Every state change forces devices to recalculate best paths, update routing tables, and advertise valid routes to peers. Excessive flapping causes devices to consume significant processing resources and results in routing protocol synchronization issues. By dampening an interface, the system removes the unstable interface from the network until it stabilizes.

Table 25: Feature History Table

Feature Name	Release	Description
IP event dampening	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
IP event dampening	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: - 8011-32Y8L2H2FH - 8011-12G12X4Y-A/D
IP event dampening	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release	Description
IP event dampening	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) You can now enhance network stability by suppressing unnecessary flapping through IP Event Dampening. This feature prevents interfaces from repeatedly transitioning between up and down states due to intermittent issues, reducing operational disruptions. By configuring dampening parameters, network operators can control the penalty for each flap and set thresholds for interface recovery, thus ensuring smoother network performance and reliability. This proactive approach minimizes unnecessary alerts and interventions, allowing for focused attention on more critical network events, ultimately improving overall network uptime and efficiency. *Previously this feature was supported on Q200 and Q100. It is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM

Benefits of IP event dampening

The deployment of IP event dampening offers these advantages for network performance:

- It isolates failed interfaces to prevent the propagation of network disturbances.
- It reduces the utilization of system processing resources by neighboring network devices.
- It improves overall network stability and convergence times.

Interface state change events

This topic describes how the IP Event Dampening feature handles interface state change events by using a configurable exponential decay mechanism to identify flapping interfaces, assign penalties, suppress them if necessary, and make them available again when they stabilize.

Interface state change state is a IP event dampening mechanism that

- employs a configurable exponential decay mechanism
- suppresses the effects of excessive interface flapping, and
- filters excessive route updates for routing protocols.

The feature manages interface stability through several operational steps:

- The system identifies flapping interfaces.
- The feature assigns penalties to these interfaces.
- The mechanism suppresses the interface if the penalty exceeds a specific threshold.
- The system makes the interface available to the network once it stabilizes.

Suppress threshold

This topic describes the suppress threshold, which is the accumulated penalty value that triggers the router to place a flapping interface in the dampened state when the penalty reaches the default or preconfigured suppress threshold.

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

Half-life period

This topic describes the half-life period, which determines how fast the accumulated penalty on a dampened interface decays exponentially after the interface stabilizes and stops flapping, and lists the configurable range and default value.

- determines the rate at which accumulated penalties decay exponentially
- monitors interface state changes during dampening, and
- reduces penalties until they reach the reuse threshold.

The system manages interface dampening through these processes:

- The router monitors the interface for additional state changes while the interface remains in a dampened state.
- The system reduces the penalty by half after each half-life period expires if the interface stabilizes.
- The interface remains in a dampened state as long as the accumulated penalty exceeds the suppress threshold.
- The half-life period timer supports a configurable range from 1 to 45 minutes with a default value of 1 minute.

Reuse threshold

This topic describes the reuse threshold, which is the penalty value at which a dampened route is unsuppressed and made available again to other devices in the network, and lists the configurable range and default value.

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

Maximum suppress time

This topic describes the maximum suppress time, which is the longest time an interface can remain dampened when a penalty is assigned, and describes how the maximum accumulated penalty is derived from this value, the reuse threshold, and the half-life period.

The maximum suppress time represents the maximum time an interface can remain dampened when a penalty is assigned to an interface.

These are the guidelines for maximum suppress time:

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

Affected components during IP event dampening

This topic describes how the IP Event Dampening feature affects routing protocol behavior only when an interface is suppressed, and states that unconfigured or unsuppressed interfaces continue to behave normally during state transitions.

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

Route types

This topic describes how connected and static routes on an interface are affected by IP event dampening, including how the routes are removed from the routing table when the interface is dampened and reinstalled when the interface is unsuppressed.

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

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

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

Supported protocols

This topic lists the routing protocols that are impacted by the IP Event Dampening feature, including BGP, EIGRP, HSRP, OSPF, RIP, and VRRP, and states the ping and SSH behavior on a dampened interface IP address.

All the protocols that are used are impacted by the IP Event Dampening feature. The IP Event Dampening feature supports:

- Border Gateway Protocol (BGP)
- Enhanced Interior Gateway Routing Protocol (EIGRP)
- Hot Standby Routing Protocol (HSRP)
- Open Shortest Path First (OSPF)
- Routing Information Protocol (RIP)
- VRRP

These restrictions apply:

- Ping and SSH to the concerned interface IP address does not work.
- The IP Event Dampening feature has no effect on any routing protocols if it is not enabled or an interface is not dampened.

Guidelines and limitations for IP event dampening

This topic describes the guidelines and limitations to configure IP event dampening, including how penalties accumulate and decay, how the interface transitions between the dampened and unsuppressed states, and the supported interfaces and IP client behavior.

Penalty accumulation and decay

Consider the following behavior for how penalties are assigned and reduced when IP event dampening is enabled on an interface:

- When dampening is enabled, a penalty value is assigned to an interface. This value starts at 0 and increases by 1000 each time the interface state transitions from up to down.
- For each flap of the interface, a certain penalty is added. The penalty decays exponentially when parameters are configured.

Dampening state transitions

Consider the following behavior for how the interface transitions between the dampened and unsuppressed states:

- When the penalty exceeds a certain high level, the interface is dampened. It is unsuppressed when the penalty decays below a low level.

- When an interface is dampened, the IP address and the static routes are removed from the interface. All the clients of IP get an IP delete notification.
- When an interface is unsuppressed, the IP address and the relevant routes are added back. All the clients of IP get an IP address add notification for all the IP addresses of the interface.

Applicability and IP client impact

Consider the following applicability guidelines when configuring IP event dampening:

- All Layer 3 interfaces that are configured on the Ethernet interface, port changes, and SVI support this feature.
- Due to changes in the netstack-IP component, all IP clients observe the impact of interface dampening.

Enable IP event dampening

This topic describes how to enable the IP Event Dampening feature on an interface by using the **dampening** command in interface configuration mode, and how to verify the configuration by using the **show** commands.

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

Procedure

1. Run the **configure terminal** command to enter the global configuration mode.

   Example

   ```
   Router#configure terminal
   ```

2. Run the **interface** *type number* command to enter interface configuration mode and configure the specified interface.

   Example

   ```
   Router(config)#interface HundredGigE 0/0/0/0
   ```

3. Run the **dampening** [*half-life-period reuse-threshold*] [*suppress-threshold max-suppress* [*restart-penalty*]] command to enable interface dampening.

   Example

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

   - Entering the **dampening** command without any arguments enables interface dampening with default configuration parameters.
   - When manually configuring the timer for the *restart-penalty* argument, the values must be manually entered for all arguments.

4. Run the **end** command to exit interface configuration mode.

Example

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

5. Run the **show dampening interface** or **show interface dampening** commands to verify the configuration of the IP Event Dampening feature.

Example

```
Router#show dampening interface
```

or

```
Router#show interface dampening
```

- **show dampening interface**–Displays dampened interfaces.
- **show interface dampening**–Displays dampened interfaces on the local router.

14 Link bundling

Topics:

This chapter provides high bandwidth and link redundancy for Cisco 8000 series routers running IOS XR software.

Link bundling

This topic describes how link bundling groups multiple point-to-point links into a single logical interface to provide higher bandwidth, redundancy, and load balancing between two routers.

Link bundling is a network configuration that

- aggregates multiple ports into a single logical interface
- utilizes a shared MAC and IP address, and
- supports a unified configuration set.

Table 26: Feature History Table

Feature Name	Release	Description
Configure link bundling	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Configure link bundling	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is now supported on: - 8711-48Z-M - 8011-32Y8L2H2FH - 8011-12G12X4Y-A/D
Configure link bundling	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release	Description
Configure link bundling	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) You can now enhance network resilience and efficiency by aggregating multiple physical interfaces into a single logical link. This bundling improves redundancy by ensuring continued data transmission even if individual links fail. It supports load balancing across bundled links, optimizing data flow and minimizing congestion. The feature is easily configurable and integrates seamlessly with existing network setups, providing a scalable solution for managing increased traffic demands. By utilizing link bundling, you achieve a more robust and flexible network infrastructure, promoting continuous connectivity and performance. *Previously this feature was supported on Q200 and Q100. It is now extended to: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM
1023 Ethernet Bundle Interfaces Support	Release 7.3.15	With the introduction of this enhancement, the maximum system-wide bundle interface scale has increased from 512 to 1023 bundle interfaces. The default value remains at 64-member links for each bundle.
64-bit Bandwidth Support	Release 7.3.15	With this release, the Cisco 8000 Series Router supports 64-bit bandwidth field, as opposed to the previous 32-bit bandwidth field. 64-bit bandwidth enables the system to support interface bandwidths greater than 4.2T.

A link bundle is a group of one or more ports that the system aggregates and treats as a single link.

Each bundle has

- a single MAC address
- a single IP address, and
- a single configuration set (such as ACLs).

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

Operational specifications for link bundling

These are the operational specifications for configuring link bundling:

- The router supports both Layer 2 and Layer 3 link bundles. If the link bundle is a Layer 3 interface, the system requires an IP address. If the link bundle is a Layer 2 interface, the system does not require an IP address. A link bundle on the router may contain Layer 2 and Layer 3 subinterfaces within it. In that case, the Layer 3 subinterfaces require IP addresses, but the link bundle interface does not require an IP address.
- The router supports bundling for Ethernet interfaces.
- All the individual links within a single bundle must be of the same type.
- Cisco IOS XR software supports IEEE 802.3ad– A standard technology that employs a Link Aggregation Control Protocol (LACP) to ensure that all the member links in a bundle are compatible. The system automatically removes the links from a bundle that are incompatible or have failed.
- Prior to Cisco IOS XR Software Release 7.3.15, the interface bandwidth was stored and processed as a 32-bit value, which supported bundles with an aggregate bandwidth of up to 4.2 Gbps (the sum of its members). Starting with Cisco IOS XR Software Release 7.3.15, the interface bandwidth is stored and processed as a 64-bit value. The 64-bit value supports significantly larger aggregate bandwidths, accommodating bundles with high-bandwidth members whose combined bandwidth can exceed 4.2 Tbps.

Benefits of link bundling

These are the advantages of using link bundles:

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

IEEE 802.3ad standard

This topic describes how the IEEE 802.3ad standard defines a method of forming Ethernet link bundles by exchanging system, bundle, port, and aggregation-status information between the two systems that host each end of the link bundle.

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

- defines Ethernet link bundle formation
- facilitates information exchange between systems, and
- determines link compatibility via link aggregation group identifiers (LAG IDs).

For each link configured as a bundle member, The systems exchange the following information to form the LAG ID:

- A globally unique local system identifier distinguishes one router from another.
- An operational key identifies the bundle of which the link is a member.
- An identifier , also known as port ID for the link.
- The current aggregation status indicates the state of the link.

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

The system guarantees the uniqueness of the system identifier by using a MAC address from the router. The bundle and link identifiers maintain significance only to the router that assigns them, ensuring that no two links or bundles share the same identifier. The system combines information from the peer system with local system data to determine the compatibility of links configured as bundle members.

The router assigns bundle MAC addresses from a set of reserved addresses in the backplane. This MAC address remains with the bundle for the duration of the interface existence unless the user configures a different address. Member links utilize the bundle MAC address when passing bundle traffic. The system also applies any unicast or multicast addresses configured on the bundle to all member links.

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

Link aggregation through LACP

This topic describes how the Link Aggregation Control Protocol (LACP) communicates between two directly connected systems to verify the compatibility of bundle members and monitor the operational state of link bundles.

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

LACP monitors the operational state of link bundles to ensure that

- all links terminate on the same two systems
- both systems consider the links to be part of the same bundle, and
- all links have the appropriate settings on the peer.

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

LACP fallback

This topic describes how the LACP Fallback feature allows an active LACP interface to establish a Link Aggregation Group (LAG) port-channel before the port-channel receives LACP protocol data units (PDUs) from its peer.

The LACP Fallback is a network feature thatThe LACP Fallback feature allows an active LACP interface to establish a Link Aggregation Group (LAG) port-channel before the port-channel receives the Link Aggregation and Control Protocol (LACP) protocol data units (PDU) from its peer.

- allows an active Link Aggregation and Control Protocol (LACP) interface to establish a LAG port-channel before receiving LACP protocol data units (PDUs)
- enables servers to connect to PXE servers for boot image downloads, and
- maintains one active port until the server completes the boot process.

Benefits of LACP fallback

The LACP Fallback feature provides several operational benefits:

- The router allows the server to bring up the LAG before the system receives any LACP PDUs.
- The server establishes a connection to a PXE server over one Ethernet port to download its boot image.
- The server fully forms an LACP port-channel once the boot process concludes.

LACP short period time intervals

This topic describes how configuring short period time intervals for sending LACP packets enables faster detection and recovery from link failures on bundle member links.

The LACP short period time interval is a configuration setting that

- enables faster detection of link failures
- facilitates rapid recovery from link timeouts, and
- verifies the stability of member links.

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

The system supports specific parameters for short period time intervals:

- Configuration occurs in milliseconds.
- Values increase in increments of 100 milliseconds.
- The range spans from 100 to 1000 milliseconds.
- The default setting equals 1000 milliseconds (1 second).
- The system supports up to the maximum available member links.
- The system processes up to 1280 packets per second (pps).

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

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

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

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

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

Load balancing on link bundles

This topic describes how load balancing distributes traffic over multiple links in a bundle using Layer 2, Layer 3, and Layer 4 routing information.

Load balancing is a forwarding mechanism that

- distributes traffic over multiple links based on certain parameters
- enables bandwidth-based load-balancing for Layer 3 unicast flows, and
- supports all links in a bundle using Layer 2, Layer 3, and Layer 4 routing information.

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

- Per-flow load balancing on non-bundle interfaces using Layer 3 and 4 routing information.
- Pseudowire (PW) load balancing.

Layer 3 egress load balancing on link bundles

This topic describes how Layer 3 egress load balancing selects the same bundle member on both ingress and egress line cards using a common hash value, and provides load distribution for outbound multicast IPv4 and IPv6 traffic.

The Layer 3 egress load balancing mechanism is a network traffic distribution method that

- enables global link bundle load balancing by default
- utilizes consistent hashing between ingress and egress linecards, and
- optimizes packet forwarding across network processors.

The ingress linecard performs bundle member selection and forwards the packet to the linecard and network processor that corresponds to the selected bundle member. The system uses the same hash value for both ingress and egress linecards. Consequently, the egress linecard selects the same bundle member as the ingress linecard.

Multicast IPv4 and IPv6 traffic

The system manages multicast traffic distribution through these processes:

- The system predetermines a set of egress linecards for outbound multicast IPv4 and IPv6 traffic.
- The system selects the bundle member for each outgoing interface based on the multicast group address.
- The egress linecard performs network processor selection when bundle members reside across multiple processors within the egress linecard.
- The system uses a 5-tuple hash to select a bundle member within a network processor when a packet arrives at the egress linecard. This provides better resiliency for bundle member state changes within a network processor.

Link failover

This topic describes how when a member link in a bundle fails, the system redirects traffic to the remaining operational member links and traffic flow remains uninterrupted.

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

Link switchover

This topic describes how you can implement 1:1 link protection for a bundle by designating one active link and one or more dedicated standby links.

The link bundle protection mechanism is a network configuration feature that

- manages active link capacity
- provides redundancy through standby links, and
- facilitates rapid traffic switchover during link failures.

The system provides several configuration options for managing link bundle protection:

- A bundle supports a maximum of 64 active links by default, and the system redirects traffic to remaining operational members if a link fails.
- You can implement 1:1 link protection by setting the **bundle maximum-active links** command to 1, which designates one active link and one or more dedicated standby links.
- The system performs a switchover to a standby link immediately upon the failure of an active link to ensure uninterrupted traffic.
- If the active and standby links are running LACP, you can choose between an IEEE standard-based switchover (the default) or a faster proprietary optimized switchover. If the active and standby links are not running LACP, the proprietary optimized switchover option is used.
- Regardless of the type of switchover you are using, you can disable the wait-while timer, which expedites the state negotiations of the standby link and causes a faster switchover from a failed active link to the standby link. To do so, you can use the **lacp switchover suppress-flaps** command.

Nonstop forwarding during card failover

This topic describes how nonstop forwarding ensures no change in the state of link bundles when a failover occurs between active and standby paired RP cards.

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

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

> **Note**
>
> Failover is always onto the standby RP. You do not need to configure anything to guarantee that the system maintains the standby interface configurations.

QoS and link bundling

This topic describes how when QoS is applied on a bundle for either the ingress or egress direction, QoS is applied at each member interface.

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

VLANs on an Ethernet link bundle

This topic describes how you can configure 802.1Q VLAN subinterfaces on 802.3ad Ethernet link bundles to support Ethernet Flow Points (EFPs) and Layer 3 services.

You can configure 802.1Q VLAN subinterfaces on 802.3ad Ethernet link bundles.

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

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

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

After you create a VLAN on an Ethernet link bundle, the system supports all VLAN subinterface configuration on that link bundle.

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

You can configure Layer 3 VLAN subinterfaces as follows:

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

Designate a member link as unviable

This topic describes how you can mark a member link as unviable to introduce a delay during which the link is treated as standby, so that the link configuration is fully established before it is used for data transmission.

The link bundle is a network configuration that

- aggregates multiple physical links into a single logical link, negotiates link aggregation through the Link Aggregation Control Protocol (LACP), and
- manages the availability of member links for data transmission.

When a member link is added to a link bundle, Link Aggregation Control Protocol (LACP) communication is established with the peer to negotiate and control the link aggregation. LACP does not have any provision to incorporate a delay before letting data transmission over the link. Therefore, the peer starts using the link when the LACP communication is up. Occasionally, even though the link status is up and LACP communication is up, the hardware programming for data-path packet forwarding does not complete. In such scenarios, the transmitted data is lost without any notification or error message to the source or destination of the traffic.

Table 27: Feature History Table

Feature Name	Release	Description
Designate a member link as unviable	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Designate a member link as unviable	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Designate a member link as unviable	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM

Feature Name	Release	Description
Designate a member link as unviable	Release 7.10.1	Earlier, when a member link is added to an interface link bundle, the peer starts using the link as soon as the LACP communication is up. Sometimes, the hardware programming for the data-path does not get complete in this time resulting in packet loss without any notification to the source. You can now mark a member link as unviable to introduce a delay during which the link is treated as standby. By delaying the usage of the member link for data transmission, you can ensure that the link configuration is fully established, which enables successful data transmission. This feature introduces these changes: • CLI: **forwarding-unviable** • YANG Data Model: New XPath for `Cisco-IOS-XR-bundlemgr-oper` (see Github, YANG Data Models Navigator).

You can manage the readiness of member links using these commands:

- You can delay the use of such member links, which are not fully ready to handle data transmission, using the **forwarding-unviable** command. This command configures the link as forwarding-unviable and the member link is considered "standby" for bundle management. As standby member links of a bundle are not used for data transmission, the usage of forwarding-unviable member links is delayed.
- When the member link is fully up—that is, the packet forwarding data-path is also completely programmed—you can disable forwarding-unviability of the link using the **no forwarding-unviable** command. This removes the forwarding-unviable configuration of the link. The link is then treated as an active member of the bundle and is used in data transmission and load balancing.

Note

We recommend that you wait for a few minutes before running the **no forwarding-unviable** command to ensure that the packet forwarding data-path is completely programmed.

Guidelines and restrictions for designating member links as unviable

This topic describes the guidelines and restrictions when you designate a bundle member link as unviable using the forwarding-unviable configuration.

Consider these guidelines and restrictions while designating a member link as unviable:

- Forwarding-unviable is disabled on all Ethernet interfaces by default. Therefore, by default, all member links in a bundle are considered "active".
- A link bundle is considered up only if at least one member link is active. Only the active member links in the link bundle are used for data transmission, load balancing, and redundancy.

- If a link bundle has only one member link, which is forwarding-unviable, the bundle state is considered "down".
- If all the member links in a bundle are forwarding-unviable, the bundle state is considered "down".
- Other existing threshold parameters such as minimum-active links, maximum-active links, and maximum-active bandwidth, which are considered to determine the bundle state, continue to function along with the forwarding-unviable functionality.
- The forwarding-unviable configuration has no effect on individual Ethernet interfaces that are not part of a link bundle. Irrespective of the configuration, such non-member interfaces continue to attempt data transmission and reception.

Link flap err-disables

This topic describes how the link-flap err-disable feature prevents excessive link flapping by automatically disabling an interface that frequently toggles between "up" and "down" states, and provides manual or timer-based recovery.

A link-flap err-disable is a network reliability feature that:

- Automatically shuts down an interface if it frequently toggles between "up" and "down" states.
- Ensures network reliability by allowing traffic to reroute to stable backup paths when a port becomes unstable.
- Is optimized for short-distance links within data centers, such as those connecting servers and Top-of-Rack (ToR) switches.

Table 28: Feature History Table

Feature Name	Release	Description
Link flap err-disables	Release 26.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: K100)] You can achieve reliable network performance and a stable system state by isolating unstable interfaces that flap excessively, preventing routing protocol disruptions and ensuring effective workload failover. With this feature, the system monitors link transitions and automatically moves physical, breakout, or bundle member interfaces into an error-disabled state if a defined flap-count threshold is breached within a specific time window. This automated mechanism guarantees traffic redirection to stable paths while providing the flexibility of manual restoration or a timer-based automatic recovery once the link stabilizes. This feature introduces these changes: CLI: • **fault management link-flap error-disable** • **error-disable recovery cause** • **clear error-disable** • **show error-disable**

Supported interface types:

- Physical interfaces
- Breakout interfaces
- Bundle member interfaces

Syslog messages

When an interface is disabled due to excessive flapping, the system generates this syslog message:

```
%PKT_INFRA-ERRDIS-6-ERROR_DISABLE : GigabitEthernet0/0/0/0: Error disabling
due to reason: link-flap-exceeded
```

Limitations for configuring link-flap err-disable

This topic describes the limitations when you configure link-flap err-disable on the Cisco 8000 Series Router, including which link state changes count toward the flap threshold and how the feature interacts with interface dampening, carrier-delay, and object-tracking.

Consider these limitations while configuring link-flap err-disable:

- Only hardware-level link state changes are counted. Upper-layer protocol changes, such as Layer 2 protocols or line protocols, do not trigger a flap count.
- The link-flap err-disable feature works independently of the interface dampening feature. If both are configured, the first feature to reach its threshold takes action.
- To ensure optimal coexistence, review interactions with carrier-delay and object-tracking, as these may impact feature functionality.

Configure link bundling

This topic describes how link bundle configuration involves creating a link bundle, assigning an IP address to the virtual interface, and adding member interfaces to the bundle.

This procedure provides a general overview of the link bundle configuration process. Ensure that you clear all previous network layer configuration before adding it to a bundle:

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

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

Prerequisites to configure link bundling

This topic describes the prerequisites before you configure link bundling on the Cisco 8000 Series Router.

Before configuring link bundling, ensure that you meet the following tasks and conditions:

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

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

Configuration guidelines and restrictions for Ethernet link bundles

This topic covers the configuration guidelines and restrictions that apply whne you confiigure Ethernet link bundles.

These guidelines and restrictions apply when you configure Ethernet link bundles:

Mixed-speed bundle support

Starting with Cisco IOS XR Release 7.2.1, the router supports mixed-speed bundles, allowing member links with different bandwidths to be included in the same bundle. Traffic distribution across bundle members is based on the bandwidth of each link. Mixed-speed bundles are subject to a maximum bandwidth ratio of 10:1 between the fastest and slowest member links.

For example, you can combine a 10 Gbps and a 100 Gbps link, or a 100 Gbps and a 400 Gbps link, in the same bundle; however, a 10 Gbps and a 400 Gbps link cannot be bundled together. Load balancing is performed in proportion to the bandwidth of each member link. Typical valid combinations include:

- 400G, 100G
- 400G, 40G
- 400G, 100G, 40G
- 100G, 40G
- 100G, 10G
- 100G, 40G, 10G
- 40G and 10G

Bundle weight constraints

- Additionally, the total weight of the bundle must not exceed 64.
-
- The weight of each bundle member is the ratio of its bandwidth to the lowest-bandwidth member. The total weight of the bundle is the sum of the weights or relative bandwidth of each bundle member. Because the weight for a bundle member is greater than or equal to 1 and less than or equal to 10, the total number of links in a bundle is less than 64 in the mixed-bundle case.
-

Bundle scale and interface support

Bundle scale and interface support properties apply to Ethernet link bundles on the router.

- Any type of Ethernet interface can be bundled, with or without the use of Link Aggregation Control Protocol (LACP).

- With Cisco IOS XR Release 7.3.15, a single router can support up to 1023 bundle interfaces.

  Each bundle can accommodate up to 64 member links.

  If adding a new line card causes these limits to be exceeded, the system will experience continuous Out of Resource (OOR) failures. To resolve these errors, you must either reduce the scale or disable the affected line card.
- Physical-layer and link-layer configuration is performed on individual member links of a bundle.
- Configuration of network-layer protocols and higher-layer applications is performed on the bundle itself.
- IPv4 and IPv6 addressing is supported on Ethernet link bundles.
- Only physical links can be bundle members.

Bundle operational characteristics

Operational characteristics apply to bundles and their member links.

- A bundle can be administratively enabled or disabled.
- Each individual link within a bundle can be administratively enabled or disabled.
- Ethernet link bundles are created in the same way as Ethernet channels, where the user enters the same configuration on both end systems.
- The MAC address that is set on the bundle becomes the MAC address of the links within that bundle.
- Load balancing (the distribution of data between member links) is done by flow instead of by packet. Data is distributed to a link in proportion to the bandwidth of the link in relation to its bundle.
- QoS is supported and is applied proportionally on each bundle member.
- All links within a single bundle must terminate on the same two systems.
- Bundled interfaces are point-to-point.
- A link must be in the up state before it can be in the distributing state in a bundle.

LACP timer configuration on 8404-SYS-D

The minimum supported LACP timer is 1 second. When this timer is configured, it is preferred to use the **redundancy switchover** command rather than the **reload** command to trigger an RPFO.

The **redundancy switchover** command ensures operational continuity by preventing the LACP timeouts and bundle disconnections that typically occur during a reload operation.

Configure Ethernet link bundles

This topic describes how to configure an Ethernet link bundle by creating the Bundle-Ether virtual interface, assigning an IP address, and setting optional bundle parameters such as minimum-active bandwidth, minimum-active links, 1:1 link protection, and LACP fast-switchover.

Aggregate multiple physical Ethernet interfaces into a single logical link bundle to provide higher bandwidth, redundancy, and load balancing between two routers.

For an Ethernet bundle to be active, you must perform the same configuration on both connection endpoints of the bundle.

Note

When you use either physical Ethernet subinterfaces or bundle Ethernet subinterfaces, the device adds broadcast packets exiting a subinterface to the output counters of the main interface. The output counters of the subinterface do not increase. For example, when you send 100 broadcast packets through an egress physical or bundle subinterface, the main interface output counters increase by 100, and the output broadcast packet counter of the subinterface remains at zero.

Tip

You can programmatically perform the configuration using `openconfig-lacp.yang`, `openconfig-if-aggregate.yang` OpenConfig data models, `Cisco-IOS-XR-bundlemgr-oper.yang` Cisco IOS XR native data model, or `Cisco-IOS-XR-um-lacp-cfg.yang` Unified data model.

Procedure

1. Enter global configuration mode using the **configure** command.

   Example

   ```
   Router# configure
   ```

2. Create a new Ethernet link bundle with the specified *bundle-id* in the range 1 to 65535, and enter interface configuration submode.

   Example

   ```
   Router(config)# interface Bundle-Ether 3
   ```

3. Assign an IP address and subnet mask to the virtual interface.

   Example

   ```
   Router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
   ```

   Only a Layer 3 bundle interface requires an IP address.

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

   Example

   ```
   Router(config-if)# bundle minimum-active bandwidth 580000
   ```

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

Example

```
Router(config-if)# bundle minimum-active links 2
```

6. (Optional) Implement 1:1 link protection for the bundle.

   Example

   ```
   Router(config-if)# bundle maximum-active links 1 hot-standby
   ```

   This causes the highest-priority link in the bundle to become active and the second-highest-priority link to become the standby. It also specifies that a switchover between active and standby LACP-enabled links is implemented per a proprietary optimization.

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

7. (Optional) If you enabled 1:1 link protection on a bundle with member links running LACP, disable the wait-while timer in the LACP state machine.

   Example

   ```
   Router(config-if)# lacp fast-switchover
   ```

   Disabling this timer causes a bundle member link in standby mode to expedite its normal state negotiations, thereby enabling a faster switchover from a failed active link to the standby link.

8. Exit interface configuration submode.

   Example

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

What to do next

Perform the Add Ethernet interface to bundle sub-task next to add member Ethernet interfaces to the bundle and commit the configuration on both connection endpoints.

Add Ethernet interface to bundle

This topic describes how to add a member Ethernet interface to an existing Bundle-Ether link bundle, set optional per-member parameters, commit the configuration on both connection endpoints, and verify that the bundle is operational.

Perform this task after you create the Bundle-Ether interface and set the required bundle parameters. Repeat this task on the remote end of the connection so that both endpoints share the same bundle configuration.

Procedure

---

1. Enter interface configuration mode for the Ethernet interface to add to the bundle.

Example

```
Router(config)# interface GigabitEthernet 1/0/0/0
```

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

2. Add the link to the specified bundle using the **bundle id** *bundle-id* [**mode** {**active** | **on** | **passive**}] command.

   Example

   ```
   Router(config-if)# bundle id 3
   ```

   To enable active or passive LACP on the bundle, include the optional **mode active** or **mode passive** keywords in the command string. To add the link to the bundle without LACP support, include the optional **mode on** keywords.

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

3. (Optional) If you set the **bundle maximum-active links** command to 1, set the priority of the active link to the highest priority (lowest value) and the standby link to the second-highest priority.

   Example

   ```
   Router(config-if)# bundle port-priority 1
   ```

4. (Optional) If a link is in the down state, bring it up.

   Example

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

   The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

5. (Optional) Repeat the interface addition steps to add more member links to the bundle.

   Example

   ```
   Router(config-if)# exit
   Router(config)# interface GigabitEthernet 1/0/2/1
   Router(config-if)# bundle id 3
   Router(config-if)# bundle port-priority 2
   Router(config-if)# no shutdown
   ```

6. Save the configuration changes using the **end** or **commit** command.

   Example

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

7. Perform all the previous steps on the remote end of the connection to bring up the other end of the link bundle.
8. (Optional) Verify the specified Ethernet link bundle.

Example

```
Router# show bundle Bundle-Ether 3
Router# show lacp bundle Bundle-Ether 3
```

Confirm that the bundle is operational, that all expected member links appear, and that LACP is operational on the ports and their peers.

Configure VLAN bundles

This topic describes how to configure a VLAN bundle by creating an Ethernet bundle, creating VLAN subinterfaces and assigning them to the Ethernet bundle, and assigning Ethernet links to the Ethernet bundle.

Configure 802.1Q VLAN subinterfaces on an Ethernet link bundle to support Layer 3 services on the bundle.

The creation of a VLAN bundle involves three main tasks:

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

> **Note**
>
> For a VLAN bundle to be active, you must perform the same configuration on both ends of the bundle connection.

Procedure

Enter global configuration mode using the **configure** command.

Example

```
Router# configure
```

What to do next

Perform the following sub-tasks in order to complete the VLAN bundle configuration:

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

Create an Ethernet bundle

This topic describes how to create an Ethernet link bundle by adding a Bundle-Ether virtual interface.

Create an Ethernet link bundle to serve as the parent interface for the VLAN subinterfaces that you configure in the next sub-task.

Perform this sub-task from global configuration mode.

Procedure

1. Create a new Ethernet link bundle using the **interface Bundle-Ether** *bundle-id* command.

   Example

   ```
   Router(config)# interface Bundle-Ether 3
   ```

2. Assign an IP address and subnet mask to the virtual interface.

   Example

   ```
   Router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
   ```

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

   Example

   ```
   Router(config-if)# bundle minimum-active links 2
   ```

4. Exit interface configuration submode.

   Example

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

What to do next

Perform the Create VLAN subinterfaces and assign them to the Ethernet bundle sub-task next.

Create VLAN subinterfaces and assign them to the Ethernet bundle

This topic describes how to create 802.1Q VLAN subinterfaces on an existing Ethernet link bundle, set the Layer 2 encapsulation, assign IP addresses to the subinterfaces, and bring the subinterfaces up as the second stage in configuring a VLAN bundle.

Create one or more 802.1Q VLAN subinterfaces on the Ethernet bundle and configure the Layer 2 encapsulation and IP address for each subinterface.

Perform this sub-task from global configuration mode, after you create the Ethernet bundle.

Procedure

1. Create a new VLAN subinterface and assign it to the Ethernet bundle you created earlier.

Example

```
Router(config)# interface Bundle-Ether 3.1
```

Replace the *bundle-id* with the ID you created earlier. Replace the *vlan-id* with a subinterface identifier. The range is from 1 to 4094, inclusive (0 and 4095 are reserved).

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

2. Set the Layer 2 encapsulation of the subinterface.

   Example

   ```
   Router(config-subif)# encapsulation dot1q 100
   ```

3. Assign an IP address and subnet mask to the subinterface.

   Example

   ```
   Router(config-subif)# ipv4 address 10.1.2.3/24
   ```

4. (Optional) If a link is in the down state, bring it up.

   Example

   ```
   Router(config-subif)# no shutdown
   ```

5. Exit subinterface configuration mode for the VLAN subinterface.

   Example

   ```
   Router(config-subif)# exit
   ```

6. (Optional) Repeat the VLAN subinterface creation steps to add more VLANs to the bundle.

What to do next

Perform the Assign Ethernet links to the Ethernet bundle sub-task next.

Assign Ethernet links to the Ethernet bundle

This topic describes how to add physical Ethernet member links to the Ethernet bundle, optionally enable LACP fast-switchover, and commit the configuration as the final stage in configuring a VLAN bundle.

Add physical Ethernet interfaces to the Ethernet bundle so that the VLAN subinterfaces become active on the bundle.

Perform this sub-task from global configuration mode, after you create the VLAN subinterfaces on the Ethernet bundle.

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

Procedure

1. Enter interface configuration mode for the Ethernet interface to add to the bundle.

   Example

   ```
   Router(config)# interface GigabitEthernet 1/0/0/0
   ```

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

2. (Optional) Disable the wait-while timer in the LACP state machine for faster switchover.

   Example

   ```
   Router(config-if)# lacp fast-switchover
   ```

   Use this option if you enabled 1:1 link protection (you set the value of the **bundle maximum-active links** command to 1) on a bundle with member links running LACP.

3. Save the configuration changes using the **end** or **commit** command.

   Example

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

Configure LACP fallback

This topic describes how to configure the LACP Fallback feature on an Ethernet link bundle to allow the peer server to bring up the LAG and boot before receiving any LACP PDUs.

Enable LACP fallback so that an active LACP interface can establish a Link Aggregation Group (LAG) port-channel and keep one port active while the server completes its boot process over that port.

This task configures the LACP fallback timeout on a Bundle-Ether interface.

Procedure

1. Enter global configuration mode using the **configure** command.

   Example

   ```
   Router# configure
   ```

2. Create and name a new Ethernet link bundle using the **interface Bundle-Ether** *bundle-id* command.

   Example

   ```
   Router(config)# interface Bundle-Ether 3
   ```

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

3. Specify a primary IPv4 address for the interface.

   Example

   ```
   Router(config-if)# ipv4 address 192.168.1.27 255.0.0.0
   ```

4. Enable the LACP Fallback feature using the **bundle lacp-fallback timeout** *number* command.

   Example

   ```
   Router(config-if)# bundle lacp-fallback timeout 4
   ```

5. Save the configuration changes using the **end** or **commit** command.

   Example

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

6. (Optional) Verify the MA information of the bundle manager for the configured bundle.

   Example

   ```
   Router# show bundle infrastructure database ma bdl-info Bundle-e1010 | inc
   "fallback"
   Router# show bundle infrastructure database ma bdl-info Bundle-e1015 | inc
   "fallback"
   ```

   Confirm that the fallback configuration is reflected in the bundle manager output.

Configure the default LACP short period time interval

This topic describes how to configure the default LACP short period time interval of 1000 milliseconds for sending and receiving LACP packets on a Gigabit Ethernet interface.

Enable the LACP short period using the default time interval to speed up detection and recovery from link failures on bundle members.

This task configures the default short period time interval for sending and receiving LACP packets on a Gigabit Ethernet interface. This procedure also enables the LACP short period.

Procedure

1. Enter global configuration mode using the **configure** command.

Example

```
Router# configure
```

2. Create a Gigabit Ethernet interface and enter interface configuration mode.

   Example

   ```
   Router(config)# interface HundredGigE 0/1/0/1
   ```

3. Specify the bundle interface and put the member interface in active mode.

   Example

   ```
   Router(config-if)# bundle id 1 mode active
   ```

4. Configure the short period time interval for sending and receiving LACP packets using the default time period of 1000 milliseconds (1 second).

   Example

   ```
   Router(config-if)# lacp period short
   ```

5. Save the configuration changes using the **end** or **commit** command.

   Example

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

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

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

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

Configure custom LACP short period time intervals

This topic describes how to configure a custom LACP short period time interval, in the range 100 to 1000 milliseconds, for sending and receiving LACP packets on a Gigabit Ethernet interface.

Set a custom LACP short period time interval to tune the frequency of LACP packet exchanges for faster detection and recovery from link failures.

This task configures a custom short period time interval for sending and receiving LACP packets on a Gigabit Ethernet interface. The interval can be in the range 100 to 1000 milliseconds, in multiples of 100.

Procedure

1. Enter global configuration mode using the **configure** command.

   Example

   ```
   Router# configure
   ```

2. Create a Gigabit Ethernet interface and enter interface configuration mode.

   Example

   ```
   Router(config)# interface HundredGigE 0/1/0/1
   ```

3. Specify the bundle interface and put the member interface in active mode.

   Example

   ```
   Router(config-if)# bundle id 1 mode active
   ```

4. Configure a custom period time interval for sending and receiving LACP packets using the **lacp period** *time-interval* command.

   Example

   ```
   Router(config-if)# lacp period 300
   ```

   The interval can be in the range 100 to 1000 milliseconds, in multiples of 100.

5. Save the configuration changes using the **end** or **commit** command.

   Example

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

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

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

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

Configure a member link as unviable

This topic describes how to designate a bundle member link as forwarding-unviable to introduce a delay during which the link is treated as standby before it is used for data transmission.

Delay the use of a member link that is not yet fully ready to handle data transmission by marking it as forwarding-unviable so that the peer treats it as a standby link.

Perform this task to designate a member link as unviable on a bundle member interface.

Procedure

1. Enter interface configuration mode for the bundle member interface.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/0/34
   ```

2. Designate the member link as unviable using the **forwarding-unviable** command.

   Example

   ```
   Router(config-if)# forwarding-unviable
   ```

3. Save the configuration changes using the **commit** command, and exit the configuration session.

   Example

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

4. Verify the running configuration on the interface.

   Example

   ```
   Router# show running-config interface HundredGigE 0/0/0/34
   interface HundredGigE0/0/0/34
    forwarding-unviable
   !
   ```

   Confirm that the **forwarding-unviable** command is present in the interface configuration.

5. Verify the forwarding-viable status of LAG members using the **show bundle** command.

   Example

   ```
   Router# show bundle

   Bundle-Ether3
     Status:                                  Down
     Local links <active/standby/configured>:   0 / 0 / 1
     Local bandwidth <effective/available>:     0 (0) kbps
     MAC address (source):                    78c6.9991.3504 (Chassis pool)
     Inter-chassis link:                      No
     Minimum active links / bandwidth:        1 / 1 kbps
     Maximum active links:                    64
   ```

```
  Wait while timer:                          2000 ms

  Port                  Device           State        Port ID         B/W,
kbps
  --------------------  ---------------  -----------  --------------
----------
  Hu0/0/0/34            Local            Standby      0x8000, 0x0001
100000000
      Link is not forwarding viable and in standby state
```

Confirm that the member interface is in the standby state and that the output reports "Link is not forwarding viable and in standby state" for that interface.

Configure link-flap err-disable

This topic describes how to configure link-flap err-disable to place interfaces that flap excessively into an error-disabled state, and recover err-disabled interfaces manually or automatically after they stabilize.

Prevent repeated link flaps from destabilizing the network by automatically disabling affected interfaces and providing options for recovery.

Link-flap err-disable detects frequent interface flaps and disables the interface to maintain network stability. Recovery methods allow administrators to restore normal operation as needed.

Before you begin

- Determine the target interface and the desired recovery method: manual or automatic.

Procedure

1. Configure link-flap err-disable with a specific count and interval.

   Example

   ```
   Router# configure
   Router(config)# interface GigabitEthernet0/0/0/0
   Router(config-if)# fault-management link-flap error-disable count 30 interval
    420
   Router(config-if)# commit
   ```

2. Verify which interfaces are err-disabled.

   Example

   ```
   Router# show error-disable
   Interface           Error-Disable reason           Retry (s)  Time disabled
   ------------------  -----------------------------  ---------  ---------------
   Fo0/1/0/25          link-flap-exceeded                  ---        10:43:18
   Fo0/1/0/26          link-flap-exceeded                  ---        10:43:18
   ```

   Confirm the interfaces that are in the err-disabled state and their disable reasons.

3. Recover an err-disabled interface using one of these methods.

- Configure automatic recovery to bring up the interface after a specific time using the **error-disable recovery cause** command.

```
Router(config)# error-disable recovery cause link-flap-exceeded interval
30
```

The default timer is 300 seconds for the **error-disable recovery cause link-flap-exceeded interval** command.

- Manually recover an interface by resetting the interface state.

```
Router(config)# interface <interface-type> <interface-path>
Router(config-if)# shut
Router(config-if)# no shut
Router(config-if)# commit
```

> **Note**
>
> Although shut and no shut resets the interface state, the recommended CLIs to clear err-disable are **clear error-disable interface** and **clear error-disable location**.

4. Verify the err-disable state on the interface.

Example

```
Router# show error-disable interface TenGigE 0/0/0/1
Interface           Error-Disable reason           Retry (s)  Time disabled
------------------  -----------------------------  ---------  ---------------
Te0/0/0/1           link-flap-exceeded                   200        11:35:07
```

Confirm the recovery retry timer and the current disabled duration for the interface.

5. Clear the err-disable state from an interface explicitly.

Example

```
Router# clear error-disable interface Hu0/0/0/0
```

6. Clear the err-disable state from all interfaces in a specific location.

Example

```
Router# clear error-disable location interface TenGigE 0/0/0/1
```

Verify a 128-member Ethernet link bundle

This topic describes how to verify the configured member count, operational member state, and load-balancing membership for a configured 128-member Ethernet link bundle, and confirm the supported bundle scale for the specified location on the Cisco 8000 Series Router.

Use these checks after you add the expected physical interfaces to the Bundle-Ether interface.

On supported systems, an Ethernet link bundle can contain up to 128 physical member interfaces.

Before you begin

Identify the bundle ID and the node that hosts the bundle members.

Procedure

1. Verify the operational status of the Ethernet link bundle and its member interfaces using these commands.

   Example

   ```
   Router# show bundle brief
   Router# show bundle bundle-ether bundle-id
   Router# show interfaces bundle-ether bundle-id
   ```

   Confirm that the configured-member count is 128. Confirm that the active-member count matches the number of operational members and that every expected member interface appears in the output.

2. Verify the supported bundle scale for the specified location.

   This example output is from a Cisco 8808 8-Slot modular system with an 88-LC2-36EF-M line card.

   Example

   ```
   Router# show ethernet hardware capability location 0/3/CPU0

   System Interface Scale:
       Max L3 Sub-Interfaces                : 30000
       Max L2 Sub-Interfaces                : 131072

     Bundle Scale:
       Max Bundles                          : 1023
       Max Bundle Members per Bundle        : 128

     LC Interface Scale:
       Max L3 Sub-Interfaces                : 16000
       Max L2 Sub-Interfaces                : 57344

     Main Interface Scale:
       Max L3 Sub-Interfaces                : 2000
       Max L2 Sub-Interfaces                : 14336
       Max Sub-Interfaces                   : 14336
   ```

   This example output is from a Cisco P200 64x800G OSFP 3RU fixed system, 8223-64EF-M.

   Example

   ```
   Router# show ethernet hardware capability location 0/RP0/CPU0

   System Interface Scale:
       Max L3 Sub-Interfaces                : 8000
       Max L2 Sub-Interfaces                : 57344

     Bundle Scale:
       Max Bundles                          : 1023
       Max Bundle Members per Bundle        : 128

     LC Interface Scale:
   ```

```
    Max L3 Sub-Interfaces                : 8000
    Max L2 Sub-Interfaces                : 57344

  Main Interface Scale:
    Max L3 Sub-Interfaces                : 2000
    Max L2 Sub-Interfaces                : 14336
    Max Sub-Interfaces                   : 14336
```

15 Virtual loopback and null interfaces

Topics:

This chapter describes configuring virtual loopback and null interfaces to improve routing reliability and manage network traffic effectively.

Prerequisites for configuring virtual interfaces

This topic describes the prerequisites that you must meet before you configure virtual loopback, null, and virtual IPv4 interfaces on the Cisco 8000 Series Router.

Before you configure virtual interfaces, ensure that you meet the following prerequisite:

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

Virtual loopback interface

This topic describes the virtual loopback interface, its always-up single-endpoint behavior, and its typical uses as a routing-protocol session endpoint and a target for ping verification on the Cisco 8000 Series Router.

The virtual loopback interface is a virtual interface that

- remains active at all times
- emulates a physical interface, and
- processes transmitted packets locally.

Virtual loopback interfaces perform these functions in Cisco IOS XR Software:

- Loopback interfaces can act as a termination address for routing protocol sessions. This allows routing protocol sessions to stay up even if the outbound interface is down.
- You can ping the loopback interface to verify that the router IP stack is working properly.
- In applications where other routers or access servers attempt to reach a virtual loopback interface, you must configure a routing protocol to distribute the subnet assigned to the loopback address.
- Packets routed to the loopback interface are rerouted back to the router or access server, and processed locally. IP packets routed out to the loopback interface but not destined to the loopback interface are dropped. Under these two conditions, the loopback interface can behave like a null interface.

Null interface

This topic describes the null interface, which is always up, and provides an alternative to access lists for filtering unwanted traffic on the Cisco 8000 Series Router.

The null interface is a virtual interface that

- remains permanently active
- discards all received traffic, and
- provides an alternative method for filtering undesired network traffic.

A null interface functions similarly to the null devices available on most operating systems. This interface is always up and can never forward or receive traffic; encapsulation always fails. The null interface provides an alternative method of filtering traffic. You can avoid the overhead that is involved with using access lists by directing undesired network traffic to the null interface.

Key characteristics of null interfaces

The system supports several key characteristics and configuration options for null interfaces:

- The only interface configuration command that you can specify for the null interface is the **ipv4 unreachables** command.
- With the **ipv4 unreachables** command, if the software receives a nonbroadcast packet destined for itself that uses a protocol it does not recognize, it sends an Internet Control Message Protocol (ICMP) protocol unreachable message to the source.
- If the software receives a datagram that it cannot deliver to its ultimate destination because it knows of no route to the destination address, it replies to the originator of that datagram with an ICMP host unreachable message.
- By default, the system enables the **ipv4 unreachables** command. If you do not want ICMP to send protocol unreachable messages, configure the **ipv4 icmp unreachable disable** command.
- By default, the system creates the Null 0 interface during the boot process and you cannot remove it. You can configure the **ipv4 unreachables** command for this interface, but most configuration is unnecessary because this interface discards all packets that the system sends to it.
- Use the **show interfaces null0** command to display the Null 0 interface.

Virtual management interface

This topic describes the virtual management interface that enables management access to the Cisco 8000 Series Router from a single address.

The virtual management interface is a management configuration that

- provides single-address access to a router with a management network without prior knowledge of which RP is active
- persists across route switch processor failovers, and
- shares a common subnet with management Ethernet interfaces on both the RPs.

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

Active and standby RPs and virtual interface configuration

This topic describes how virtual interfaces persist across active and standby route processor (RP) failover on the Cisco 8000 Series Router, and how to ensure that protocols such as TACACS+, SNMP, NTP, and syslog use the virtual IPv4 address as their source address.

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

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

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

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

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

Configure virtual loopback interfaces

This topic describes how to configure a basic virtual loopback interface on the Cisco 8000 Series Router.

Use this procedure to configure a basic virtual loopback interface to serve as an always-up endpoint for routing protocol sessions or as a target for ping-based reachability verification.

Procedure

1. Enter global configuration mode using the **configure** command.

   Example

   ```
   Router# configure
   ```

2. Enter interface configuration mode and name the new loopback interface.

   Example

   ```
   Router(config)# interface Loopback 3
   ```

3. Assign an IP address and subnet mask to the virtual loopback interface.

   Example

   ```
   Router(config-if)# ipv4 address 172.18.189.38/32
   ```

4. Save the configuration changes using the **end** or **commit** command.

   Example

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

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.

5. (Optional) Display the configuration of the loopback interface.

   Example

```
Router# show interfaces Loopback 3
```

6. Use the **show interfaces Loopback** command to verify the configuration.

   Example

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

Loopback3 is up, line protocol is up
 Hardware is Loopback interface(s)
 Internet address is 172.18.189.38/32
 MTU 1514 bytes, BW Unknown
   reliability 0/255, txload Unknown, rxload Unknown
 Encapsulation Loopback, loopback not set
 Last clearing of "show interface" counters never
 5 minute input rate 0 bits/sec, 0 packets/sec
 5 minute output rate 0 bits/sec, 0 packets/sec
   0 packets input, 0 bytes, 0 total input drops
   0 drops for unrecognized upper-level protocol
   Received 0 broadcast packets, 0 multicast packets
   0 packets output, 0 bytes, 0 total output drops
   Output 0 broadcast packets, 0 multicast packets
```

Guidelines and restrictions for configuring virtual loopback interfaces

This topic describes the IP-address uniqueness restrictions that you must observe when you assign an IP address to a virtual loopback interface on the Cisco 8000 Series Router.

Observe the following restrictions when you configure a virtual loopback interface:

- The IP address of a loopback interface must be unique across all routers on the network.

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

Configure null interfaces

This topic describes how to configure a basic null interface on the Cisco 8000 Series Router by entering global configuration mode, entering the Null 0 interface configuration mode, saving the configuration, and verifying the interface.

Configure a basic null interface to serve as an always-up sink that discards undesired traffic without the overhead of an access list.

Procedure

1. Enter global configuration mode using the **configure** command.

   Example

   ```
   Router# configure
   ```

2. Enter the Null 0 interface configuration mode.

   Example

   ```
   Router(config)# interface null 0
   ```

3. Save the configuration changes using the **end** command.

   Example

   ```
   Router(config-null0)# end
   ```

4. Verify the configuration of the null interface.

   Example

   ```
   Router# show interfaces Null 0

   Null0 is up, line protocol is up
    Hardware is Null interface
    Internet address is Unknown
    MTU 1500 bytes, BW Unknown
      reliability 0/255, txload Unknown, rxload Unknown
    Encapsulation Null, loopback not set
    Last clearing of "show interface" counters never
    5 minute input rate 0 bits/sec, 0 packets/sec
    5 minute output rate 0 bits/sec, 0 packets/sec
      0 packets input, 0 bytes, 0 total input drops
      0 drops for unrecognized upper-level protocol
      Received 0 broadcast packets, 0 multicast packets
   ```

```
   0 packets output, 0 bytes, 0 total output drops
   Output 0 broadcast packets, 0 multicast packets
```

Configure virtual IPv4 interfaces

This topic describes how to configure an IPv4 virtual address for the management Ethernet interface on the Cisco 8000 Series Router so that you can access the router from a single virtual address that persists across route processor (RP) failover.

Configure an IPv4 virtual address so that you can access the router from a single virtual address on the management network, without prior knowledge of which RP is active.

Procedure

1. Enter global configuration mode using the **configure** command.

   Example

   ```
   Router# configure
   ```

2. Define an IPv4 virtual address for the management Ethernet interface.

   Example

   ```
   Router(config)# ipv4 virtual address 10.3.32.154/8
   ```

3. Save the configuration changes using the **end** or **commit** command.

   Example

   ```
   Router(config)# end
   ```

   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.

16 GRE Tunnels

Topics:

This chapter describes configuring GRE tunnels, covering encapsulation, decapsulation, and NVGRE load-balancing for Cisco 8000 series routers.

GRE tunnels

This topic describes Generic Routing Encapsulation (GRE) tunnels on the Cisco 8000 Series Router, including how they encapsulate a payload of one protocol inside an outer IP packet for delivery between two tunnel endpoints.

The Generic Routing Encapsulation is a tunneling protocol that

- provides a simple approach to transport packets of one protocol over another protocol through encapsulation
- encapsulates a payload inside an outer IP packet while the system delivers this inner packet to a destination network, and
- behaves as a virtual point-to-point link between two endpoints that identify this link using the tunnel source and tunnel destination address.

Other IP routers along the way do not parse the payload (the inner packet); they only parse the outer IP packet as they forward it toward the GRE tunnel endpoint. Upon reaching the tunnel endpoint, GRE encapsulation is removed and the payload is forwarded to the packet's ultimate destination.

Table 29: Feature History Table

Feature Name	Release Information	Description
Network Virtualization using Generic Routing Encapsulation hash field selections	Release 26.2.1	You can now improve load balancing for Network Virtualization using Generic Routing Encapsulation (NVGRE) traffic by excluding the NVGRE payload from the hash calculation. This feature optimizes traffic distribution across multiple paths, preventing uneven load caused by hashing on the NVGRE payload. The feature works by modifying the Cisco Express Forwarding (CEF) load-balancing hash algorithm to exclude the NVGRE payload field. This adjustment ensures that the hash calculation uses only relevant header fields, leading to better traffic distribution and network performance. This feature introduces these changes: CLI: • **cef platform load-balancing nvgre payload exclude** • **show cef platform load-balancing**
GRE tunnel	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
GRE tunnel	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D

Feature Name	Release Information	Description
Disabling time-to-live (TTL) decrement at GRE encapsulation	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Disabling time-to-live (TTL) decrement at GRE encapsulation	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM
GRE tunnel	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700)(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*). The Generic Routing Encapsulation (GRE) feature that transports packets of one protocol over another protocol in a simplified manner using encapsulation is now supported on the following hardware. *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 88-LC1-36EH
Disabling time-to-live (TTL) decrement at GRE encapsulation	Release 7.3.2	This feature allows you to disable the time-to-live (TTL) decrement of the incoming packets. The result is that encapsulation of the original incoming packet takes place without any change in the TTL value. This feature avoids dropping incoming packets with a TTL value equal to one after GRE encapsulation. Before this release, the TTL value of incoming packets was decremented by one before GRE decapsulation. This feature introduces the **tunnel ttl disable** command.

Feature Name	Release Information	Description
GRE tunnel	Release 7.3.1	Generic Routing Encapsulation (GRE) provides a simple approach to transporting packets of one protocol over another protocol using encapsulation. This capability is now extended to the Cisco 8000 Series Routers. This feature supports: • Unidirectional GRE encapsulation • Unidirectional GRE decapsulation And introduces the following commands: • **show interface tunnel-ip <> accounting (encap)** • **show interface tunnel-ip <> accounting (decap)**
Outer-header hashing support for MPLSoGRE and IPoGRE traffic	Release 7.3.1	This feature allows load-balancing of GRE traffic in transit routers. A transit node distributes incoming GRE traffic evenly across all available ECMP links in a GRE tunnel topology. A hashing function uses GRE outer and inner header tuples such as source IP, destination IP, protocol, and router ID to determine traffic entropy. This capability is now extended to the Cisco 8000 Series Routers.

A tunnel configured using encapsulation mode performs encapsulation of IPv4/IPv6 payload inside the GRE header. A tunnel configured using decapsulation mode performs the opposite. Here, outer GRE header is decapsulated and the inner IPv4/IPv6/MPLS payload is forwarded to the next hop router. Both encapsulation and decapsulation tunnel interfaces collect statistics periodically. The statistics can be displayed on demand using the CLI commands `show interface tunnel-ip1 accounting` and `show policy-map type pbr address-family ipv4 statistics`. For more information, see Unidirectional GRE encapsulation (GREv4) and Unidirectional GRE decapsulation (GREv4).

To perform load-balancing of GRE traffic in transit routers, a transit node distributes incoming GRE traffic evenly across all available ECMP links in a GRE tunnel topology. Furthermore, to determine traffic entropy, a hashing function uses GRE outer and inner header tuples such as source IP, destination IP, protocol, and router ID.

Supported features on a GRE tunnel

This topic provides the encapsulation, decapsulation, and TTL-related features that a GRE tunnel supports on the Cisco 8000 Series Router.

A GRE tunnel supports the following features:

- GRE or IP-in-IP tunnels support 16 unique source addresses. These 16 unique source addresses are repeated multiple times to configure 1000 encapsulation tunnels or 64 decapsulation tunnels.
- GRE encapsulation supports the following features:
  - IPv4/IPv6 over GRE IPv4 transport
  - MPLS PoP over GRE IPv4 transport
  - ABF (Access List Based Forwarding) v4/v6 over GRE
  - VRF (Virtual Routing and Forwarding) support over GRE
- GRE decapsulation supports the following features:
  - PBR-based GRE decapsulation configuration

  - CLI-based GRE decapsulation configuration
  - IPv4/IPv6 over GRE decapsulation
  - MPLS/SRTE over GRE decapsulation
  - A GRE tunnel in decapsulation mode has only tunnel source configured, without any tunnel destination address. This decapsulated GRE tunnel behaves like a P2MP (Point-to-multipoint) tunnel, which means that an incoming GRE packet can have any source IP address and matching destination IP address to the tunnel source configured. However, once a source IP address is used for decapsulated P2MP tunnel, it cannot be re-used with other decapsulation tunnels.

- The command `tunnel ttl disable` is supported. This command controls TTL decrement of a packet being encapsulated. After configuring this command for a tunnel interface, TTL value of incoming packet is not decremented by one, and original incoming packet is encapsulated without changing the TTL. By default, `tunnel ttl disable` is not configured. This means that the TTL of incoming packets is decremented by one before GRE encapsulation.

  For example, consider an incoming packet that had the TTL value equal to one. On GRE encapsulation, the TTL value is decremented by one and becomes zero. Therefore the router will discard the packet and send an ICMP message back to the originating host. Using this feature, you can disable TTL decrement and avoid the packet discard.

Configuration example: Disable TTL decrement

```
Router# configure
Router(config)# interface tunnel-ip30016
Router(config-if)# tunnel ttl disable
Router(config-if)# commit
```

Limitations for configuring GRE tunnels

This topic lists the limitations that you must observe when you configure GRE tunnels on the Cisco 8000 Series Router.

Observe the following limitations when you configure GRE tunnels:

- GRE tunnels configured without any decapsulation or encapsulation mode support only ERSPAN feature.
- Do not create multiple GRE/IP-in-IP tunnels with the same pair of source and destination IP address or interface name. Configure all tunnels with unique source-destination pairs. In an encapsulation or decapsulation tunnel where only either source or destination is mentioned, the source-destination pair should also be unique when compared to other encapsulation or decapsulation tunnels.
- Bi-directional GRE tunnel is not supported.
- Routing protocols over GRE tunnels are not supported.
- Multicast over GRE is not supported.
- GRE KA (Keep Alive) is not supported.
- GRE parameters such as MTU (Maximum Transmission Unit) and key functionalities are not supported.

Configure a GRE tunnel

This topic describes how to configure a GRE tunnel on the Cisco 8000 Series Router by creating a tunnel interface and defining the tunnel source and destination.

Use this procedure to configuring a GRE tunnel that involves creating a tunnel interface and defining the tunnel source and destination. The router supports only unidirectional GRE with either encapsulation or decapsulation mode.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter interface configuration mode for the tunnel-ip interface.

   Example

   ```
   Router(config)# interface tunnel-ip1
   ```

3. Assign an IPv4 address and subnet mask to the tunnel interface.

   Example

   ```
   Router(config-if)# ipv4 address 101.0.1.2 255.255.255.0
   ```

4. Assign an IPv6 address to the tunnel interface.

   Example

   ```
   Router(config-if)# ipv6 address 101:0:1::2/64
   ```

5. Set the tunnel mode to GRE IPv4 with either encapsulation or decapsulation.

   Example

   ```
   Router(config-if)# tunnel mode gre ipv4 [encap | decap]
   ```

6. Configure the tunnel source address.

   Example

   ```
   Router(config-if)# tunnel source 2.2.1.1
   ```

7. Configure the tunnel destination address.

   Example

   ```
   Router(config-if)# tunnel destination 2.2.2.1/32
   ```

8. Save the configuration.

   Example

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

9. Exit interface configuration mode.

Example

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

What to do next

Bi-directional GRE tunnel supports only ERSPAN.

To configure ABFv4/v6 over GRE, use this configuration:

```
router static
  address-family ipv4 unicast
  201.0.1.0/24 tunnel-ip1
  address-family ipv6 unicast
  201:0:1::0/64 tunnel-ip1

ipv4 access-list abf-gre
  1 permit ipv4 any any nexthop1 ipv4 201.0.1.2
ipv6 access-list abf6-gre
  1 permit ipv6 any any nexthop1 ipv6 201:0:1::2

interface HundredGigE0/0/0/24
  ipv4 address 24.0.1.1/24
  ipv6 address 24:0:1::1/64
  ipv4 access-group abf-gre ingress
  ipv6 access-group abf6-gre ingress
!
```

To configure MPLS PoP label over GRE, use this configuration:

```
router static
  address-family ipv4 unicast
  201.0.1.0/24 tunnel-ip1
  address-family ipv6 unicast
  201:0:1::0/64 tunnel-ip1

mpls static
  interface HundredGigE0/0/0/24
  lsp gre
    in-label 30501 allocate
    forward path 1 resolve-nexthop 201.0.1.2 out-label pop
!
```

GRE encapsulation and decapsulation over BVI

This topic describes GRE encapsulation and decapsulation over a Bridge-Group Virtual Interface (BVI) on the Cisco 8000 Series Router.

Bridge-Group Virtual Interface (BVI) is a virtual interface that

- acts like a normal routed interface
- serves as a gateway for the corresponding bridge-domain to a routed interface, and
- represents the link between the bridging and the routing domains on the router.

Table 30: Feature History Table

Feature Name	Release Information	Description
GRE encapsulation and decapsulation over BVI	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-36EH+A8:B12 • 88-LC1-52Y8H-EM
GRE encapsulation and decapsulation over BVI	Release 7.5.4	You can now transport packets using the GRE protocol over Bridge-Group Virtual Interfaces (BVI). This feature uses GRE to encapsulate packets between two endpoints and transmit the encapsulated packets over a BVI interface. At the destination, the GRE packet is decapsulated. GRE encapsulation and decapsulation over BVI allows transmitting packets securely using network layer protocols while maintaining Layer 2 connectivity between the physical interfaces.

How GRE encapsulation and decapsulation over BVI works

The BVI processes GRE traffic through these steps:

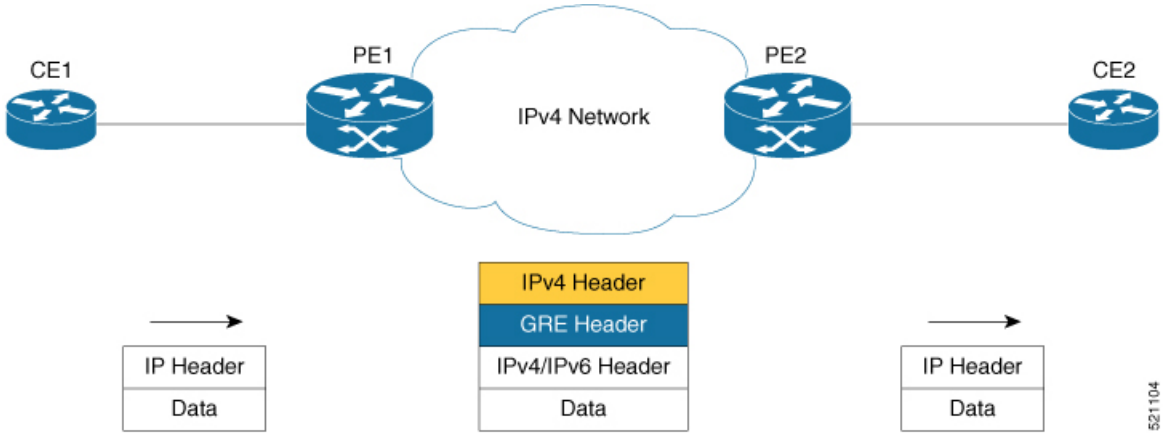
- The system adds the GRE header to the original IP packet before it sends the packet to the BVI.
- The BVI bridges the encapsulated packet to the destination interface.
- The destination interface removes the GRE header from the original IP packet.
- The interface forwards the resulting IP packet to the final destination.

For information on BVI, see the *Integrated Routing and Bridging* section in the *L2VPN Configuration Guide for Cisco 8000 Series Routers*.

Unidirectional GRE encapsulation (GREv4)

This topic describes unidirectional GRE encapsulation (GREv4) on the Cisco 8000 Series Router, where the tunnel encapsulates an IPv4 or IPv6 payload inside a GRE header for transport across an IP network.

A tunnel configured using encapsulation mode performs encapsulation of IPv4/IPv6 payload inside the GRE header. The following figure shows GRE encapsulation. Routers in the IP cloud have no knowledge of encapsulated IP source address or destination address.



Configuration example

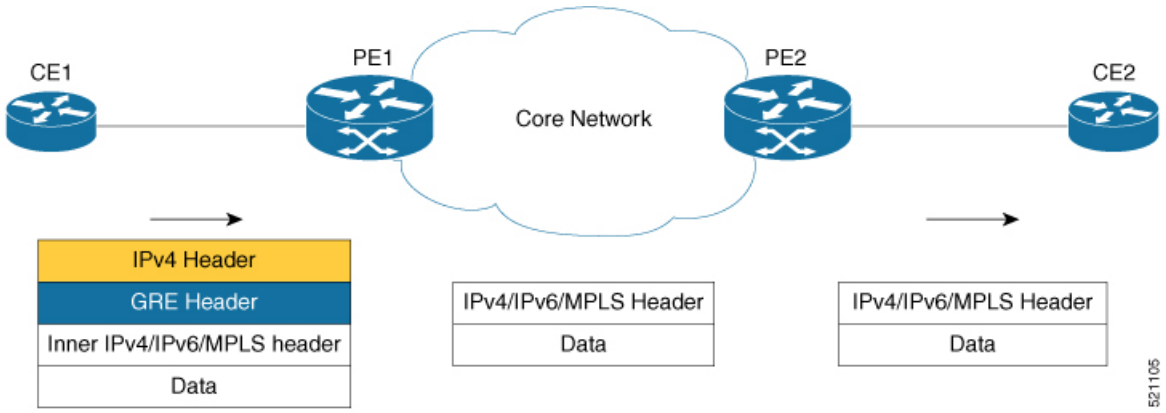
This example shows how to configure GRE tunnel encapsulation:

```
interface tunnel-ip1
  ipv4 address 101.0.1.1/24
  ipv6 address 101:0:1::1/64
  tunnel mode gre ipv4 encap
  tunnel source [ loopback1 | <any-ipaddres> | any-interface]
  tunnel destination [ 20.0.1.1/32 | 20.0.1.0/24 | 20.0.1.0/28]

router static
  address-family ipv4 unicast
  201.0.1.0/24 tunnel-1

router static
  address-family ipv6 unicast
  201:0:1::0/64 tunnel-1
```

Unidirectional GRE decapsulation (GREv4)

This topic describes unidirectional GRE decapsulation (GREv4) on the Cisco 8000 Series Router, where the router strips the outer GRE header and forwards the inner IPv4, IPv6, or MPLS payload to the next-hop router.

In unidirectional GRE decapsulation, the outer GRE header is decapsulated and the inner IPv4/IPv6/MPLS payload is forwarded to the next hop router. The following figure shows GRE decapsulation. In the figure, PE1 strips off outer GRE header and inner payload is forwarded as regular IPv4/IPv6/MPLS forwarding.



Configuration example

There are two methods to configure GRE tunnel decapsulation:

1. CLI-based tunnel decapsulation configuration

```
interface tunnel-ip1
  ipv4 address 101.0.1.1/24
  ipv6 address 101:0:1::1/64
  tunnel mode gre ipv4 decap
  tunnel source [ loopback1 | <any-ipaddres> | any-interface]
  tunnel destination [ 20.0.1.1/32 | 20.0.1.0/24 | 20.0.1.0/28]
```

2. PBR-based tunnel decapsulation configuration

```
class-map type traffic match-all test_gre1
                 match protocol gre
                 match destination-address ipv4 10.0.1.2 255.255.255.255
                 match source-address ipv4 10.10.10.1 255.255.255.255
                 end-class-map
policy-map type pbr P1-test
                 class type traffic test_gre1  decapsulate gre
vrf-policy vrf default address-family ipv4 policy type pbr input P1-test
```

ECMP and LAG hashing for NVGRE flows

This topic describes ECMP and LAG hashing for NVGRE flows on the Cisco 8000 Series Router, which lets transit routers load-balance NVGRE traffic across ECMP and LAG paths using the GRE payload.

Network Virtualization using Generic Routing Encapsulation (NVGRE) endpoint is a network device that

- acts as an interface between physical and virtual networks
- encapsulates Ethernet data frames for GRE tunnels, and
- decapsulates GRE packets to recover original Ethernet frames.

The NVGRE endpoint performs the following operations:

- The endpoint encapsulates Ethernet data frames for transport through a GRE tunnel.
- The system bridges and routes the encapsulated GRE packet to the destination.
- The destination endpoint decapsulates the GRE packet to recover the original Ethernet frame.

- The RFC 7637 document describes the NVGRE protocol.

Table 31: Feature History Table

Feature Name	Release Information	Feature Description
ECMP and LAG Hashing for NVGRE Flows	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
ECMP and LAG Hashing for NVGRE Flows	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
ECMP and LAG Hashing for NVGRE Flows	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-36EH+A8:B12 • 88-LC1-52Y8H-EM
ECMP and LAG Hashing for NVGRE Flows	Release 7.5.2	This feature allows transit routers to load balance the GRE traffic, based on GRE payload. A transit node distributes incoming GRE traffic across ECMP and LAG paths in a GRE tunnel topology. A hashing function uses GRE payload that consists of inner Ethernet frame with destination MAC and source MAC addresses, to derive the traffic entropy. ECMP and LAG hashing is enabled on Cisco 8000 series routers by default.

NVGRE uses the following header information for encapsulation:

Header	Parameters
Outer Ethernet Header	Destination MAC address, Source MAC address
Outer IP Header	IPv4 and IPv6 addresses as delivery protocol for GRE
GRE Header	GRE protocol type 0x6558 (transparent Ethernet bridging
GRE Payload	Inner Ethernet frame with Destination MAC address and Sou

For load balancing the GRE traffic, the transit router uses GRE payload that consists of inner Ethernet frame with destination MAC and source MAC addresses. The transit router derives the traffic entropy information from the GRE payload.

The hashing function considers the following parameters of GRE packets, along with Router ID, for load balancing the GRE traffic:

Header	Parameters
Outer IPv4 Header	Source IP address, Destination IP address, IP Protocol (G
Outer IPv6 Header	Source IP address, Destination IP address, Flow Label, IP N
Inner Header	Destination MAC address, Source MAC address, Ether Ty

Restrictions for ECMP and LAG hashing for NVGRE flows

This topic lists the restrictions that you must observe when you use ECMP and LAG hashing for NVGRE flows on the Cisco 8000 Series Router.

ECMP and LAG hashing does not support:

- Outer IPv4 header with Options field.
- Outer IPv6 header with extension headers.

Network Virtualization using Generic Routing Encapsulation hash field selections

This topic describes Network Virtualization using Generic Routing Encapsulation (NVGRE) hash field selections on the Cisco 8000 Series Router, which control whether NVGRE inner payload fields are included in the load-balancing hash calculation on Cisco Silicon One Q100 and Q200 ASIC-based systems.

Network Virtualization using Generic Routing Encapsulation (NVGRE) hash field selections are load-balancing configuration options that

- provide the ability to control whether NVGRE inner payload fields are included in the hash calculation, and
- configure hash field selection to consider only the outer headers, ensuring deterministic load distribution when the NVGRE payload is non-IP.

Benefits of NVGRE hash field selections

These are the benefits of NVGRE hash field selections:

- Balances NVGRE traffic load across Equal Cost Multipath (ECMP) and Link Aggregation Group (LAG) paths.
- Reduces traffic congestion and bottlenecks in virtualized network environments.
- Enhances network efficiency by using inner Ethernet frame information for hashing.

Configuration guidelines for NVGRE hash field selection

This topic lists the guidelines that you must follow to configure NVGRE hash field selection on the Cisco 8000 Series Router to ensure predictable and reliable load-balancing.

These guidelines apply for configuring NVGRE hash field selection:

- Use the GRE payload fields, including inner Ethernet frame source and destination MAC addresses, for hash computation.
- Supported only on Cisco Silicon One Q100 and Q200 ASIC-based systems.
- Use the feature to distribute NVGRE traffic across multiple physical paths for better resource utilization.

Restrictions for NVGRE hash field selection

This topic lists the restrictions that you must observe when you configure NVGRE hash field selection on the Cisco 8000 Series Router.

These restrictions apply to NVGRE hash field selection:

- It does not support outer IPv4 headers that contain options.
- It does not support outer IPv6 headers with extension headers.

How NVGRE hash field selection works

This topic describes how the Cisco 8000 Series Router determines which packet header fields are used for load-balancing NVGRE traffic, based on the operator configuration.

Summary

NVGRE hash field selection works by allowing configurable hash field selection, ensuring predictable load distribution for NVGRE traffic across network links.

The key components involved in the NVGRE hash field selection process are:

- ASIC (Q100/Q200): Performs packet inspection and hash calculation as directed by configuration.
- Network administrator: Chooses the hash field mode to optimize NVGRE deployment outcomes.
- Forwarding engine: Uses the calculated hash to make load-balancing decisions for NVGRE traffic.

Workflow

These stages describe how a router with a Cisco Silicon One Q100 or Q200 ASIC-based system processes NVGRE traffic and selects header fields for load-balancing hash calculations:

1. The router receives an incoming packet and determines whether it is an NVGRE-encapsulated packet by checking the protocol value.
   - Packet arrives at the device ingress interface.
   - ASIC inspects the protocol value for `0x6558` (NVGRE).

   This identification is necessary to apply the NVGRE-specific hash field selection logic.

   Once the NVGRE packet is detected, the device prepares the hash selection procedure as configured for NVGRE traffic.
2. The ASIC checks the device configuration to determine which headers should be used for hash calculation on NVGRE packets.

- Device configuration is queried for the presence of `cef platform load-balancing nvgre payload exclude`.

Configuration specifies whether only the NVGRE outer header or both outer and inner headers will participate in hashing. This is a static setting, chosen for deployment needs.

No hash calculation occurs until the mode is identified.

3. The ASIC performs the hash calculation according to the selected mode:
   - If `nvgre payload exclude` is enabled: Only the outer header fields (such as source and destination IP) are included in the hash.
   - If `nvgre payload exclude` is disabled (default): Both outer and inner header fields (including inner MAC or IP) are included.

   This stage determines load-balancing accuracy, especially when handling non-IP payloads.
4. The forwarding engine applies the computed hash to select the egress path for the packet.
   - Hash value is used by ECMP or port-channel logic to assign the packet to a specific link.

   The outcome is consistent forwarding of NVGRE flows according to deployment requirements and traffic type.

   Traffic forwarding completes based on the device's hash selection, ensuring predictable NVGRE load-balancing behavior.

Result

The process enables deterministic, user-directed load-balancing for NVGRE traffic on Cisco Q100 and Q200 ASIC platforms.

Configure NVGRE hash field selection

This topic describes how to configure NVGRE hash field selection on the Cisco 8000 Series Router to control the load-balancing behavior for NVGRE traffic.

- NVGRE hash field selection is relevant primarily to transit devices that do not decapsulate NVGRE traffic.
- Only applicable to Cisco Silicon One Q100 and Q200 ASIC-based systems.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Access the global configuration prompt to apply platform-wide settings. Ensure you have the necessary privilege level for configuration changes.
2. Set NVGRE load-balancing to use only the outer header, and exclude inner payload fields.

   Example

   ```
   Router(config)# cef platform load-balancing nvgre payload exclude
   ```

This command configures the ASIC to ignore inner payload fields for NVGRE traffic when calculating the load-balancing hash, ensuring predictable balancing especially with non-IP NVGRE payloads.

The router now hashes only outer IP headers for NVGRE traffic when making load-balancing decisions.

3. Save the configuration.

Example

```
Router(config)# commit
Router(config)# end
```

17 802.1Q VLAN interfaces

Topics:

This chapter describes configuring 802.1Q VLAN interfaces, subinterfaces, and attachment circuits for Cisco 8000 Series Routers.

Prerequisites for configuring 802.1Q VLAN interfaces

This topic lists the prerequisites that you must meet before you configure 802.1Q VLAN interfaces on the Cisco 8000 Series Router.

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

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

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

802.1Q VLAN

This topic describes 802.1Q VLANs on the Cisco 8000 Series Router, including how the IEEE 802.1Q protocol standard uses VLAN tags to divide networks into logically separated broadcast domains.

VLAN is a logical network configuration that

- enables devices on different physical LAN segments to communicate as if they share a single wire, when in fact, they are located on several different LAN segments
- provides flexibility for user and host management as they are based on logical instead of physical connections, and
- optimizes bandwidth allocation and resource usage.

Table 32: Feature History Table

Feature Name	Release Information	Feature Description
802.1Q VLAN	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
802.1Q VLAN	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: - 8011-32Y8L2H2FH - 8011-12G12X4Y-A/D
802.1Q VLAN	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
802.1Q VLAN	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The 802.1Q VLAN functionality is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
802.1Q VLAN	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) * The 802.1Q VLAN functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-36EH+A8:B12 • 88-LC1-12TH24FH-E
802.1Q VLAN	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) A VLAN is a logical grouping of devices across different LAN segments that communicate as if they are on the same physical network, offering flexibility in user management, bandwidth allocation, and resource optimization. The IEEE 802.1Q protocol standard helps divide large networks into smaller segments to efficiently manage broadcast and multicast traffic, enhancing bandwidth usage and security. * This feature is now supported on routers with 88-LC1-36EH line cards.
Double-Tagged 802.1ad Encapsulation Options for Layer 3 Physical and Bundle Subinterfaces	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) The support for Double-Tagged 802.1ad Encapsulation Options for Layer 3 Physical and Bundle Subinterfaces is now extended to all Systems in the Cisco 8000 Series Routers.
Double-Tagged 802.1ad Encapsulation Options for Layer 3 Physical and Bundle Subinterfaces	Release 7.3.2	This feature enables you to increase the number of VLAN tags in an interface and an subinterface. You can enable this feature either on a physical interface or a bundle interface. When you configure this feature with the dual tag, interfaces check for IP addresses along with MAC addresses. Verified Scalability Limits: • Total number of VLAN tags in an interface and a subinterface with single tag: 4094 • Total number of VLAN tags in an interface and a subinterface with double tag: 4094 * 4094

The IEEE 802.1Q protocol standard provides several benefits for network management:

- It divides large networks into smaller segments to reduce unnecessary broadcast and multicast traffic.
- It enhances security between internal network segments.
- It establishes a standard method for inserting VLAN membership information into Ethernet frames.

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

802.1Q tagged frames

This topic describes IEEE 802.1Q tag-based VLANs on the Cisco 8000 Series Router, including how the tag is inserted into the Ethernet frame and the components that switches process across the network.

The IEEE 802.1Q tag-based VLAN is a network configuration method that

- uses an extra tag in the MAC header to identify frame membership across bridges
- supports both static and dynamic VLAN creation, and
- facilitates VLAN and Quality of Service (QoS) priority identification.

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

The IEEE 802.1Q frame structure includes specific components that switches process across the network:

- The VLAN ID associates a frame with a specific VLAN.
- A tagged frame measures four bytes longer than an untagged frame.
- The Tag Protocol Identifier (TPID) resides within the type and length field of the Ethernet frame.
- The Tag Control Information (TCI) starts after the source address field of the Ethernet frame.

Subinterfaces

This topic describes subinterfaces on the Cisco 8000 Series Router, which are logical interfaces created on a hardware interface to segregate traffic into separate logical channels and better utilize the available bandwidth.

Subinterfaces, created on a hardware interface, are logical interfaces that

- allows the segregation of traffic into separate logical channels on a single hardware interface, and
- allows for the better utilization of the available bandwidth on the physical interface.

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

Before the system allows a subinterface to pass traffic, it must have a valid tagging protocol encapsulation and VLAN identifier assigned. All Ethernet subinterfaces always default to the 802.1Q VLAN encapsulation. However, you must explicitly define the VLAN identifier.

Supported encapsulation

Table 33: 802.1ad encapsulation support for Layer 3 interfaces and subinterfaces

Interface Type	Encapsulation	Standard	Support Status
Layer 3 interface	Single-Tag Encapsulation	dot1ad	Supported (From Cisco IOS XR Software Release 24.4.1 onwards)
Layer 3 subinterface		dot1q	Supported.
Layer 3 bundle subinterface	Double-Tag Encapsulation	dot1ad <> dot1q<>	Supported.
		dot1q <> dot1q<>	[1]Supported.

For information about supported encapsulation for Layer 2 interfaces and subinterfaces, see Virtual LANs in Layer 2 VPNs.

Subinterface MTU

This topic describes how a subinterface on the Cisco 8000 Series Router inherits its maximum transmission unit (MTU) from the parent physical interface, with an additional four bytes allowed for the 802.1Q VLAN tag.

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

Native VLAN

This topic describes how the Cisco 8000 Series Router provides the equivalent of native VLAN functionality.

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

encapsulation dot1q TAG-ID

Layer 2 VPN on VLANs

This topic describes how Layer 2 Virtual Private Network (L2VPN) on VLANs works on the Cisco 8000 Series Router, and how to configure VLAN attachment circuits (ACs) for L2VPN services.

The Layer 2 Virtual Private Network (L2VPN) feature enables Service Providers (SPs) to provide Layer 2 services to geographically disparate customer sites.

The configuration model for configuring VLAN attachment circuits (ACs) is similar to the model used for configuring basic VLANs, where the user first creates a VLAN subinterface, and then configures that VLAN in subinterface configuration mode. To create an AC, you need to include the **l2transport** keyword in the **interface** command string to specify that the interface is a Layer 2 interface.

VLAN ACs support three modes of L2VPN operation:

- Basic Dot1Q AC—The AC covers all frames that are received and sent with a specific VLAN tag.

[1] The **encapsulation dot1q <x> second-dot1q <y>** encapsulation type is supported on Q200-based line cards from Cisco IOS XR Software Release 24.1.1 onwards and supported for all hardware platforms in the Cisco 8000 Series Routers from Cisco IOS XR Software Release 24.4.1 onwards.

- QinQ AC– Only outer tag (s-tag) of 0x88a8 and inner tag (c-tag) of 0x8100 is supported.

Cisco IOS XR software supports 255 ACs per LC when you configure L2VPN on a VLAN.

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

Configure 802.1Q VLAN subinterfaces

This topic describes how to configure 802.1Q VLAN subinterfaces on the Cisco 8000 Series Router by setting the Layer 2 encapsulation and assigning an IP address to the subinterface.

Use this procedure to configure 802.1Q VLAN subinterfaces. To remove these subinterfaces, see Remove an 802.1Q VLAN subinterface.

Note

- You can programmatically configure and retrieve the VLAN interfaces and subinterfaces parameters using `openconfig-vlan.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.
- The OpenConfig VLAN data model uses structured `vlan/match` containers to represent VLAN matching. The `vlan-id` leaf supports only a single VLAN value and does not represent VLAN lists or ranges. See the OpenConfig model documentation for details.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter subinterface configuration mode and specify the interface type, location, and subinterface number.

   Example

   ```
   Router(config)# interface TenGigE 0/2/0/4.10
   ```

   - Replace the *interface-path-id* argument with one of the following instances:
   - Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack*/*slot*/*module*/*port*, and a slash between values is required as part of the notation.
   - Ethernet bundle instance. Range is from 1 through 65535.
   - Replace the *subinterface* argument with the subinterface value. Range is from 0 through 4095.
   - Naming notation is *interface-path-id.subinterface*, and a period between arguments is required as part of the notation.

3. Set the Layer 2 encapsulation of the interface.

Example

```
Router(config-subif)# encapsulation dot1q 100
```

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

4. Assign an IP address and subnet mask to the subinterface.

Example

```
Router(config-subif)# ipv4 address 178.18.169.23/24
```

- 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.

5. Save the configuration.

Example

```
Router(config-subif)# exit
Router(config)# commit
```

Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session. When you issue the **end** command, the system prompts you to commit changes.

6. Verify the configuration using the **show interfaces** command.

Example

```
Router# show interfaces fourHundredGigE 0/5/0/1.100 brief

                Intf        Intf        LineP              Encap  MTU
BW
                Name        State       State               Type  (byte)
(Kbps)
--------------------------------------------------------------------------------

      FH0/5/0/1.100         up        up             802.1Q  1518  400000000
```

```
Router# show interfaces brief location 0/5/CPU0 | include 802.1Q
      FH0/5/0/1.100        up        up             802.1Q  1518  400000000
```

Configure an attachment circuit on a VLAN

This topic describes how to configure a Layer 2 attachment circuit (AC) on a VLAN on the Cisco 8000 Series Router.

Use this procedure to configure an attachment circuit on a VLAN.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure terminal
   ```

2. Enter subinterface configuration mode and specify the interface type, location, and subinterface number.

   Example

   ```
   Router(config)# interface TenGigE 0/1/0/0.1 l2transport
   ```

   - Replace the argument with one of the following instances:
   - Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack*/*slot*/*module*/*port*, and a slash between values is required as part of the notation.
   - Ethernet bundle instance. Range is from 1 through 65535.
   - Replace the *subinterface* argument with the subinterface value. Range is from 0 through 4095.
   - Naming notation is *instance.subinterface*, and a period between arguments is required as part of the notation.
   - You must include the **l2transport** keyword in the command string; otherwise, the configuration creates a Layer 3 subinterface rather than an AC.

3. Set the Layer 2 encapsulation of the interface.

   Example

   ```
   Router(config-subif)# encapsulation dot1q 100
   ```

4. Save the configuration.

   Example

   ```
   Router(config-if-l2)# commit
   ```

   Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session. When you issue the **end** command, the system prompts you to commit changes.

5. (Optional) Display statistics for interfaces on the router.

Example

```
Router# show interfaces TenGigE 0/3/0/0.1
```

Remove an 802.1Q VLAN subinterface

This topic describes how to remove a previously configured 802.1Q VLAN subinterface on the Cisco 8000 Series Router.

Use this procedure to remove 802.1Q VLAN subinterfaces that have been previously configured using the Configure 802.1Q VLAN subinterfaces task in this module.

Procedure

1. Enter global configuration mode.

Example

```
Router# configure
```

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

Example

```
Router(config)# no interface TenGigE 0/2/0/4.10
```

- Replace the *instance* argument with one of the following instances:
- Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack*/*slot*/*module*/*port*, and a slash between values is required as part of the notation.
- Ethernet bundle instance. Range is from 1 through 65535.
- Replace the *subinterface* argument with the subinterface value. Range is from 0 through 4095.

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

> **Note**
>
> Repeat this step to remove other VLAN subinterfaces.

3. Save the configuration.

Example

```
Router(config)# commit
```

Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session. When you issue the **end** command, the system prompts you to commit changes.

VLAN subinterfaces example

This topic provides configuration examples that show how to create VLAN subinterfaces on the Cisco 8000 Series Router, including subinterfaces on a physical Ethernet interface and on an Ethernet bundle.

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

```
Router# configure
Router(config)# interface TenGigE 0/2/0/4.1
Router(config-subif)# encapsulation dot1q 100
Router(config-subif)# ipv4 address 10.0.10.1/24
Router(config-subif)# interface TenGigE0/2/0/4.2
Router(config-subif)# encapsulation dot1q 101
Router(config-subif)# ipv4 address 10.0.20.1/24
Router(config-subif)# interface TenGigE0/2/0/4.3
Router(config-subif)# encapsulation dot1q 102
Router(config-subif)# ipv4 address 10.0.30.1/24
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# exit

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

Router# show vlan interface
Interface     St Ly   MTU   Subs   L3       Up   Down  Ad-Down
Te0/2/0/4.1         802.1Q              10  up
Te0/2/0/4.2         802.1Q              20  up
Te0/2/0/4.3         802.1Q              30  up

Router# show vlan trunks brief
BE1           Up L3   1514    1000        0    1000     1000       0       0
Summary                       1000        0    1000     1000       0       0
Te0/2/0/4           802.1Q (Ether)        up
```

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

```
Router# configure
Router(config)# interface bundle-ether 2
Router(config-if)# ipv4 address 192.168.2.1/24
Router(config-if)# exit
Router(config)# interface bundle-ether 2.1
Router(config-subif)# encapsulation dot1q 100
Router(config-subif)# ipv4 address 192.168.100.1/24
Router(config-subif)# exit
Router(config)# interface bundle-ether 2.2
Router(config-subif)# encapsulation dot1q 200
Router(config-subif)# ipv4 address 192.168.200.1/24
```

```
Router(config-subif)# exit
Router(config)# commit
```

Layer 2 interface VLAN encapsulation using VLAN ranges and lists

This topic describes Layer 2 interface VLAN encapsulation using VLAN ranges and lists on the Cisco 8000 Series Router, which lets you group multiple VLAN IDs together for encapsulation to simplify configuration and improve scalability.

VLAN encapsulation is a networking technique that:

- embeds VLAN tags within Ethernet frames to distinguish and identify network traffic
- relies on unique VLAN IDs, organized through ranges and lists, to manage and distinguish different VLANs, and
- ensures logical separation of traffic types to enhance network performance, security, and manageability.

Table 34: Feature History Table

Feature Name	Release Information	Feature Description
Layer 2 interface VLAN encapsulation using VLAN ranges and lists	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Layer 2 interface VLAN encapsulation using VLAN ranges and lists	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Layer 2 interface VLAN encapsulation using VLAN ranges and lists	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.

Layer 2 interface VLAN encapsulation using VLAN ranges and lists	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700 [ASIC: K100])(select variants only*); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q200, P100])(select variants only*) You can now leverage the VLAN ranges and lists to effectively separate networks operating over shared links and devices. VLAN encapsulation is typically determined by the access network and customer edge (CE) device, limiting the network provider's control over the VLAN tag or Ethernet type of customer traffic. The VLAN ranges and lists support various customer traffic types, enhancing network flexibility and management. *This feature is supported on: • 8201-32FH • 8201-24H8FH • 8202-32FH-M • 8212-48FH-M • 8608 • 8711-32FH-M • 8712-MOD-M • 88-LC0-34H14FH • 88-LC0-36FH • 88-LC0-36FH-M • 88-LC1-36EH • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E This feature modifies these changes: CLI: • **encapsulation dot1ad** • **encapsulation dot1ad dot1q** • **encapsulation dot1q** • **encapsulation dot1q second-dot1q**

VLAN ranges and lists

VLAN ranges and lists differ in their approach to VLAN encapsulation:

- A VLAN range is a sequence of contiguous VLAN IDs used for encapsulation. For instance, VLAN IDs from 200 to 250 can be grouped in a range.

- A VLAN list is a specific enumeration of VLAN IDs used for encapsulation. For instance, VLAN IDs 100, 123, and 150 can be grouped in a list.

VLAN list and range can be used together in configuration.

Benefits of Layer 2 interface VLAN encapsulation using VLAN ranges and lists

The VLAN encapsulation using VLAN ranges and lists offers several key benefits and is crucial for efficient network management:

- Simplified configuration: Configuring multiple VLANs with a single command, rather than individually, reduces the time and effort required for network setup and modification.
- Enhanced manageability: Managing VLANs becomes easier when grouped into lists or ranges. Changes can be applied to multiple VLANs simultaneously, simplifying network management tasks.
- Improved scalability: Grouping VLANs supports more efficient scaling in large-scale network environments, enabling the network to handle more VLANs without significantly increasing configuration complexity. A single subinterface can be used instead of multiple subinterfaces.
- Reduced configuration errors: Using fewer commands to configure multiple VLANs decreases the likelihood of configuration errors, resulting in a more stable and reliable network setup.

Supported encapsulations

- encapsulation dot1q <x> , <y> (For example, encapsulation dot1q 200, 210, 300, 310, 400)
- encapsulation dot1q <x> - <y> (For example, encapsulation dot1q 200 – 400, 600-801, 901-1000)
- encapsulation dot1q <x> second-dot1q <y>, <z>
- encapsulation dot1q <x> second-dot1q <y> - <z>
- encapsulation dot1q <x> - <y> second-dot1q <z>
- encapsulation dot1ad <x> dot1q <y>, <z>
- encapsulation dot1ad <x> dot1q <y> - <z>
- encapsulation dot1ad <x> - <y> dot1q <z>
- encapsulation dot1ad <x>, <y> dot1q <z>
- encapsulation dot1ad <x>, <y>
- encapsulation dot1ad <x> - <y>

Restrictions for Layer 2 interface VLAN encapsulation using VLAN ranges and lists

This topic lists the restrictions that you must observe when you configure Layer 2 interface VLAN encapsulation using VLAN ranges and lists on the Cisco 8000 Series Router.

These are the restrictions of Layer 2 interface VLAN encapsulation using VLAN ranges and lists:

- Supported only on Q200, P100, and K100 line cards and systems.
- Double-tagged encapsulations support the inner or outer tag as a list or range of VLAN IDs, but not both. For example:
  - encapsulation dot1ad 100-200 dot1q 300 (supported)
  - encapsulation dot1ad 100 dot1q 200,300 (supported)
  - encapsulation dot1ad 10-20 dot1q 30-40 (not supported)

- Supported only on Layer 2 subinterfaces.
- The maximum number of list entries, individual or range per encapsulation, is 64.
- Ranges of up to 4094 VLAN IDs are supported.
- For lists with more than ten list items, use the **encapsulation list-extended** command.

  For example, **encapsulation list-extended dot1q <x1>, … , <x10>**

Configure a VLAN range and a VLAN list on Layer 2 subinterfaces

This topic describes how to configure a VLAN range and a VLAN list on Layer 2 subinterfaces on the Cisco 8000 Series Router by using the appropriate encapsulation method.

Define the matching criteria to map VLAN-tagged frames to the appropriate service instance.

Procedure

1. Configure a VLAN range and a VLAN list on Layer 2 subinterfaces using the appropriate encapsulation method based on your requirements.
   - If you want to configure a service with a list of noncontiguous customer VLAN (C-VLAN) ranges:

     ```
     Router# configure
     Router(config)# interface HundredGigE0/3/0/0.1 l2transport
     Router(config-subif)# encapsulation dot1ad 200-400,600-801
     Router(config-subif)# commit
     ```

   - If you want to configure a service with a list of C-VLANs:

     ```
     Router# configure
     Router(config)# interface HundredGigE0/2/0/0.1 l2transport
     Router(config-subif)# encapsulation dot1q 200,210,300,310,400
     Router(config-subif)# commit
     ```

   - If you want to configure a service with a range of service provider VLANs (S-VLANs) and a single inner C-VLAN:

     ```
     Router# configure
     Router(config)# interface HundredGigE0/4/0/0.1 l2transport
     Router(config-subif)# encapsulation dot1ad 100-200 dot1q 300
     Router(config-subif)# commit
     ```

   - If you want to configure a QinQ service with a single S-VLAN and a list of C-VLANs:

     ```
     Router# configure
     Router(config)# interface HundredGigE0/5/0/0.1 l2transport
     Router(config-subif)# encapsulation dot1q 100 second-dot1q 200,300
     Router(config-subif)# commit
     ```

2. Run the **show interfaces** command to verify the VLAN range and VLAN list on the Layer 2 subinterface.

   The corresponding value of the outer match appears depending on the VLAN configuration. The *Outer Match* field corresponds to the Dot1ad VLAN range and list in this example.

   ```
   Router# show interfaces HundredGigE0/3/0/0.1
   HundredGigE0/3/0/0.1 is UP, line protocol is UP
   Interface state transitions: 0
   ```

```
Hardware is VLAN sub-interface(s), address is dc05.39c7.9440
Layer 2 Transport Mode
MTU 1518 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
reliability Unknown, txload Unknown, rxload Unknown
Encapsulation 802.1ad Virtual LAN,
Outer Match: Dot1ad VLAN 200-400,600-801
Ethertype Any, MAC Match src any, dest any
loopback not set,
Last input never, output never
Last clearing of "show interface" counters never
0 packets input, 0 bytes
0 input drops, 0 queue drops, 0 input errors
0 packets output, 0 bytes
0 output drops, 0 queue drops, 0 output errors
```

In this example, the *Outer Match* and *Inner Match* fields correspond to the Dot1ad VLAN range 100-200 and VLAN list Dot1Q VLAN 300.

```
Router# show interfaces HundredGigE0/4/0/0.1

HundredGigE0/4/0/0.1 is UP, line protocol is UP
Interface state transitions: 0
Hardware is VLAN sub-interface(s), address is dc05.39c7.9440
Layer 2 Transport Mode
MTU 1522 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
reliability Unknown, txload Unknown, rxload Unknown
Encapsulation 802.1ad-802.1Q Virtual LAN,
Outer Match: Dot1ad VLAN 100-200
Inner Match: Dot1Q VLAN 300
Ethertype Any, MAC Match src any, dest any
loopback not set,
Last input never, output never
Last clearing of "show interface" counters never
0 packets input, 0 bytes
0 input drops, 0 queue drops, 0 input errors
0 packets output, 0 bytes
0 output drops, 0 queue drops, 0 output errors
```

18 IP-in-IP tunnels

Topics:

This chapter describes configuring IP-in-IP tunnels, including encapsulation, decapsulation, TTL management, and load balancing.

IP-in-IP tunnels

This topic describes IP-in-IP tunneling on the Cisco 8000 Series Router, which encapsulates one IP packet as a payload within another IP packet to transport traffic across an intermediate IP network.

IP-in-IP tunneling is a mechanism that

- provides transport of packets of one protocol within another protocol through encapsulation
- encapsulates and decapsulates an IP packet as a payload in another IP packet, and
- supports all four combinations of inner and outer IP versions: IPv4 over IPv4, IPv6 over IPv4, IPv4 over IPv6, and IPv6 over IPv6.

For example, an IPv4 over IPv6 refers to an IPv4 packet as a payload encapsulated within an IPv6 packet and routed across an IPv6 network to reach the destination IPv4 network, where it is decapsulated.

Table 35: Feature History Table

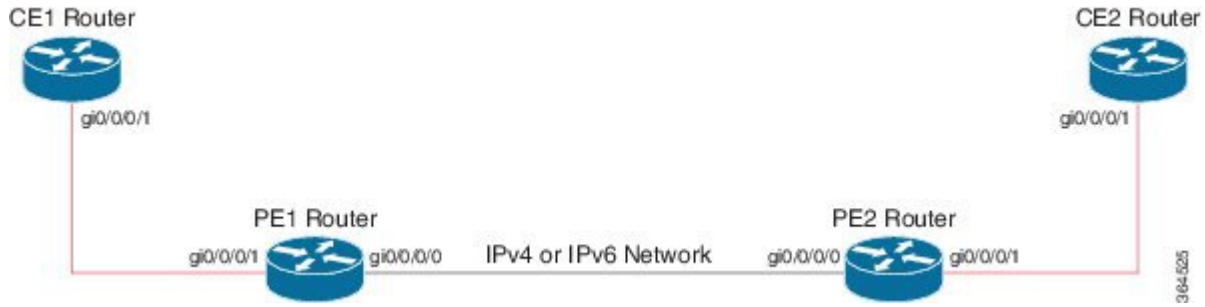
Feature Name	Release Information	Feature Description
IPv4 packets with IPv6 outer header	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
IPv4 packets with IPv6 outer header	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is now supported on: - 8711-48Z-M - 8011-32Y8L2H2FH - 8011-12G12X4Y-A/D
Increased IP-in-IP tunnel scale for gRIBI-based nexthops	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q200]) This feature enables the router to efficiently load balance and manage high traffic volumes by increasing the scale for encapsulation and decapsulation nexthops programmed using gRPC Routing Information Base Interface (gRIBI) to 12K IP-in-IP tunnels. This feature introduces these changes: CLI: The **hw-module profile cef iptunnel scale** command is modified.

Feature Name	Release Information	Feature Description
Increase in IP-in-IP decapsulation tunnels support	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200]; Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q200]) With this release, we have revised the maximum number of IPv4 and IPv6 IP-in-IP decapsulation tunnels from 64 to 200 on Cisco Silicon One Q200 ASIC-based systems. An increased number enhances the router's ability to support larger and more complex IP-in-IP tunneling scenarios improving scalability, efficiency, and flexibility in network design.
IPv4 packets with IPv6 outer header	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
IPv4 packets with IPv6 outer header	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700)(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*). This feature that allows decapsulation of IPv4 and IPv6 tunnels with IPv6 headers helps the administrators to benefit from an improved IPv6 routing and security without upgrading their entire network to IPv6. *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 88-LC1-36EH
IPv4 packets with IPv6 outer header	Release 7.5.3	With this release, decapsulation of IPv4 and IPv6 tunnels with IPv6 outer headers are supported. This feature helps the administrators to take advantage of the benefits of IPv6, such as improved routing and security, without having to upgrade their entire network to IPv6.

IP-in-IP tunneling can be used to connect remote networks securely or provide virtual private network (VPN) services.

This simplified network topology provides the transport VRF as the default VRF for an IPv4 or IPv6 network.

Figure 13: IP-in-IP tunnel network topology



A maximum of 3500 IP-in-IP tunnels are supported until Cisco IOS XR Release 25.4.1. From Cisco IOS XR Release 25.4.1, you can use the **hw-module profile cef iptunnel scale** command to configure up to 12,000 IP-in-IP tunnels that are shared between the gRIBI-based encapsulation and decapsulation next-hop tunnels. For more information, see Configure improved scale for IP-in-IP tunnels.

Restrictions for IP-in-IP tunnel configuration

This topic lists the restrictions that you must observe when you configure IP-in-IP tunnels on the Cisco 8000 Series Router.

Observe the following restrictions when you configure IP-in-IP tunnels:

- The feature does not support decapsulation tunnels on subinterfaces.
- Only the default Virtual Routing and Forwarding (VRF) instance is supported.
- IPv6 link local addresses are not supported.
- Regular tunnels cannot use a configured IP address as the tunnel source; only a non-existent IP address can be used.
- Configuring multiple interfaces with the same IP address is not supported.
- Each line card can have different number of Network Processor (NP) slices.
- The maximum IPv4 and IPv6 IP-in-IP decapsulation tunnels supported is 64 per slice.
- From Cisco IOS XR Release 25.3.1 onwards, the maximum IPv4 and IPv6 IP-in-IP decapsulation tunnels supported is 200 per slice.

Configure improved scale for IP-in-IP tunnels

This topic describes how to configure the improved IP-in-IP tunnel scale to support up to 12,000 tunnels for gRIBI-based encapsulation and decapsulation next-hops on the Cisco 8000 Series Router.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter the **hw-module profile cef iptunnel scale** command to configure the maximum IP-in-IP tunnel scale to 12K for gRIBI-based encapsulation and decapsulation next-hops.

Example

```
Router(config)# hw-module profile cef iptunnel scale
```

3. Save the configuration and exit the configuration mode.

Example

```
Router(config)# commit
Router(config)# exit
```

4. Reload the line cards with the **reload location all** command to enable the increased IP-in-IP scale limit.

Example

```
Router# reload location all
```

5. Execute the **show hw-module profile cef** command to view the configuration status of CEF hardware modules.

Example

```
Router# show hw-module profile cef

Fri Aug 22 13:23:17.796 UTC
-----------------------------------------------------------------
Knob                          Status          Applied   Action
-----------------------------------------------------------------
CBF Enable                    Unconfigured    N/A       None
CBF forward-class-list        Unconfigured    N/A       None
BGPLU                         Unconfigured    N/A       None
LPTS ACL                      Unconfigured    N/A       None
Dark Bandwidth                Unconfigured    N/A       None
SR-OPT                        Unconfigured    N/A       None
IP Redirect Punt              Unconfigured    N/A       None
IPv6 Hop-limit Punt           Unconfigured    N/A       None
MPLS Per Path Stats           Unconfigured    N/A       None
SRv6 Per Path Stats           Unconfigured    N/A       None
Tunnel TTL Decrement          Unconfigured    N/A       None
High-Scale No-LDP-Over-TE     Unconfigured    N/A       None
Label over TE counters        Unconfigured    N/A       None
Highscale LDPoTE No SRoTE     Unconfigured    N/A       None
LPTS Pifib Entry Counters     Unconfigured    N/A       None
Unipath surpf                 Unconfigured    N/A       None
Source-based rtbh             Unconfigured    N/A       None
Vxlan ipv6 tunnel scale       Unconfigured    N/A       None
Encap Exact                   Unconfigured    N/A       None
Iptunnel scale                Configured      Yes       None
BGPLU over RSVPTE Enable      Unconfigured    N/A       None
MPLS Decap Stats              Unconfigured    N/A       None
```

Configuration example for IPv4 tunnel

This topic provides a configuration example for an IPv4 IP-in-IP tunnel between PE and CE routers on the Cisco 8000 Series Router.

This example provides the configuration for an IPv4 tunnel.

Table 36: PE1 and PE2 router configuration

PE1 router configuration	PE2 router configuration

`interface GigabitEthernet0/0/0/0`
` !! Link between PE1-PE2`
` ipv4 address 100.1.1.1/24`
`!`
`interface GigabitEthernet0/0/0/1`
` !! Link between CE1-PE1`
` ipv4 address 20.1.1.1/24`
` ipv6 address 20::1/64`
`!`
`interface tunnel-ip 1`
` ipv4 address 10.1.1.1/24`
` ipv6 address 10::1/64`
` tunnel mode ipv4`
` tunnel source GigabitEthernet0/0/0/0`
 	
` tunnel destination 100.1.1.2`
`!`
`router static`
` address-family ipv4 unicast`
`  30.1.1.0/24 tunnel-ip1`
` address-family ipv6 unicast`
`  30::0/64 tunnel-ip1`
` !`
`!`
	`interface GigabitEthernet0/0/0/0`
	` !! Link between PE1-PE2`
	` ipv4 address 100.1.1.2/24`
	`!`
	`interface GigabitEthernet0/0/0/1`
	` !! Link between PE2-CE2`
	` ipv4 address 30.1.1.1/24`
	` ipv6 address 30::1/64`
	`!`
	`interface tunnel-ip 1`
	` ipv4 address 10.1.1.2/24`
	` ipv6 address 10::2/64`
	` tunnel mode ipv4`
	` tunnel source GigabitEthernet0/0/0/0`
	
	` tunnel destination 100.1.1.1`
	`!`
	`router static`
	` address-family ipv4 unicast`
	`  20.1.1.0/24 tunnel-ip1`
	` address-family ipv6 unicast`
	`  20::0/64 tunnel-ip1`
	` !`
	`!`

Table 37: CE1 and CE2 router configuration

CE1 router configuration	CE2 router configuration

`interface GigabitEthernet0/0/0/1`
` !! Link between CE1-PE1`
` ipv4 address 20.1.1.2 255.255.255.0`
` ipv6 address 20::2/64`
`!`
`router static`
` address-family ipv4 unicast`
`  30.1.1.0/24 20.1.1.1`
` address-family ipv6 unicast`
`  30::0/64 20::1`
` !`
`!`
	`interface GigabitEthernet0/0/0/1`
	` !! Link between CE2-PE2`
	` ipv4 address 30.1.1.2 255.255.255.0`
	` ipv6 address 30::2/64`
	`!`
	`router static`
	` address-family ipv4 unicast`
	`  20.1.1.0/24 30.1.1.1`
	` address-family ipv6 unicast`
	`  20::0/64 30::1`
	` !`
	`!`

Configuration example for IPv6 tunnel

This topic provides a configuration example for an IPv6 IP-in-IP tunnel between PE and CE routers on the Cisco 8000 Series Router.

This example provides the configuration for an IPv6 tunnel.

Table 38: PE1 and PE2 router configuration

PE1 router configuration	PE2 router configuration
interface GigabitEthernet0/0/0/0 **!! Link between PE1-PE2** **ipv6 address 100::1/64** **!** **interface GigabitEthernet0/0/0/1** **!! Link between CE1-PE1** **vrf RED** **ipv4 address 20.1.1.1/24** **ipv6 address 20::1/64** **!** **interface tunnel-ip 1** **vrf RED** **ipv4 address 10.1.1.1/24** **ipv6 address 10::1/64** **tunnel mode ipv6** **tunnel source GigabitEthernet0/0/0/0** **tunnel destination 100::2** **!** **vrf RED** **address-family ipv6 unicast** **import route-target** **2:1** **!** **export route-target** **2:1** **!** **address-family ipv4 unicast** **import route-target** **2:1** **!** **export route-target** **2:1** **!** **router static** **vrf RED** **address-family ipv4 unicast** **30.1.1.0/24 tunnel-ip1** **address-family ipv6 unicast** **30::0/64 tunnel-ip1** **!** **!** **!**	interface GigabitEthernet0/0/0/0 !! Link between PE1-PE2 ipv6 address 100::2/64 ! interface GigabitEthernet0/0/0/1 !! Link between PE2-CE2 vrf RED ipv4 address 30.1.1.1/24 ipv6 address 30::1/64 ! interface tunnel-ip 1 vrf RED ipv4 address 10.1.1.2/24 ipv6 address 10::2/64 tunnel mode ipv6 tunnel source GigabitEthernet0/0/0/0 tunnel destination 100::1 ! vrf RED address-family ipv6 unicast import route-target 2:1 ! export route-target 2:1 ! address-family ipv4 unicast import route-target 2:1 ! export route-target 2:1 ! router static vrf RED address-family ipv4 unicast 20.1.1.0/24 tunnel-ip1 address-family ipv6 unicast 20::0/64 tunnel-ip1 ! ! !

Table 39: CE1 and CE2 router configuration

CE1 router configuration	CE2 router configuration

CE1 router configuration	CE2 router configuration

`interface GigabitEthernet0/0/0/1`
` !! Link between CE1-PE1`
` ipv4 address 20.1.1.2 255.255.255.0`
` ipv6 address 20::2/64`
`!`
`router static`
` address-family ipv4 unicast`
`  30.1.1.0/24 20.1.1.1`
` address-family ipv6 unicast`
`  30::0/64 20::1`
` !`
`!`
	`interface GigabitEthernet0/0/0/1`
	` !! Link between CE2-PE2`
	` ipv4 address 30.1.1.2 255.255.255.0`
	` ipv6 address 30::2/64`
	`!`
	`router static`
	` address-family ipv4 unicast`
	`  20.1.1.0/24 30.1.1.1`
	` address-family ipv6 unicast`
	`  20::0/64 30::1`
	` !`
	`!`

Control the TTL value of inner payload header

This topic describes how the Cisco 8000 Series Router controls the TTL value of the inner payload header of IP-in-IP tunnel packets before the packets are forwarded to the next-hop router.

The IP-in-IP tunnel configurations are network settings that

- control the TTL value of inner payload headers
- enable a router to forward custom-formed IP-in-IP stacked packets even if the inner packet TTL is 1, and
- facilitate end-to-end link-state and path reachability measurements.

The Cisco 8000 Series Router maintains specific operational behaviors for IP-in-IP tunnels:

- After you enable or disable the decrement of the TTL value of the inner payload header of a packet, you do not need to reload the line card.
- The router allows you to control the TTL value of the inner payload header of IP-in-IP tunnel packets before forwarding them to the next-hop router.
- The router supports a maximum of 16 IP-in-IP decapsulation tunnels with unique source addresses starting from Cisco IOS XR Release 7.3.3.
- The system updates the TTL decrement setting without requiring a line card reload.
- The OOR State displays *Red* in the **show controllers npu resources sipidxtbl location all** command output.

Disable inner payload header TTL decrement

This topic describes how to disable the decrement of the TTL value of the inner payload header of an IP-in-IP packet on the Cisco 8000 Series Router.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Disable the decrement of the TTL value of the inner payload header of an IP-in-IP packet with the **hw-module profile cef ttl tunnel-ip decrement disable** command.

   Example

```
Router(config)# hw-module profile cef ttl tunnel-ip decrement disable
Router(config)# commit
```

Time-to-Live uniform mode

This topic describes Time-to-Live (TTL) uniform mode on the Cisco 8000 Series Router, a mechanism that synchronizes TTL values between inner and outer packet headers during encapsulation and decapsulation.

Time-to-Live (TTL) uniform mode is a mechanism that

- ensures consistent TTL management by synchronizing the TTL values between inner and outer packet headers during encapsulation and decapsulation, allowing the receiving device to accurately interpret the packet's remaining lifespan
- allows you to copy the TTL values from inner headers to outer headers during encapsulation and from outer headers to inner headers during decapsulation, and
- ensures consistent TTL management across various network scenarios.

For more information on the various network scenarios, see Use cases for TTL uniform mode on a router.

Table 40: Feature History Table

Feature Name	Release Information	Feature Description
Copy TTL value to IP headers	Release 25.2.1	Introduced in this release on: Fixed Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on the Cisco 8404-SYS-D router.
Copy TTL value to IP headers	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200]; Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q200]) Support for Time-to-Live (TTL) uniform mode is introduced. This mode ensures consistent TTL management by synchronizing the TTL values between inner and outer packet headers during encapsulation and decapsulation, allowing the receiving device to accurately interpret the packet's remaining lifespan. TTL uniform mode is enabled only for the **pbr vrf-redirect** mode in IP-in-IP tunnels.

Enabling TTL uniform mode offers these advantages:

- **Enhanced packet integrity and lifespan accuracy**: TTL uniform mode ensures consistent TTL management by allowing the copying of TTL values between inner and outer headers during encapsulation and decapsulation. This consistency helps in accurately interpreting the packet's remaining lifespan at the receiving device.
- **Network diagnostics and troubleshooting**: By controlling and monitoring the TTL values, you can better diagnose and troubleshoot network paths and performance issues.

Configuration guidelines for TTL uniform mode

This topic lists the configuration guidelines that you must follow when you enable TTL uniform mode for IP-in-IP tunnels on the Cisco 8000 Series Router.

These configuration guidelines apply to the TTL uniform mode:

- **Hardware and feature prerequisites**: TTL uniform mode is enabled only on the Cisco Silicon One Q200 ASIC-based systems when the **pbr vrf-redirect** mode in the **hw-module profile** command is enabled.
- **Post-configuration requirements**: You must reload the router by using the **reload location all** command for the configuration changes to take effect.

Use cases for TTL uniform mode on a router

This topic provides the various encapsulation and decapsulation scenarios and their corresponding TTL actions when TTL uniform mode is enabled on the Cisco 8000 Series Router.

This table details the various scenarios of encapsulation and decapsulation, and their corresponding TTL action that the router performs.

Table 41: Use cases for TTL uniform mode on a router

Use case if the TTL uniform mode is..	Then..	Example
encapsulation-only	The router decrements the TTL value and copies the value from the inner header to outer header.	Consider this example for the encapsulation-only use case for a packet: - **Initial state**: TTL value of the packet is 100. - **Decrement**: The router decrements the inner header TTL value by one, making it 99. - **Copy**: The router then copies the inner header TTL value to the outer header TTL value. The outer header TTL value becomes 99. - **Result**: The inner and outer header TTL becomes 99, making it uniform.

Use case if the TTL uniform mode is..	Then..	Example
decapsulation-only	The router decrements the TTL value and copies the value from the outer header to inner header.	Consider this example for the decapsulation-only use case for a packet: • **Initial state**: The outer header TTL value is 77 and the inner header TTL value is 99. • **Decrement**: The router decreases the outer header TTL value by 1, making it 76. • **Copy**: The router then copies the outer header TTL value to the inner header TTL value. • **Removal**: The router removes the outer header TTL value. • **Result**: The final packet TTL value becomes 76.

Use case if the TTL uniform mode is..	Then..	Example

decapsulation and encapsulation	The router copies the TTL value from the outer header and applies it to the new outer header. Decrements the TTL value by 1 and forwards the packet.	Consider this example for the encapsulation and decapsulation use case. Such scenarios apply when a packet travels through IP tunnels:
		• Encapsulation
		  • **Initial state**: TTL value of the packet is 100.
		  • **Decrement**: The router decrements the inner header TTL value by one, making it 99.
		  • **Copy**: The router then copies the inner header TTL value to the outer header TTL value. The outer header TTL value becomes 99.
		  • **Result**: The inner and outer header TTL becomes 99, making it uniform.
		• Decapsulation
		  • **Initial state**: The outer header TTL value is 77 and the inner header TTL value is 99.
		  • **Decrement**: The router decreases the outer header TTL value by 1, making it 76.
		  • **Copy**: The router then copies the outer header TTL value to the inner header TTL value.
		  • **Removal**: The router removes the outer header TTL value.
		  • **Result**: The final packet TTL value becomes 76.

Use case if the TTL uniform mode is..	Then..	Example
decapsulation and lookup	The router copies the TTL value from the outer header to the inner header. Decrements the TTL value by 1 and forwards the packet.	Consider this example for the decapsulation and lookup use case for a packet: • **Initial state**: The outer header TTL value is 77 and the inner header TTL value is 99. • **Decrement**: The router decreases the outer header TTL value by 1, making it 76. • **Copy**: The router copies the outer header TTL value to the inner header TTL value. • **Removal**: The router removes the outer header TTL value. • **Lookup**: The router then forwards the packet to the next hop.
repair	The router keeps the inner header TTL value unchanged and forwards the packet as usual. The repair use case occurs after the encapsulation-only and decapsulation-encapsulation use cases.	Consider this example for the repair use case for a packet: • **Initial state**: The primary path for a packet on a tunnel is unavailable. • **Initiate recycle**: The router begins the recycling of the packet by re-entering the packet into the ingress pipeline from the egress pipeline. • **Header update**: The router updates the packet by replacing the old outer header with a new outer header.

IP-in-IP decapsulation

This topic describes IP-in-IP decapsulation on the Cisco 8000 Series Router, which removes the outer IP header of an encapsulated packet so that the router can direct the packet to the next hop through the network.

IP-in-IP encapsulation involves the insertion of an outer IP header over the existing IP header. The source and destination addresses in the outer IP header point to the endpoints of the IP-in-IP tunnel. The stack of IP headers is used to direct the packet over a predetermined path to the destination, provided that the network administrator knows the loopback addresses of the routers transporting the packet. This tunneling mechanism can be used for determining availability and latency for most network architectures. The entire path from source to destination does not have to be included in the headers; a segment of the network can be chosen for directing the packets.

In IP-in-IP encapsulation and decapsulation, there are two types of packets. The original IP packets that are encapsulated are called inner packets, and the IP header stack added during encapsulation is called the outer packet.

The router supports only decapsulation, not encapsulation. Encapsulation is performed by remote routers.

Starting with Cisco IOS XR Release 7.3.5, the routers can handle multistack for-us packets within IP-in-IP tunnels efficiently. The for-us packets refer to IP packets with the source and destination terminating on the same router. Such

packets are multistack when they undergo multiple layers of IP header encapsulation at ingress, loopback, and egress interfaces. With this improvement, your routers can identify IP packets with multistack tunneling IP header encapsulations and perform multiple IP header terminations on these packets. This capability enables you to configure IP packets to travel from the source endpoint to the destination endpoint and then back to the source, facilitating streamlined fault identification and isolation processes.

Table 42: Feature History Table

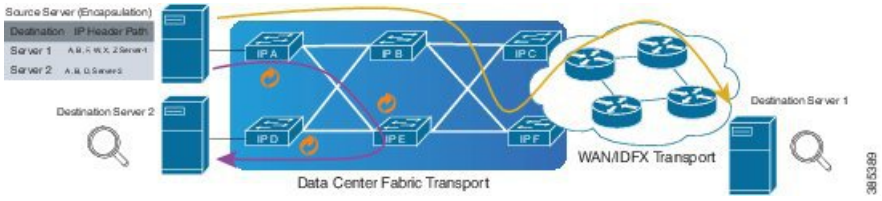
Feature Name	Release Information	Description
IP header decapsulation for multistack packet in IP-in-IP tunnels	Release 7.3.5	Your routers are now enhanced to identify IP packets with multistack IP header encapsulations and perform multiple IP header decapsulations on these packets. With this ability, you can use IP packets to travel from the source endpoint to the destination endpoint and then back to the source in an IP-in-IP tunnel. This feature facilitates identification of IP packets and streamlines fault identification and isolation processes in your IP-in-IP network. This feature is enabled by default and does not require any configuration changes.

Topology for IP-in-IP decapsulation

This topic describes a use case topology where IP-in-IP encapsulation and decapsulation are used for different segments of the network from source to destination on the Cisco 8000 Series Router.

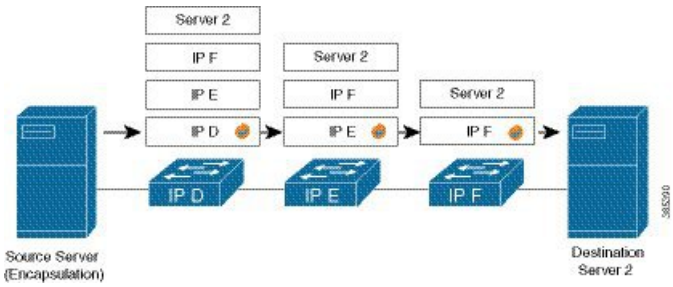
The following topology describes a use case where IP-in-IP encapsulation and decapsulation are used for different segments of the network from source to destination. The IP-in-IP tunnel consists of multiple routers that are used to decapsulate and direct the packet through the data center fabric network.

Figure 14: IP-in-IP decapsulation through a data center network



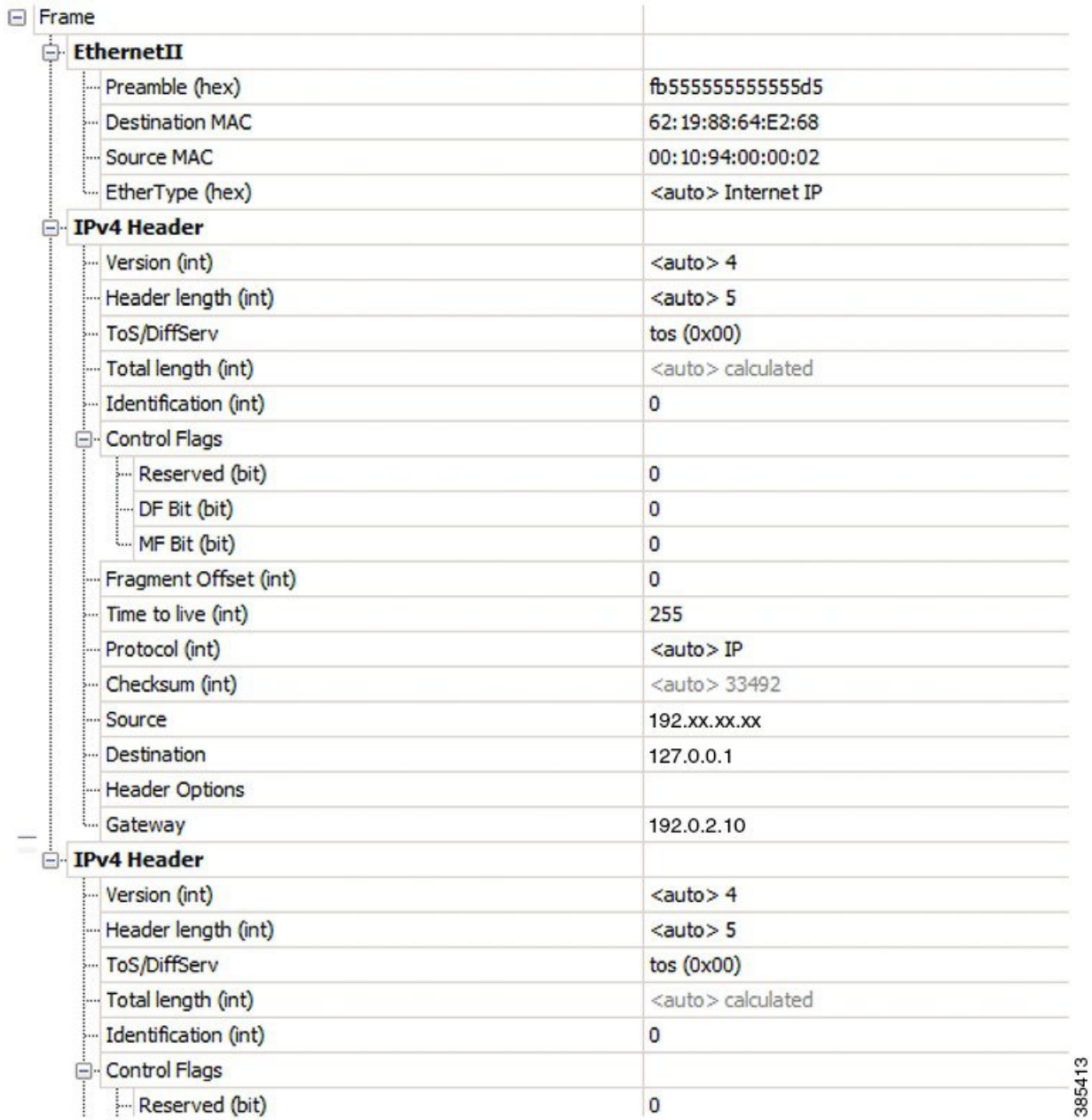
The following illustration shows how the stacked IPv4 headers are decapsulated as they traverse through the decapsulating routers.

Figure 15: IP header decapsulation



The encapsulated packet has an outer IPv4 header that is stacked over the original IPv4 header, as shown in the following illustration.

Figure 16: Encapsulated packet



Decapsulate packets in IP-in-IP tunnel

This topic describes how to configure a router to decapsulate packets as they traverse an IP-in-IP tunnel on the Cisco 8000 Series Router.

Use this procedure to decapsulate the packet as it traverses the IP-in-IP tunnel.

Procedure

1. Enter the interface configuration mode for the loopback interface and assign an IPv4 address.

   Example

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

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

2. Create the tunnel interface using the `tunnel-ip` keyword.

   Example

   ```
   Router(config-if)# interface tunnel-ip 10
   ```

3. Configure the interface to process IPv4 packets without an explicit address by referencing the loopback interface.

   Example

   ```
   Router(config-if)# ipv4 unnumbered loopback 1
   ```

4. Set the tunnel mode for decapsulation. Specify that the tunnel interface performs IP-in-IP decapsulation.

   Example

   ```
   Router(config-if)# tunnel mode ipv4 decap
   ```

5. Define the tunnel source. Specify the source address for the decapsulation tunnel.

   Example

   ```
   Router(config-if)# tunnel source loopback 0
   ```

6. Apply an extended ACL that matches on the outer header for IP-in-IP decapsulation.

Example

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

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

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

Decapsulation using tunnel source direct

This topic describes decapsulation using the **tunnel source direct** command on the Cisco 8000 Series Router, which enables decapsulation of tunnels on any Layer 3 interface for troubleshooting and traffic analysis.

To debug faults in various large networks, you might have to capture and analyze the network traffic at a packet level. In data center networks, administrators face problems with the volume of traffic and the diversity of faults. To troubleshoot faults in a timely manner, data center network administrators must identify affected packets inside large volumes of traffic. They must track them across multiple network components, analyze traffic traces for fault patterns, and test or confirm potential causes.

In some networks, IP-in-IP decapsulation is used in network management to verify ECMP availability and to measure the latency of each path within a data center.

You can perform these actions to manage network traffic using IP-in-IP decapsulation:

- The Network Management System (NMS) sends IP-in-IP (IPv4 or IPv6) packets with a stack of predefined IPv4 or IPv6 headers (device IP addresses).
- The destination device removes the outer header at each hop.
- The device performs a lookup on the next header in the stack.
- The device forwards the packet if a valid route exists.
- Select specific ECMP links for troubleshooting by using the tunnel source direct command.
- Using the **tunnel source direct** command, you can choose specific IP Equal-Cost Multipath (ECMP) links for troubleshooting when there are multiple IP links between two devices.
- You can programmatically configure and manage the Ethernet interfaces using the `openconfig-ethernet-if.yang` and `openconfig-interfaces.yang` OpenConfig data models.

Table 43: Feature History Table

Feature Name	Release Information	Feature Description
Decapsulation using tunnel source direct	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
Decapsulation using tunnel source direct	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Decapsulation using tunnel source direct	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Decapsulation using tunnel source direct	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM
Decapsulation using tunnel source direct	Release 7.5.3	Tunnel source direct allows you to decapsulate the tunnels on any Layer 3 interface on the router. You can use the **tunnel source direct** configuration command to choose the specific IP Equal-Cost Multipath (ECMP) links for troubleshooting, when there are multiple IP links between two devices.

Guidelines and limitations for decapsulation using tunnel source direct

This topic lists the guidelines and limitations that apply to decapsulation using the **tunnel source direct** command on the Cisco 8000 Series Router.

The following guidelines apply to decapsulation using tunnel source direct:

- The **tunnel source direct** command is only compatible with **tunnel mode decap** for IP-in-IP decapsulation.
- The source-direct tunnel is always operationally `up` unless it is administratively shut down. The directly connected interfaces are identified using the **show ip route direct** command.
- All Layer 3 interfaces that are configured on the device are supported.
- The platform can accept and program only a certain number of IP addresses. The number of IP addresses depends on the make of the platform line card (LC). Each LC can have a different number of Network Processor (NP) slices and interfaces.
- Only one source-direct tunnel per address family is supported for configuration.
- Regular decapsulation tunnels that have a specific source address are supported. However, the tunnel's specific source address must not be part of any interface.

The following functionalities are not supported for the **tunnel source direct** option:

- GRE tunneling mode.
- VRF (only default VRF is supported).
- ACL and QoS on the tunnels.
- Tunnel encapsulation.
- Tunnel NetIO DLL: Decapsulation is not supported if the packet is punted to slow path.

Configure decapsulation using tunnel source direct

This topic describes how to configure IP-in-IP tunnel decapsulation on any directly connected IP address by using the **tunnel source direct** command on the Cisco 8000 Series Router.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure terminal
   ```

2. Configure IP-in-IP tunnel decapsulation on directly connected IPv4 addresses with the **tunnel mode ipv4 decap** and **tunnel source direct** commands.

   Example

   ```
   Router(config)# interface Tunnel4
   Router(config-if)# tunnel mode ipv4 decap
   Router(config-if)# tunnel source direct
   Router(config-if)# no shutdown
   ```

3. Alternatively, configure IP-in-IP tunnel decapsulation on IPv6-enabled networks with the **tunnel mode ipv6 decap** and **tunnel source direct** commands.

   Example

```
Router(config)# interface Tunnel6
Router(config-if)# tunnel mode ipv6 decap
Router(config-if)# tunnel source direct
Router(config-if)# no shutdown
```

4. Verify the configuration with the **show running-config interface** and **show interface** commands.

   Example

```
Router# show running-config interface tunnel 1
interface Tunnel1
  tunnel mode ipv6ipv6 decapsulate-any
  tunnel source direct
  no shutdown

Router# show interface tunnel 1
Tunnel1 is up    Admin State: up
MTU 1460 bytes, BW 9 Kbit
Tunnel protocol/transport IPv6/DECAPANY/IPv6
Tunnel source - direct
Tx    0 packets output, 0 bytes    Rx    0 packets input, 0 bytes
```

Tunnel destination with an object group

This topic describes assigning an object group as the destination for an IP-in-IP decapsulation tunnel on the Cisco 8000 Series Router, which allows a single tunnel to accept traffic from multiple IPv4 or IPv6 addresses.

In IP-in-IP decapsulation, the router accepts a packet on a tunneled interface only when the tunnel IP address matches the source IP address of the incoming packets. With this implementation, you need to configure separate interface tunnels for each IP address for which the router receives traffic packets. This limitation often leads to configuration overload on the router.

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

Table 44: Feature History Table

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

Restrictions for tunnel destination with an object group

This topic lists the restrictions that apply when you configure an object group as the tunnel destination for IP-in-IP decapsulation on the Cisco 8000 Series Router.

These restrictions apply to the tunnel destination with an object group feature:

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

Configure tunnel destination with an object group

This topic describes how to configure an object group as the destination for an IP-in-IP decapsulation tunnel on the Cisco 8000 Series Router, so that a single tunnel accepts traffic from multiple IPv4 or IPv6 addresses.

Before you begin

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

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create an IPv4 network object group.

   Example

   ```
   Router(config)# object-group network ipv4 Test_IPv4
   ```

   This command creates a new object group called `Test_IPv4` for IPv4 network addresses and enters the object-group configuration mode.

3. Add network address to the object group.

   Example

   ```
   Router(config-object-group-ipv4)# 192.0.2.0/24
   Router(config-object-group-ipv4)# 198.51.100.0/24
   Router(config-object-group-ipv4)# 203.0.113.0/24
   ```

   This adds the IPv4 subnets 192.0.2.0 with a 24-bit prefix length, 198.51.100.0/24, and 203.0.113.0/24 to the object group.

4. Save the configuration and exit the object-group configuration mode.

   Example

   ```
   Router(config-object-group-ipv4)# commit
   Router(config-object-group-ipv4)# exit
   ```

5. Enter the tunnel interface configuration mode.

   Example

   ```
   Router(config)# interface tunnel-ip 1
   ```

6. Set the tunnel mode to IPv4 decapsulation.

Example

```
Router(config-if)# tunnel mode ipv4 decap
```

7. Configure the tunnel source as direct, so the tunnel accepts all packets with a destination address matching the IP addresses on the router.

   Example

   ```
   Router(config-if)# tunnel source direct
   ```

8. Configure the tunnel destination as the defined IPv4 object group.

   Example

   ```
   Router(config-if)# tunnel destination object-group ipv4 Test_IPv4
   ```

9. Enable the tunnel interface, save the configuration and exit the configuration mode.

   Example

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

   Configure the IPv6 tunnel destination with an object group. Follow the same steps as Define an IPv6 object group with the network elements that must be accepted by the decapsulation tunnel, enter the tunnel configuration mode, set the tunnel mode to IPv6 decap, configure the tunnel source as direct so that the tunnel accepts all packets with a destination address matching the IP addresses on the router, and configure the destination of the tunnel as the defined object group.

   ```
   Router# configure
   Router(config)# object-group network ipv6 Test_IPv6
   Router(config-object-group-ipv6)# 2001:DB8::/32
   Router(config-object-group-ipv6)# 2001:DB8::/48
   Router(config-object-group-ipv6)# commit
   Router(config-object-group-ipv6)# exit
   Router(config)# interface tunnel-ip 2
   Router(config-if)# tunnel mode ipv6 decap
   Router(config-if)# tunnel source direct
   Router(config-if)# tunnel destination object-group ipv6 Test_IPv6
   Router(config-if)# no shutdown
   Router(config-if)# commit
   Router(config-if)# exit
   ```

Hashing for load balancing

This topic describes hashing for load balancing on the Cisco 8000 Series Router, a technique that distributes traffic flows or packets efficiently across multiple available paths, links, or resources.

Hashing is a technique used by network devices such as routers and switches to distribute traffic flows or packets efficiently across multiple available paths, links, or resources. Hashing

- ensures all packets from the same traffic flow (for example, TCP connection) follow the same path, avoiding out-of-order delivery
- distributes multiple traffic flows across all paths, thus preventing overload of any single path, and
- works automatically as paths or links are added or removed.

The hashing process involves these steps:

1. Field extraction: The device extracts certain fields from the packet header, such as source and destination IP addresses, ports, and protocol.
2. Hash function: These fields are combined and fed into a mathematical function called a hash function. This function produces a fixed-size output known as the hash value.
3. Path selection: The resulting hash value is used to select one of several possible paths, links, or next hops. For example, if there are four links in a bundle, the device might use `path = hash_value % 4`, where `%` indicates the modulus operation.

If multiple routers use the same hash algorithm and field selection, the same flows might be mapped to the same paths on each router, resulting in polarization. This can lead to uneven traffic distribution and underused network capacity.

By increasing the variety of hash profiles using extended entropy profiles, each router hashes differently, thus reducing the chance of polarization while improving the overall load balancing. For more information, see Hashing functions for load balancing.

Configure hash rotation value

This topic describes how to configure the hash rotation value (`node-id`) that influences how the hashing algorithms process input fields to combat polarization on the Cisco 8000 Series Router.

These configurations control the `node-id` (also referred to as HASH_ROTATE), which influences how the hashing algorithms process input fields, effectively reordering them to de-correlate flows. This is a mechanism to combat polarization.

Procedure

1. Configure the hash rotation value using one of these options.
   - Set a specific `node-id` (hash rotation value) for the entire chassis. The `node-id` determines one of the 216 possible permutations of input fields for the hashing algorithm.

     ```
     Router# configure
     Router(config)# cef platform load-balancing algorithm adjust 10
     ```

     A specific value might be chosen to optimize load balancing for a known network topology or to address observed polarization issues.
   - Set a specific `node-id` (hash rotation value) for a particular NPU instance on a specific line card. Use location 0/0/CPU0 and instance 0 to identify the NPU. This `node-id` provides granular control, allowing different NPUs within the same chassis to use distinct hash rotation values.

     ```
     Router# configure
     Router(config)# cef platform load-balancing algorithm adjust 20 instance
     0 location 0/0/CPU0
     ```

     This configuration can be useful in complex scenarios where traffic characteristics vary significantly across different NPU-handled interfaces.

- Enable the system to automatically determine and set the `node-id` (hash rotation value) globally for the entire chassis. This configuration aims to dynamically select an optimal `node-id` to improve load balancing and minimize polarization without requiring manual intervention.

```
Router# configure
Router(config)# cef platform load-balancing algorithm adjust auto-global
```

- Enable the system to automatically determine and set the `node-id` (hash rotation value) for each individual NPU instance across all NPUs in the chassis. This allows each NPU to independently optimize its hash rotation based on its specific traffic load and characteristics.

```
Router# configure
Router(config)# cef platform load-balancing algorithm adjust auto-instance
```

2. Execute the **show running-config** command to view the `node-id` (hash rotation value) specified on the location 0/0/CPU0.

Example

```
Router(config)# show running-config | incl cef platform load-balancing
Tue Nov 18 07:14:00.722 UTC
cef platform load-balancing algorithm adjust 20 instance 0 location 0/0/CPU0
Router(config)#
```

Hashing functions for load balancing

This topic describes hashing functions for load balancing on the Cisco 8000 Series Router. A hashing function computes the hash value on specific packet header fields to enable efficient traffic distribution across network paths.

A hashing function is a network algorithm that

- computes a hash value based on selected user-defined packet header fields such as source IP, destination IP, source port, destination port, and next header or protocol
- enables efficient distribution of traffic flows across network paths, and
- minimizes traffic polarization by supporting varied entropy profiles.

Table 45: Feature History Table

Feature Name	Release Information	Description
Enhanced hashing functions using extended entropy profiles	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q200]) This enhancement significantly reduces the risk of traffic polarization, thus ensuring more even and efficient distribution of traffic across multiple network paths in large-scale networks. Load balancing hashing functions have been improved with extended entropy profiles that generate thousands of unique hash function selections on certain user-defined header fields. This feature introduces these changes: CLI: ▪ A new keyword, **extended-entropy**, is added to the **cef platform load-balancing** command. ▪ Two new keywords, **ecmp-seed** and **spa-seed**, are added to the **cef platform load-balancing algorithm adjust** command. ▪ Two new keywords, **hash** and **ip-field-duplication**, are added to the **hw-module cef** command.

Addressing traffic polarization with extended entropy profiles: In large-scale networks, optimal load balancing is critical. Routers use hash algorithms to distribute traffic flows across multiple paths. However, if the hashing functions on consecutive routers are too similar, this can cause polarization, where many flows get mapped to the same path on multiple routers, reducing network efficiency and resilience.

Starting with Cisco IOS XR Release 25.4.1, load balancing hashing functions have been improved with extended entropy profiles. These profiles use specific packet header fields, such as IP addresses and port numbers, to generate thousands of unique hash function combinations. This feature introduces the **extended-entropy** keyword in the **cef platform load-balancing** command to configure the extended entropy.

Configuring ECMP and SPA seed values: You can configure 16-bit seed values for ECMP and SPA (System port aggregate), which is an internal load balancing component. Seed values introduce an additional element of randomness into the hashing process, further helping to de-correlate flows and prevent polarization. To set the seed values for ECMP and SPA, use the **cef platform load-balancing algorithm adjust ecmp-seed** and **cef platform load-balancing algorithm adjust spa-seed** commands, respectively.

Enabling ASIC property for single IP header traffic: We recommend that you enable the **hw-module profile cef hash ip-field-duplication** command when using **cef platform load-balancing extended-entropy**. This setting activates an ASIC property that enhances load balancing for traffic with a single IP header. Without this configuration,

certain devices might not detect all varying bits in IPv4 or IPv6 plain traffic, potentially causing traffic polarization on some nodes.

How enhanced hashing functions for load balancing work

This topic describes the process workflow for enhanced hashing functions for load balancing on the Cisco 8000 Series Router.

Summary

The key components involved in the process are:

- Algorithm adjustments: Modify the load balancing algorithms to use additional entropy from inner header fields during hash calculation.
- Extended entropy profiles: Define two sets of offsets and widths for fields within IPv4 or IPv6 inner headers, increasing the available entropy for hashing.

  The **cef platform load-balancing extended-entropy** configuration leverages the variation or entropy used in the inner header fields and incorporates this information into the outer header's hash calculation. This approach is beneficial when the inner headers have sufficient entropy, but the outer header does not vary.
- ECMP and SPA seed values: Use specific seed values to further diversify the hash results, ensuring unique traffic distribution across routers.

Enhanced hashing functions for load balancing optimize how traffic is distributed across network paths by incorporating greater entropy from packet headers. This process leverages extended entropy profiles, algorithm adjustments, and specific ECMP and SPA seed values to create highly diversified hash results, minimizing the risk of traffic collisions while promoting balanced network utilization.

Workflow

These stages describe the enhanced hashing functions for load balancing:

1. Profile selection: The system selects one of 256 available extended-entropy profiles, each specifying which fields within the inner packet header are used for entropy.
2. Hash input extraction: The selected profile identifies two sets of offsets and widths in the IPv4 or IPv6 inner header, and extracts these field values as hash input.
3. Algorithm application: The load balancing algorithm incorporates the extracted entropy, along with ECMP and SPA seed values, into the hash calculation.
4. Hash calculation: The router computes a hash value based on the combined entropy from the inner header and the seed values.
5. Traffic distribution: The computed hash value determines how each packet flow is distributed across available network paths, ensuring varied and balanced load sharing.

Result

The enhanced hashing process produces highly unique hash inputs, enabling routers to split traffic flows more effectively and consistently achieve balanced utilization of all network paths, even when outer packet headers lack sufficient variability.

Benefits of enhanced hashing functions

This topic lists the benefits of enhanced hashing functions using extended entropy profiles on the Cisco 8000 Series Router.

These are some benefits of the enhanced hashing functions using extended entropy profiles:

- Improves traffic distribution across paths and devices, even with random node-id assignments.
- Suitable for very large, multi-router, multi-path networks.
- Supports a range of configurations and traffic types.

Configuration guidelines for enhanced hashing functions

This topic lists the configuration guidelines that you must follow to avoid conflicts and optimize load balancing when you configure enhanced hashing functions on the Cisco 8000 Series Router.

Follow these configuration guidelines for enhanced hashing functions:

- Do not configure both **cef platform load-balancing extended-entropy** and **cef platform load-balancing fields user-data** at the same time. These configurations are mutually exclusive.
- When using **cef load-balancing algorithm adjust auto-instance** on adjacent devices with small and symmetric topologies, use **cef load-balancing algorithm adjust auto-global** to avoid uneven load balancing.
- Do not configure the same seed value on adjacent devices or within the same topology.
- Enable **cef platform load-balancing extended-entropy** before enabling **hw-module profile cef hash ip-field-duplication**.

Restrictions for enhanced hashing functions

This topic lists the restrictions that apply when you configure enhanced hashing functions on Cisco Silicon One Q200-based systems on the Cisco 8000 Series Router.

The following restriction applies to enhanced hashing functions:

- Only Cisco Silicon One Q200-based systems support the enhanced hashing functions for load balancing.

Configure ECMP and SPA seed values

This topic describes how to set the ECMP and SPA seed values to customize hash algorithm behavior and introduce randomness into the hash function on the Cisco 8000 Series Router.

By default, the hashing algorithm uses a 16-bit seed from the router ID, providing differentiation across devices.

You can customize ECMP or SPA seeds to introduce randomness into the hash function. To avoid load-balancing issues, configure different seed values on adjacent devices or within the same topology.

Procedure

1. Set a specific hexadecimal ECMP seed value (`0xaa` in this example) for the chassis to ensure that the traffic is distributed consistently and without bias across multiple equal-cost paths, preventing polarization.

   Example

   ```
   Router# configure
   Router(config)# cef platform load-balancing algorithm adjust ecmp-seed 0xaa
   Router(config)# commit
   ```

2. Set a specific hexadecimal SPA seed value (`0xbb` in this example) for the chassis so that the seed is incorporated into the hashing calculation to add an element of uniqueness, which can help distribute traffic more evenly across multiple paths.

   Example

   ```
   Router# configure
   Router(config)# cef platform load-balancing algorithm adjust spa-seed 0xbb
   Router(config)# commit
   ```

3. Set a specific hexadecimal ECMP seed value (`0xdd` in this example) on a particular NPU instance to provide NPU-specific control over the ECMP hashing randomness, which can be beneficial in complex network designs.

   Example

   ```
   Router# configure
   Router(config)# cef platform load-balancing algorithm adjust ecmp-seed 0xdd
   instance 0 location 0/0/CPU0
   Router(config)# commit
   ```

4. Set a specific hexadecimal SPA seed value (`0xcc` in this example) on a particular NPU instance. This seed allows for fine-grained control, enabling different NPUs to use distinct SPA seed values if their traffic patterns or load-balancing requirements differ.

   Example

   ```
   Router# configure
   Router(config)# cef platform load-balancing algorithm adjust spa-seed 0xcc
   instance 0 location 0/0/CPU0
   Router(config)# commit
   ```

5. Verify the ECMP and SPA seed value configurations with the **show running-config** command.

   Example

   ```
   Router# show running-config | include cef platform load-balancing
   Tue Nov 18 07:08:56.704 UTC
   cef platform load-balancing algorithm adjust spa-seed 0xcc instance 0 location
    0/0/CPU0
   cef platform load-balancing algorithm adjust ecmp-seed 0xdd instance 0 location
    0/0/CPU0
   Router#
   ```

Configure extended entropy profile

This topic describes how to configure the **cef platform load-balancing extended-entropy** command to increase the number of uncorrelated hash selection algorithms on the Cisco 8000 Series Router.

This procedure allows you to configure **cef platform load-balancing extended-entropy**. This configuration increases the number of uncorrelated hash selection algorithms by modifying how input fields are processed.

Before you configure the extended entropy profile, ensure that you have configured the ECMP and SPA seed values and algorithm adjustments so that the enhanced hashing functions for load balancing work correctly.

Procedure

---

1. Configure the extended entropy profile using one of these commands.

   - Enable the system to automatically select the most suitable extended entropy profile globally for the entire chassis. This is the recommended option.

     ```
     Router# configure
     Router(config)# cef platform load-balancing extended-entropy auto-global
     ```

- Enable the system to automatically select the most suitable extended entropy profile for each individual NPU instance across all NPUs in the chassis. This provides NPU-level optimization, allowing each NPU to dynamically choose a profile that best suits the traffic it is processing, further enhancing load balancing and polarization avoidance.

```
Router# configure
Router(config)# cef platform load-balancing extended-entropy auto-instance
```

- Select a specific extended entropy profile (identified by `profile-index 10` in this example) to be used for hashing across the entire chassis. These profiles define how the hashing functions are applied to various header fields, such as inner source or destination IP, and TCP or UDP ports, to maximize entropy and de-correlation. This allows an administrator to explicitly choose a profile optimized for a particular traffic mix or to address specific polarization issues.

```
Router# configure
Router(config)# cef platform load-balancing extended-entropy profile-index
 10
```

2. Optional: Enable the ASIC property using the **hw-module profile cef hash ip-field-duplication** command. Enabling the ASIC property improves load balancing for traffic with a single IP header. Without this configuration, some devices might not capture the varying bits in IPv4 or IPv6 plain traffic, which results in polarization on certain nodes.

   Example

```
Router# configure
Router(config)# hw-module profile cef hash ip-field-duplication
Router(config)# commit
Router(config)# exit
Router# reload location all
```

3. Execute the **show running-config** command to view the running configuration.

   Example

```
Router# show running-config | incl cef platform load-balancing
Mon Nov 17 08:16:06.749 UTC
cef platform load-balancing extended-entropy auto-global
Router(config)#
```

ECMP hashing support for load balancing

This topic describes Equal-Cost Multi-Path (ECMP) hashing support for load balancing on the Cisco 8000 Series Router, which uses an n-tuple hash algorithm to distribute traffic across multiple equal-cost paths.

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

The load balancing performs these functions:

- Incoming data traffic is distributed over multiple equal-cost connections.
- Incoming data traffic is distributed over multiple equal-cost connections member links within a bundle interface.

- Layer 2 bundle and Layer 3 (network layer) load-balancing decisions are taken on IPv4 and IPv6. If it is an IPv4 or an IPv6 payload, then an n-tuple hashing is done.
- An n-tuple hash algorithm provides more granular load balancing and is used for load balancing over multiple equal-cost Layer 3 (network layer) paths. The Layer 3 (network layer) path is on a physical interface or on a bundle interface.

The n-tuple load-balance hash calculation contains:

- Source IP address
- Destination IP address
- IP protocol type
- Router ID
- Source port
- Destination port
- Input interface
- Flow-label (for IPv6 only)

User-defined fields for ECMP hashing

This topic describes user-defined fields for ECMP hashing on the Cisco 8000 Series Router, which lets you specify additional packet header fields for ECMP path computation so that traffic is distributed evenly across all available equal-cost paths.

ECMP hashing is used to distribute traffic across multiple equal-cost paths. See ECMP hashing support for load balancing for the default static hashing algorithm details.

You can now add user-defined packet header fields for ECMP path calculation for IPv4 and IPv6 flows using the **cef platform load-balancing fields user-data** command. Ensure you specify these user-defined fields based on the type of traffic flow that requires load balancing.

Table 46: Feature History Table

Feature Name	Release Information	Description
User-defined fields for ECMP hashing	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC:K100]))(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.
User-defined fields for ECMP hashing	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: - 8711-48Z-M - 8011-32Y8L2H2FH - 8011-12G12X4Y-A/D

Feature Name	Release Information	Description
User-defined fields for ECMP hashing	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
User-defined fields for ECMP hashing	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM

Feature Name	Release Information	Description
User-defined fields for ECMP hashing	Release 7.5.5 Release 24.2.11	We ensure that in cases where multiple paths are used to carry packets from source to destination, each path is utilized for this purpose and no path is over-utilized or congested. This is made possible because we now provide customized ECMP hashing fields that are used for path computation. Previously, the router relied on fixed packet header fields for hashing, which were not user configurable. With additional user-defined bytes considered for hashing, the granularity at which the traffic can be analyzed for ECMP load balancing increases, resulting in better load balancing and path utilization. This feature introduces these changes: **CLI**: • **cef load-balancing fields user-data** • The **show cef exact-route** command is modified with a new **user-data** keyword. • The **show cef ipv4 exact-route** command is modified with a new **user-data** keyword. • The **show cef ipv6 exact-route** command is modified with a new **user-data** keyword. **YANG**: • New Xpath for Cisco-IOS-XR-8000-fib-platform-cfg.yang

You can include these parameters:

- **Hash header**: The hash header specifies which packet header is being considered for load balancing. You can enable any or all of the available six profiles.
  - IPv4: tcp, udp, non-tcp-udp
  - IPv6: tcp, udp, non-tcp-udp

  If any hash header profile is defined for load balancing, along with the fixed fields considered for hashing, additional bytes in the payload are also used for path computation.
- **Hashing offset**: The hashing offset specifies the byte location from the end of the configured header.

- **Hash size**: The hash size specifies the number of bytes that is considered from the start of the hash offset by the ECMP hashing algorithm. Range is 1 to 4 bytes.
- **Location**: This specifies the location of the ingress line card that receives the incoming traffic. The user-defined hashing configuration is applied on the specified line card.

The addition of the user-defined packet header fields increases the granularity at which the traffic is analyzed for ECMP load balancing. When multiple paths with equal cost are available for routing a specific type of packet from a source to a destination, this granularity ensures that the intended type of traffic is evenly distributed across these paths. This ensures all available paths are used efficiently and prevents congestion or over-utilization of a single path.

You can also retrieve the exact-route information based on the configured user-data using the **show cef exact-route** command with the **user-data** keyword.

Note

- When the user-defined hashing configuration is active, any additional options or optional keywords are disregarded during the parsing of incoming packets for retrieving the user-defined bytes.
- The hashing results based on user-defined hash feature is applicable to BGP or IGP ECMP and LAG hashing.
- The use of the user-defined hashing configuration changes the load balancing behavior of GRE and IP-in-IP traffic. This includes all traffic that begins with IPv4, IPv6, IPv4+UDP, IPv6+UDP, IPv4+TCP, and IPv6+TCP, regardless of the payload.

Configure user-defined fields for ECMP hashing

This topic describes how to configure additional user-defined packet header fields for the ECMP hashing algorithm on the Cisco 8000 Series Router using the **cef load-balancing fields user-data** command.

The **cef load-balancing fields user-data** command configures the additional user-defined fields that are to be considered for the hashing algorithm.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure terminal
   ```

2. Configure the additional IPv4 header fields for TCP packets.

   Example

   ```
   Router# configure terminal
   Router(config)# cef load-balancing fields user-data ipv4 tcp offset 5 size 3
    location 0/0/CPU0
   ```

   In this example, `offset 5` means the payload considered for hashing starts from byte 6 from the end of the TCP header, `size 3` means three bytes of payload are considered, and `location 0/0/CPU0` specifies the line card on which the configuration is applied. Therefore, the sixth, seventh, and eighth bytes of the payload are considered additionally for the hashing.

3. Save the configuration.

   Example

```
Router(config)# commit
```

4. Enter global configuration mode.

   Example

```
Router# configure terminal
```

5. Configure the additional IPv6 header fields for UDP packets.

   Example

```
Router(config)# cef load-balancing fields user-data ipv6 udp offset 0 size 2
 location 0/0/CPU0
Router(config)# commit
```

   In this example, `offset 0` means the payload considered for hashing starts from the end of the UDP header, `size 2` means two bytes of payload are considered, and `location 0/0/CPU0` specifies the line card on which the configuration is applied. Therefore, the first two bytes of payload of a UDP packet are considered additionally for the hashing.

6. Verify the running configuration.

   Example

```
Router# show running-config | include cef
Fri Jul 28 12:02:01.002 UTC
cef load-balancing fields user-data ipv4 tcp offset 5 size 3 location 0/0/CPU0
cef load-balancing fields user-data ipv6 udp offset 0 size 2 location 0/0/CPU0
Router#
```

7. Verify the difference in load balancing before and after applying user-defined hashing, for a flow with data that exhibits good hashing behavior. Before applying user-defined hashing:

   Example

```
Router# show interfaces accounting | i IPV6_U
  Protocol              Pkts In         Chars In     Pkts Out        Chars
Out
  IPV6_UNICAST                1               72            0
 0
  IPV6_UNICAST                1               72            0
 0
  IPV6_UNICAST                1               72            0
 0
  IPV6_UNICAST                1               72            0
 0
  IPV6_UNICAST                2              144            0
 0
  IPV6_UNICAST                1               72            0
 0
  IPV6_UNICAST               0               0      3979416       1981749168
```

```
  IPV6_UNICAST          4191438       2087336124            0
 0
  IPV6_UNICAST                1               72            0
 0
  IPV6_UNICAST                1               72            0
 0
  IPV6_UNICAST                1               72            0
 0
Router#
```

8. Verify the load balancing after applying user-defined hashing.

   Example

```
Router# show interfaces accounting | i IPV6_U
  Protocol               Pkts In         Chars In     Pkts Out        Chars
Out
  IPV6_UNICAST                0               0        39119         19481262

  IPV6_UNICAST                0               0        39801         19820898

  IPV6_UNICAST                0               0        40483         20160534

  IPV6_UNICAST                0               0        40524         20180952

  IPV6_UNICAST                0               0        40573         20205354

  IPV6_UNICAST                0               0        40614         20225772

  IPV6_UNICAST                0               0        39368         19605264

  IPV6_UNICAST                0               0        40734         20285532

  IPV6_UNICAST                0               0        40777         20306946

  IPV6_UNICAST                0               0        40171         20005158

  IPV6_UNICAST                0               0        40858         20347284

  IPV6_UNICAST                0               0        40269         20053962

  IPV6_UNICAST                0               0        41603         20718294

  IPV6_UNICAST                0               0        40363         20100774

  IPV6_UNICAST                0               0        40407         20122686

  IPV6_UNICAST                0               0        41098         20466804

  IPV6_UNICAST           850393        423495714            0
 0
```

9. View the exact route information allocated to the packets using the **show cef exact-route** command with the **user-data** keyword. The packet contains value 0x2 in the packet position for the IPv6 packet, for which the user-defined configuration has been added for a non-tcp-udp IPv6 flow.

Example

```
Router# show cef ipv6 exact-route 100::10 60::1 flow-label 0 protocol 59
source-port 0 destination-port 0 user-data 0x2 ingress-interface
HundredGigE0/0/0/2 location 0/0/cpu0
Unsupported protocol value 59
60::/16, version 1293, internal 0x1000001 0x20 (ptr 0x8b78ef00) [1], 0x400
(0x8e9cfc48), 0x0 (0x0)
 Updated Aug 14 07:50:20.022
 local adjacency to Bundle-Ether3.30

 Prefix Len 16, traffic index 0, precedence n/a, priority 2
   via Bundle-Ether3.30
  via fe80::72b3:17ff:feae:d703/128, Bundle-Ether3.30, 7 dependencies, weight
 0, class 0 [flags 0x0]
    path-idx 7 NHID 0x0 [0x8db8bed8 0x0]
    next hop fe80::72b3:17ff:feae:d703/128
    local adjacency
```

19 Generic UDP encapsulation

Topics:

This chapter details Generic UDP Encapsulation (GUE) configuration, covering tunnel setup, port assignment, and load balancing on Cisco routers.

Generic UDP encapsulation

This topic describes Generic UDP Encapsulation (GUE) on the Cisco 8000 Series Router, which encapsulates IPv4 and IPv6 packets in UDP to provide load-balancing entropy and efficient transport across networks.

Generic UDP Encapsulation (GUE) is a UDP-based network encapsulation protocol that

- encapsulates IPv4 and IPv6 packets in User Datagram Protocol (UDP),
- defines an additional header that helps determine the payload carried by the IP packet, and
- leverages the UDP source port to provide entropy for Equal Cost Multipath (ECMP) hashing and load balancing.

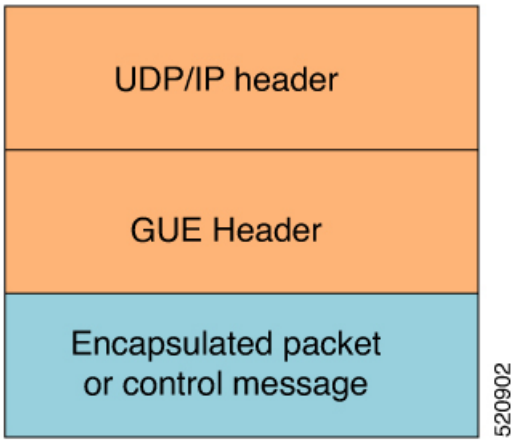
The additional header can include items, such as

- a virtual networking identifier
- security data for validating or authenticating the GUE header, and
- congestion control data

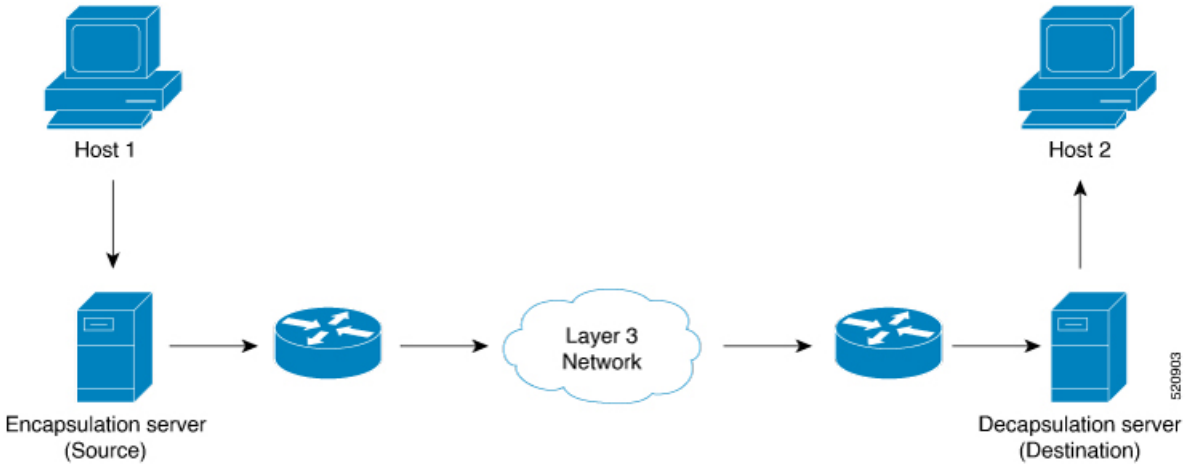
In GUE, the payload is encapsulated in an IP packet that can be IPv4 or IPv6 carrier. The UDP header is added to provide extra hashing parameters and optional payload demultiplexing. At the decapsulation node, the carrier IP and UDP headers are removed, and the packet is forwarded based on the inner payload.

A GUE packet has the general format:

Figure 17: GUE Packet Format



For example, in this scenario, if the data stream is sent from Host 1 to Host 2. The server acts as a GUE encapsulator that sends the packets from Host 1. The server, on the other end receiving the data, validates the data for the valid carrier IP and UDP header and decapsulates the data.



UDP encapsulation is a technique of adding network headers to packets and then encapsulating the packets within UDP. Encapsulating packets using UDP facilitates efficient transport across networks. By leveraging Receive Side Scaling (RSS) and Equal Cost Multipath (ECMP) routing, UDP provides significant performance benefits for load-balancing. The use of the UDP source port provides entropy to ECMP hashing and provides the ability to use the IP source or destination, and the L4 port for load-balancing entropy. Traditional mechanisms like Generic Routing Encapsulation (GRE) can handle only the outer Source IP address and parts of the destination address. They may not provide sufficient load balancing entropy.

GUE has various variants, but variant 1 of GUE allows direct encapsulation of IPv4 and IPv6 in UDP. This technique saves encapsulation overhead on links for the use of IP encapsulation, and does not need to allocate a separate UDP port number for IP-over-UDP encapsulation. Variant 1 has no GUE header, but a UDP packet carries an IP packet. The first two bits of the UDP payload is the GUE variant field and match with the first two bits of the version number in the IP header.

Starting from Cisco IOS XR Release 25.4.1, you can use a single UDP port for both IPv4 and IPv6 packets in GUE encapsulation and decapsulation. The default UDP port for IPv4 and IPv6 is 6080. You can modify the default UDP ports by using the **nve overlay-encap guev1 udp-port destination** command.

Note

The **hw-module profile gue udp-dest-port** command used to configure the UDP ports individually for IPv4, IPv6, and MPLS is deprecated from Cisco IOS XR Release 25.4.1.

Table 47: Feature History Table

Feature Name	Release Information	Feature Description
Single UDP port for IPv4 and IPv6 packets	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200]) This feature enhances memory and network performance by using a single UDP port for both IPv4 and IPv6 packets in GUE variant 1 UDP encapsulation and decapsulation. The default UDP port for IPv4 and IPv6 is 6080. As part of this enhancement, the **hw-module profile gue udp-dest-port** command is deprecated. This feature introduces these changes: CLI: • **nve overlay-encap guev1** • **show cef global udp-ports gue-v1**
Generic UDP Decapsulation for IPv6 Traffic	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Generic UDP Decapsulation for IPv6 Traffic	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
Generic UDP Decapsulation for IPv6 Traffic	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM
Generic UDP Decapsulation for IPv6 Traffic	Release 24.1.1	Starting from this release, you can decapsulate GUEv6 packets by adding an additional header to packets that identifies or authenticates the data by using User Datagram Protocol (UDP). In GUE, the payload is encapsulated in an IP packet that can be an IPv6 carrier. The UDP header is added to provide extra hashing parameters and optional payload demultiplexing. At the decapsulation node, the carrier IP and UDP headers are removed, and the packet is forwarded based on the inner payload.

Benefits of using GUE

- Allows direct encapsulation of payloads, such as IPv4 and IPv6, in the UDP packet. You can use the UDP port for demultiplexing payloads, and you can use a single UDP port that allows systems to employ parsing models to identify payloads.
- Leverages the UDP header for entropy labels by encoding a tuple-based source port.
- Leverages source IP addresses for load-balance encoding. The destination too can be terminated based on a subnet, providing additional bits for entropy.
- Avoids special handling for transit nodes because they only see an IP-UDP packet with some payload.
- Eases implementation of UDP tunneling with GUE because of the direct encapsulation method of the payloads into UDP.

Benefits of single UDP port for IPv4 and IPv6 packets

- Uses a single class map per IP header.
- Allocates one counter per class map.
- Uses simpler logic to reduce duplication of end-to-end objects.

Configuration guidelines for GUE

This topic lists the usage guidelines for Generic UDP Encapsulation (GUE) on the Cisco 8000 Series Router.

These usage guidelines apply for GUE:

- Receives IPv4 packets with the defined GUE port of 6080.
- Receives MPLS packets with the UDP over MPLS (UDPoMPLS) port of 6635.
- GUE for IPv6 traffic is supported only on the Cisco 8202-32FH-M router, and 88-LC0-36FH-M and 88-LC0-36FH line cards.
- GUE IPv6 decapsulation is supported only on Layer 3 ports.

Restrictions for GUE

This topic lists the restrictions that apply to Generic UDP Encapsulation (GUE) decapsulation on the Cisco 8000 Series Router.

These restrictions apply to GUE:

- Supports Generic UDP decapsulation for only variant 1.
- Range of source or destination ports is not supported.
- Range, source, or destination addresses are not supported, but subnet mask entries are allowed.
- Terminating GRE after GUE or GUE after GRE is not supported.
- Terminating a label such as a VPN de-aggregation after GUE termination is not supported.
- Slow path support is not available. To resolve the inner IP adjacency, use the **cef proactive-arp-nd enable** command.
- Running the **clear all** command does not clear the interface of all its existing configurations.
- GUE IPv6 is not supported over BVI interfaces.

> **Note**
>
> To use only the outer IP header (L3 and L4) for calculating the hashing for incoming GUE packets, use the **hw-module profile gue underlay-hash enable** command. Otherwise, by default, both the outer IP header (L3 and L4) and the inner IP header (L3 and L4) are considered for calculating the hashing for incoming GUE packets.
>
> The **hw-module profile gue underlay-hash enable** command is not supported on the P100-based and Q100-based ASICs.

Configure GUE for IPv4

This topic describes how to configure Generic UDP Encapsulation (GUE) variant 1 decapsulation for IPv4 packets on the Cisco 8000 Series Router.

Configure GUE variant 1 for IPv4 by defining the UDP destination port, creating a traffic class to match GUE-encapsulated IPv4 packets, defining a policy map that decapsulates matched traffic, and applying the policy to the VRF.

Procedure

1. Configure the GUE variant 1 UDP destination port for IPv4 using the **nve overlay-encapsulation guev1** command.

Example

```
Router# configure
Router(config-if)# nve overlay-encapsulation guev1
Router(config-nve-encap-guev1)# udp-port destination ipv4 8000
Router(config-nve-encap-guev1)# commit
```

2. Create a traffic class and specify the criteria for classifying GUE-encapsulated IPv4 packets.

Example

```
Router# configure
Router(config)# class-map type traffic match-all gue-v4-udp
Router(config-cmap)# match destination-address ipv4 220.100.20.0
255.255.255.255
Router(config-cmap)# match source-address ipv4 210.100.20.0 255.255.255.255
Router(config-cmap)# match protocol udp
Router(config-cmap)# end-class-map
Router(config)# commit
```

3. Define a policy map and associate the traffic class with the traffic policy.

Example

```
Router(config)# policy-map type pbr magic-decap
Router(config-pmap)# class type traffic gue-v4-udp
Router(config-pmap-c)# decapsulate gue variant 1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic class-default
Router(config-pmap-c)# exit
Router(config-pmap)# end-policy-map
Router(config)# commit
```

4. Apply the policy for each VRF and apply this policy on all the interfaces that are part of the VRF.

Example

```
Router# configure
Router(config)# vrf-policy
Router(config-vrf-policy)# vrf default address-family ipv4 policy type pbr
input magic-decap
Router(config-vrf-policy)# commit
```

5. Run the **show policy-map type pbr addr-family ipv4 statistics** command to view the counter values accumulated for the packets that match the class-map.

Example

```
Router# show policy-map type pbr addr-family ipv4 statistics

VRF Name:       default
Policy-Name:    pmap
Policy Type:    pbr
Addr Family:    IPv4

Class:     cmap-loop3
     Classification statistics      (packets/bytes)
```

```
      Matched             :       198325306/17849277540
    Transmitted statistics          (packets/bytes)
      Total Transmitted   :       198325306/17849277540
```

6. Run the **show nve global** command to view the default configuration if the GUE variant 1 encapsulation is not configured for IPv4 packets.

   Example

```
Router# show nve global
NVE Global details
   VNI Scope Local : No
   VxLAN l3vni bring up mode: V1
   GUEv1 UDP Destination Port (IPv4): 6080
   GUEv1 UDP Destination Port (IPv6): 6615
   GUEv1 UDP Destination Port (mpls): 6635
   Count of NVE interfaces with mpls-udp encap: 0
   Global system mac: 0033.3ebe.8f00
```

7. Run the **show cef global udp-ports gue-v1** command to view the UDP ports for IPv4, IPv6, and MPLS packets in GUE variant 1 encapsulation.

   Example

```
Router# show cef global udp-ports gue-v1
UDP ports for gue-v1 [0x309c3a70f8]
  IPv4: 6080 (default)
  IPv6: 6080 (default)
  MPLS: 6635 (default)
Policy update time:   Not Yet Recorded
Platform update time: Not Yet Recorded
```

8. Run the **clear vrf** command to clear the policy-map counters for each class-map rule.

   Example

```
Router# clear vrf default address-family ipv4 statistics
```

Configure GUE for IPv6

This topic describes how to configure Generic UDP Encapsulation (GUE) variant 1 decapsulation for IPv6 packets on the Cisco 8000 Series Router.

Configure GUE variant 1 for IPv6 by defining the UDP destination port, creating a traffic class to match GUE-encapsulated IPv6 packets, defining a policy map that decapsulates matched traffic, and applying the policy to the VRF.

Procedure

1. Configure the GUE variant 1 UDP destination port for IPv6 using the **nve overlay-encapsulation guev1** command.

Example

```
Router# configure
Router(config-if)# nve overlay-encapsulation guev1
Router(config-nve-encap-guev1)# udp-port destination ipv6 9000
Router(config-nve-encap-guev1)# commit
```

2. Create a traffic class and specify the criteria for classifying GUE-encapsulated IPv6 packets.

Example

```
Router# configure
Router(config)# class-map type traffic match-all gue-v6-udp
Router(config-cmap)# match protocol udp
Router(config-cmap)# match destination-address ipv6 11:1:1::1/128
Router(config-cmap)# end-class-map
Router(config)# commit
```

3. Define a policy map and associate the traffic class with the traffic policy.

Example

```
Router(config)# policy-map type pbr guev6_decap
Router(config-pmap)# class type traffic gue-v6-udp
Router(config-pmap-c)# decapsulate gue variant 1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic class-default
Router(config-pmap-c)# exit
Router(config-pmap)# end-policy-map
Router(config)# commit
```

4. Apply the policy for each VRF.

Example

```
Router# configure
Router(config)# vrf-policy
Router(config-vrf-policy)# vrf default address-family ipv6 policy type pbr
input guev6_decap
Router(config-vrf-policy)# commit
```

5. Run the **show policy-map type pbr vrf default addr-family ipv6 statistics** command to view the counter values accumulated for the packets that match the class-map.

Example

```
Router# show policy-map type pbr vrf default addr-family ipv6 statistics

VRF Name:       default
Policy-Name:    guev6_decap
Policy Type:    pbr
Addr Family:    IPv6

Class:     gue-v6-udp
     Classification statistics      (packets/bytes)
       Matched             :            190/24320
     Transmitted statistics         (packets/bytes)
```

```
       Total Transmitted   :             190/24320

Class:     class-default
     Classification statistics      (packets/bytes)
       Matched             :              0/0
     Transmitted statistics         (packets/bytes)
       Total Transmitted   :              0/0
```

6. Run the **show nve global** command to view the default configuration if the GUE variant 1 encapsulation is not configured for IPv6 packets.

   Example

```
Router# show nve global
NVE Global details
   VNI Scope Local : No
   VxLAN l3vni bring up mode: V1
   GUEv1 UDP Destination Port (IPv4): 6080
   GUEv1 UDP Destination Port (IPv6): 6615
   GUEv1 UDP Destination Port (mpls): 6635
   Count of NVE interfaces with mpls-udp encap: 0
   Global system mac: 0033.3ebe.8f00
```

7. Run the **show cef global udp-ports gue-v1** command to view the UDP ports for IPv4, IPv6, and MPLS packets in GUE variant 1 encapsulation.

   Example

```
Router# show cef global udp-ports gue-v1
UDP ports for gue-v1 [0x309c3a70f8]
  IPv4: 6080 (default)
  IPv6: 6080 (default)
  MPLS: 6635 (default)
Policy update time:   Not Yet Recorded
Platform update time: Not Yet Recorded
```

Outer IP header-driven hash computation for incoming GUE packets

This topic describes outer IP header-driven hash computation for incoming Generic UDP Encapsulation (GUE) packets on the Cisco 8000 Series Router, which uses only the outer IP header to calculate the hash value for load balancing.

When multiple paths with the same cost are available for forwarding traffic, Equal Cost Multipath (ECMP) hashing is used to determine the path to select for each packet. Each packet that needs to be forwarded is processed using a hashing algorithm. The hashing algorithm considers specific packet fields such as source IP, destination IP, source port, and destination port, and generates a hash value. The generated hash value is then mapped to one of the available paths. The selected path is used to forward the packet to its destination. The goal is to distribute the traffic evenly across the available paths to prevent congestion and utilize the network resources efficiently.

You can use only the outer IP header (L3 and L4) for calculating the hash value for incoming GUE packets and completely ignore the inner IP header. This functionality is configurable using the **hw-module profile gue underlay-hash** command. It is supported for both GUE termination (decapsulation) and GUE transit (pass-through) nodes. By default, the feature is disabled, and both the outer IP header (L3 and L4) and the inner IP header (L3 and L4) are used for calculating the hashing for GUE packets.

Benefits

- Load balancing efficiency: By hashing only on the outer IP and L4 information, the packets with the same source and destination IP addresses and L4 ports consistently follow the same path in a load-balanced environment. This helps maintain session affinity or stickiness because the inner IP addresses or L4 port numbers can change dynamically within the encapsulated packets.
- Network security: Ignoring the inner IP helps preserve privacy and confidentiality within the encapsulated packets. By focusing on the outer IP and L4 headers, the network device does not have visibility into the inner IP addressing scheme or the specific content encapsulated within the packet, which enhances security.
- Network scalability: Ignoring the inner IP reduces the complexity and overhead of packet processing, and improves overall network performance and scalability, especially in high-throughput environments.

Configure outer IP header-driven hash computation for incoming GUE packets

This topic describes how to configure hashing that uses only the outer IP header for incoming Generic UDP Encapsulation (GUE) packets on the Cisco 8000 Series Router.

Use the **hw-module profile gue underlay-hash enable** command to configure hashing that uses only the outer IP header for GUE packets.

Procedure

1. Enter global configuration mode and enable outer IP header-driven hash computation for GUE packets.

   Example

   ```
   Router# configure
   Router(config)# hw-module profile gue underlay-hash enable
   Router(config)# commit
   ```

2. Verify the running configuration.

   Example

   ```
   Router(config)# show running-config
   hw-module profile gue underlay-hash enable
   end
   ```

3. Verify the configuration takes effect by comparing the **show dpa objects sys** output before and after enabling the feature.

   Example
   Output before enabling hashing with only the outer IP header for GUE packets:

   ```
   Router# show dpa objects sys location 0/RP0/CPU0 | include gue
     uint32_t gue_ipv4_port => 0
     uint32_t gue_ipv6_port => 0
     uint32_t gue_mpls_port => 0
     ofa_bool_t gue_underlay_hash => FALSE
   ```

Example
Output after enabling hashing with only the outer IP header for GUE packets:

```
Router# show dpa objects sys location 0/RP0/CPU0 | include gue
  uint32_t gue_ipv4_port => 0
  uint32_t gue_ipv6_port => 0
  uint32_t gue_mpls_port => 0
  ofa_bool_t gue_underlay_hash => TRUE
```

Flexible assignment of UDP port numbers for decapsulation

This topic describes flexible assignment of UDP port numbers for Generic UDP Encapsulation (GUE) decapsulation on the Cisco 8000 Series Router, which lets you assign UDP port numbers from 1000 through 64000 to decapsulate IPv4, IPv6, and MPLS packets.

This feature provides decapsulation support for GUE packets. In GUE, the payload is encapsulated in an IP packet–IPv4 or IPv6 carrier. The UDP header is added to provide extra hashing parameters and optional payload demultiplexing. At the decapsulation node, the carrier IP and UDP headers are removed, and the packet is forwarded based on the inner payload. Prior to Release 7.3.3, packets were decapsulated using UDP port numbers 6080, 6615, and 6635 for IPv4, IPv6, and MPLS payloads respectively. Starting from Release 7.3.3, you can assign UDP port numbers from 1000 through 64000 to decapsulate IPv4, IPv6, and MPLS packets. Define different port numbers for IPv4, IPv6, and MPLS.

Table 48: Feature History Table

Feature Name	Release Information	Feature Description
Flexible Assignment of UDP Port Numbers for Decapsulation	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Flexible Assignment of UDP Port Numbers for Decapsulation	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
Flexible Assignment of UDP Port Numbers for Decapsulation	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-36EH • 88-LC1-52Y8H-EM
Flexible Assignment of UDP Port Numbers for Decapsulation	Release 7.3.3	This feature gives you the flexibility to assign UDP port numbers from 1000 through 6400 through which IPv4, IPv6, and MPLS packets can be decapsulated. Such flexibility allows you to segregate the ingress traffic based on a QoS policy. In earlier releases, you could assign only default ports for decapsulation. The following command is introduced for this feature: **hw-module profile gue udp-dest-port ipv4** *port-number* **ipv6** *port-number* **mpls** *port-number*.

Guidelines for setting up decapsulation using flexible port numbers

This topic lists the guidelines to apply when assigning flexible UDP port numbers for Generic UDP Encapsulation (GUE) decapsulation on the Cisco 8000 Series Router.

Apply these guidelines when assigning flexible port numbers for decapsulation:

Packet	IPv4	IPv6	MPLS
UDP outer header	Configure IPv4 port on the hardware module.	Configure IPv6 port on the hardware module.	Configure MPLS port on the hardware module.
Encapsulation outer header	Configure an IPv4 encapsulation outer header that matches with the class map source.		
Inner payload	Packets are forwarded based on the inner IPv4 payload.	Packets are forwarded based on the inner IPv6 payload.	Packets are forwarded based on the inner MPLS payload.

Note

- During the decapsulation of the IPv4, IPv6, and MPLS packets, the following headers are removed:
  - The UDP outer header
  - The IPv4 encapsulation outer header
- Select different values for each of these protocols. Valid port numbers are from 1000 through 64000.

Restrictions for flexible UDP port assignment

This topic lists the restrictions that apply when configuring unique GUE destination port numbers to decapsulate IPv4, IPv6, and MPLS packets using UDP on the Cisco 8000 Series Router.

These restrictions apply when configuring unique GUE destination port numbers to decapsulate IPv4, IPv6, and MPLS packets using UDP:

- While configuring the tunnel, select one of the following:
  - Match only 16 unique source IP addresses, as shown in this example:

```
Router(config-cmap)# match source-address ipv4 210.100.20.0 255.255.255.255
```

  - Match a combination of 64 unique source and destination IP addresses, as shown in this example:

```
Router(config-cmap)# match destination-address ipv4 220.100.20.0
255.255.255.255
Router(config-cmap)# match source-address ipv4 210.100.20.0 255.255.255.255
```

- The Classless Inter-Domain Routing (CIDR) value in the source IP address subnet mask must be only /32.
- The destination address subnet mask supports all CIDR values. However, the destination address, along with the subnet mask, must be unique for all three UDP payload types–IPv4, IPv6, and MPLS. The configuration fails when the destination IP address and the subnet mask are the same for all three payloads, as seen in this example:

```
Router(config)# class-map type traffic match-all SRTE-GUE-DECAP-IPv4
Router(config-cmap)# match destination-address ipv4 10.216.101.0
255.255.255.255
..
Router(config)# class-map type traffic match-all SRTE-GUE-DECAP-IPv6
Router(config-cmap)# match destination-address ipv4 10.216.101.0
255.255.255.255
..
Router(config)# class-map type traffic match-all SRTE-GUE-DECAP-MPLS
Router(config-cmap)# match destination-address ipv4 10.216.101.0
255.255.255.255
..
```

Configure port numbers for decapsulation

This topic describes how to configure flexible UDP port numbers for Generic UDP Encapsulation (GUE) decapsulation on the Cisco 8000 Series Router to match and direct decapsulated traffic to different paths based on the port.

By configuring different port numbers on the destination router, you can match and direct traffic to different paths. For example, traffic for a specific video service can be decapsulated and sent through different ports.

> **Note**
>
> For the hardware module flexible port configuration to take effect, you must reload the line card.

Procedure

1. Configure the UDP destination ports for decapsulation of the required IPv4, IPv6, and MPLS payloads.

   Example

```
Router# configure
Router# hw-module profile gue udp-dest-port ipv4 1001 ipv6 1002 mpls 1003
```

2. Configure the traffic classes to match the destination ports.

   Example

```
Router# configure
Router(config)# class-map type traffic match-all udp-v4
Router(config-cmap)# match protocol udp
Router(config-cmap)# match source-address ipv4 210.100.20.0 255.255.255.255
Router(config-cmap)# match destination-address ipv4 220.100.20.0
255.255.255.255
Router(config-cmap)# match destination-port 1001
Router(config-cmap)# end-class-map
Router(config)# commit

Router(config)# class-map type traffic match-all udp-v6
Router(config-cmap)# match protocol udp
Router(config-cmap)# match destination-address ipv4 220.100.20.0
255.255.255.255
Router(config-cmap)# match source-address ipv4 210.100.20.0 255.255.255.255
Router(config-cmap)# match destination-port 1002
Router(config-cmap)# end-class-map
Router(config)# commit

Router(config)# class-map type traffic match-all udp-mpls1
Router(config-cmap)# match protocol udp
Router(config-cmap)# match destination-address ipv4 220.100.20.0
255.255.255.255
Router(config-cmap)# match source-address ipv4 210.100.20.0 255.255.255.255
Router(config-cmap)# match destination-port 1003
Router(config-cmap)# end-class-map
Router(config)# commit
```

3. Define a policy map and associate the traffic class with the traffic policy.

Example

```
Router(config)# policy-map type pbr magic-decap
Router(config-pmap)# class type traffic udp-v4
Router(config-pmap-c)# decapsulate gue variant 1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic udp-v6
Router(config-pmap-c)# decapsulate gue variant 1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic udp-mpls1
Router(config-pmap-c)# decapsulate gue variant 1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic class-default
Router(config-pmap-c)# exit
Router(config-pmap)# end-policy-map
Router(config)# commit
Router(config)# exit
```

4. Apply the policy for each VRF.

Example

```
Router# configure
Router(config)# vrf-policy
Router(config-vrf-policy)# vrf default address-family ipv4 policy type pbr
input magic-decap
Router(config-vrf-policy)# commit
```

5. Run the **show ofa objects sys location 0/0/CPU0 | inc gue** command in XR Config mode to verify that the unique GUE port numbers have been configured to decapsulate IPv4, IPv6, and MPLS payloads.

Example

```
Router# show ofa objects sys location 0/0/CPU0 | inc gue
uint32_t gue_ipv4_port => 1001
uint32_t gue_ipv6_port => 1002
uint32_t gue_mpls_port => 1003
```

GUEv1 static tunnel configuration over IPv4 networks

This topic describes Generic UDP Encapsulation variant 1 (GUEv1) static tunnel configuration over IPv4 networks on the Cisco 8000 Series Router, which enables scalable and secure deployment of GUEv1 tunnels to carry IPv4 and IPv6 traffic.

A GUEv1 static tunnel is a user-specified tunnel that

- is configured between two endpoints using static route configuration that includes a next-hop IP address,
- enables efficient, scalable, and secure deployment of GUEv1 tunnels, and
- carries IPv4 and IPv6 traffic over IPv4 networks.

Starting from Cisco IOS XR Release 25.4.1, you can configure GUEv1 static tunnels to carry IPv4 and IPv6 traffic over IPv4 networks.

Table 49: Feature History Table

Feature Name	Release Information	Feature Description
GUEv1 static tunnel configuration over IPv4 networks	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q200]) This feature enables highly efficient, scalable, and secure deployment of Generic UDP Encapsulation variant 1 (GUEv1) static tunnels to carry IPv4 and IPv6 traffic over IPv4 networks. In addition, this feature enables the router to use GUEv1 as the overlay encapsulation type for network virtualization endpoint (NVE) and send encapsulated overlay traffic to IPv4, IPv6, and MPLS destinations using UDP ports. With this feature, you can also increase the GUEv1 scale up to 12K tunnels. This feature introduces these changes: CLI: • The **address-family ipv4 unicast** and **address-family ipv6 unicast** commands are modified to include these keywords: **remote-next-hop**, **tunnel**, **source-ip**, **dscp**, and **ttl**. • A new command, **nve overlay-encap guev1 udp-port destination**, has been introduced.

Configuration guidelines for GUEv1 static tunnels

This topic lists the guidelines to configure Generic UDP Encapsulation variant 1 (GUEv1) static tunnels on the Cisco 8000 Series Router.

Apply these guidelines to configure GUEv1 static tunnels:

- Configure the **hw-module profile cef iptunnel scale** command to increase the GUEv1 tunnel scale up to 12K tunnels. This tunnel scale can vary based on the number of ECMP tunnels used.
- By default, up to 3500 GUEv1, MPLS over UDP, and IP-in-IP tunnels, including the number of Layer 3 interfaces, are supported.
- Configure the **nve overlay-encap guev1 udp-port destination** command for IPv4, IPv6, and MPLS to send encapsulated overlay traffic to IPv4, IPv6, and MPLS destinations using UDP ports 6080, 6615, and 6635 respectively.

Restrictions for GUEv1 static tunnels

This topic lists the restrictions that apply to Generic UDP Encapsulation variant 1 (GUEv1) static tunnel configuration on the Cisco 8000 Series Router.

These restrictions are specific to the GUEv1 static tunnel configuration:

- Traceroute over GUEv1 static tunnels is not supported in both uniform and pipe modes.

- Non-default VRF is not supported for GUEv1 static tunnels.

Configure GUEv1 IPv4 and IPv6 static tunnels

This topic describes how to configure IPv4 and IPv6 Generic UDP Encapsulation variant 1 (GUEv1) static tunnels on the Cisco 8000 Series Router.

Use the *prefix mask* **remote-next-hop** command to configure the IPv4 or IPv6 GUEv1 static tunnels.

Procedure

1. Enter global configuration mode, and configure the IPv4 GUEv1 static tunnel.

   Example

```
Router# configure
Router(config)# router static
Router(config-static)# address-family ipv4 unicast
Router(config-static-afi)# 9.8.7.6/32 remote-next-hop 5.5.5.6 tunnel gue-v1
source-ip 1.1.1.1 dscp 46 ttl 250
Router(config-static-afi)# commit
```

2. Configure the IPv6 GUEv1 static tunnel.

   Example

```
Router# configure
Router(config)# router static
Router(config-static)# address-family ipv6 unicast
Router(config-static-afi)# 11:1:1::1/128 remote-next-hop 10.10.10.11 tunnel
gue-v1 source-ip 2.2.2.2 dscp 8 ttl 250
Router(config-static-afi)# commit
```

3. Run the **show running-config router static** command to view the running configuration for the GUEv1 static tunnel.

```
Router# show running-config router static
Thu Mar 20 20:12:16.055 IST
router static
  address-family ipv4 unicast
    9.8.7.6/32 remote-next-hop 5.5.5.6 tunnel gue-v1 source-ip 1.1.1.1 dscp
46 ttl 250
  address-family ipv6 unicast
    11:1:1::1/128 remote-next-hop 10.10.10.11 tunnel gue-v1 source-ip 2.2.2.2
 dscp 8 ttl 25
!
```

4. Run the **show static afi-all topology** command to verify the IPv4 and IPv6 static routes.

   Example

```
Router# show static afi-all topology
Thu Mar 20 20:13:29.394 IST
IPv4 Unicast:
-------------
VRF: default Table Id: 0xe0000000 AFI: IPv4 SAFI: Unicast
Prefix/Len Interface Nexthop Object
```

```
Explicit-path Metrics Local-Label
127.127.127.8/32 None 11.11.11.2 None
None [0/0/1/0/0] No label

IPv6 Unicast:
-------------
VRF: default Table Id: 0xe0800000 AFI: IPv6 SAFI: Unicast
Prefix/Len Interface Nexthop Object
Explicit-path Metrics Local-Label
127:127:127::8/128 None 11.11.11.2 None
None [0/0/1/0/0] No label
```

You have configured IPv4 and IPv6 GUEv1 static tunnels. Configure the global UDP ports for GUEv1 static tunnels next.

Configure global UDP ports for GUEv1 static tunnels

This topic describes how to configure global UDP ports for Generic UDP Encapsulation variant 1 (GUEv1) static tunnels on the Cisco 8000 Series Router.

Before you begin

Ensure that you have configured GUEv1 static tunnels.

This configuration enables the router to use GUEv1 as the overlay encapsulation type for network virtualization endpoint (NVE) and to send encapsulated overlay traffic to IPv4, IPv6, and MPLS destinations using UDP ports 6080, 6615, and 6635 respectively.

Procedure

1. Enter global configuration mode and set the destination UDP port for the encapsulated packets to port 6080 on IPv4.

   Example

   ```
   Router# configure
   Router(config)# nve overlay-encap guev1 udp-port destination ipv4 6080
   Router(config)# commit
   ```

2. Set the destination UDP port for the encapsulated packets to port 6615 on IPv6.

   Example

   ```
   Router(config)# nve overlay-encap guev1 udp-port destination ipv6 6615
   Router(config)# commit
   ```

3. Set the destination UDP port for the encapsulated packets to port 6635 for MPLS.

   Example

   ```
   Router(config)# nve overlay-encap guev1 udp-port destination mpls 6635
   Router(config)# commit
   ```

4. Run the **show running-config nve overlay-encap guev1** command to verify the configured UDP ports.

Example

```
Router# show running-config nve overlay-encap guev1
Wed Nov 19 19:47:11.595 UTC
nve
 overlay-encap guev1
  udp-port destination ipv4 6080
  udp-port destination ipv6 6615
  udp-port destination mpls 6635
 !
!
```

Unidirectional GUEv6 for IPv6 traffic

This topic describes static Generic UDP Encapsulation (GUE) tunnels over an IPv6 underlay (GUEv6) on the Cisco 8000 Series Router, which enables scalable IPv4 and IPv6 overlay transport without dynamic tunnel discovery.

A static GUEv6 is a tunneling protocol feature that

- provides stateless UDP encapsulation using IPv6 as the underlay transport,
- enables overlay transport of IPv4 and IPv6 payloads using static configuration, and
- facilitates migration from legacy MPLS forwarding to IP-only designs.

Table 50: Feature History Table

Feature Name	Release Information	Feature Description
Generic UDP Encapsulation for IPv6 Traffic	Release 26.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q200]) This release introduces support for Generic UDP Encapsulation (GUE) tunnels operating over IPv6 underlays (GUEv6). With this feature, you can enable the highly efficient, scalable, and secure deployment of Generic UDP Encapsulation variant 1 (GUEv1) static tunnels to carry IPv4 and IPv6 traffic over IPv6 networks.

Benefits of GUEv6 tunnels

These are the primary benefits of static GUEv6 tunnels:

- Reduced overlay complexity by using consistent encapsulation across the IPv6 underlay.
- Elimination of dynamic tunnel negotiation and fallback mechanisms.
- Improved manageability with stateless configuration and predictable behavior.

Key attributes include:

- Supports IPv4 and IPv6 overlay payloads across an IPv6 core.
- Uses static configuration for tunnel management.

- Employs a single UDP port, 6080 by default, for encapsulation.

Common use cases for static GUEv6 tunnels include:

- Interconnecting edge routers over an IPv6-only network.
- Providing overlay connectivity where IPv4 address resources are limited.

Configuration guidelines and restrictions for GUEv6

This topic outlines best practices and restrictions for configuring and operating static Generic UDP Encapsulation (GUE) tunnels over an IPv6 underlay (GUEv6) on the Cisco 8000 Series Router.

Best practice for static GUEv6 tunnel configuration

Use static configuration for all required GUEv6 tunnels. Assign explicit local source and remote tunnel endpoint IPv6 addresses to each tunnel, and configure a single default UDP port, port 6080, for operational simplicity. Automate route distribution for tunnel endpoints using BGP, and avoid mixing overlay and underlay VRFs during deployment. Manage VRF selection with DSCP-based policies during migration phases.

- Automate route distribution for tunnel endpoints using BGP to minimize manual errors and increase network adaptability.
- Assign explicit local source and remote tunnel endpoint IPv6 addresses in the static GUEv6 configuration to ensure predictable overlay path selection.

Best practice for migration and operations

Plan IPv6 underlay migration in clear steps with phased VRF selection and deliberate route leaking. Use DSCP-based access-list and Policy-Based Routing (PBR) rules to incrementally divert traffic classes to the appropriate VRF and tunnel. Verify data plane forwarding and monitor tunnel counters after each migration phase.

- Update access lists and policies incrementally, moving traffic classes stepwise to minimize disruption.
- Verify forwarding behavior and tunnel statistics after each change to confirm data flows are consistent and no fallback to default routes occurs.

Interoperability restrictions for static GUEv6 tunnels

These restrictions apply for static GUEv6 tunnels:

- Only static tunnel configuration is available. Dynamic GUE tunnel establishment and auto-discovery are not supported.
- No fallback to default routes or null0 resolution. When a tunnel endpoint is unreachable, the routes using the tunnel are withdrawn.
- Encapsulation and decapsulation typically use the default VRF and depend on underlying ASIC support.
- Per-family or traffic class counters are not available in single UDP port mode. Only per-tunnel endpoint counters are provided.
- MPLS payloads are not supported for static GUEv6 tunnels.
- DSCP and TTL are handled in uniform and pipe mode respectively. Per-tunnel overrides are not supported.

How static GUE over IPv6 underlay works

This topic describes how static Generic UDP Encapsulation (GUE) operates over an IPv6 underlay on the Cisco 8000 Series Router, which enables scalable and interoperable IPv4 and IPv6 payload forwarding.

Static GUE over IPv6 underlay allows routers to encapsulate and transport overlay traffic, including both IPv4 and IPv6, in UDP packets across a core network that supports only IPv6. This approach supports migration from MPLS-based to IP-only transport domains, and makes efficient use of both IPv4 and IPv6 payloads.

Summary

The key components involved in static GUE over IPv6 underlay are:

- Edge routers: Tunnel endpoints that perform encapsulation and decapsulation of overlay traffic.
- Core IPv6 network: Transports encapsulated packets between edge routers.
- BGP: Optionally used to learn tunnel endpoint addresses and facilitate management.

This process creates scalable, stateless, and predictable overlay transport using manually defined tunnel endpoints and encapsulation policies. It simplifies migration from MPLS systems and provides granular control over traffic flows.

Workflow

The process involves these stages:

1. The operator provisions static GUEv6 tunnels on each edge router by specifying the local source IPv6 address and remote IPv6 endpoint. Overlay prefixes are mapped to GUEv6 tunnels with static route configuration.
   - Identify overlay prefixes (IPv4 and IPv6) that require transport.
   - Define static routes for each prefix, specifying the remote endpoint and local source address.
   - BGP can distribute or learn remote endpoints, but tunnel setup stays static.

   Result: Static tunnel mappings are established between all edge nodes.
2. Overlay traffic is selected for encapsulation based on VRF assignment and policies such as DSCP or access-list matching. The router forwards selected traffic into GUEv6 tunnels using static routes with encapsulation.
   - Ingress interfaces and VRF policies direct overlay traffic into the static tunnel, supporting phased migration and traffic engineering.
   - DSCP or Policy-Based Routing (PBR) rules classify and route traffic accordingly.

   Result: Traffic can be steered in a controlled manner, allowing fine-grained policy enforcement.
3. The edge router encapsulates IPv4 or IPv6 payloads using Generic UDP Encapsulation (GUE) version 1, forming a UDP packet with the configured UDP header. Both IPv4 and IPv6 overlays use the same tunnel interface.
   - Packet source and destination addresses are set to the local and remote IPv6 addresses.
   - DSCP and TTL handling are applied per platform capability (usually uniform or pipe modes).
   - Encapsulated packets traverse the IPv6 core using standard routing policies.

   Result: Unified UDP port and stateless encapsulation ensure efficiency and consistent platform behavior.
4. Upon receiving, the edge router decapsulates GUE packets to recover the original payload and forwards it within the destination VRF according to routing policies.
   - The ASIC performs decapsulation to ensure hardware-level scalability, while class maps determine the forwarding decisions.
   - Traffic is restored to the original IPv4 or IPv6 form, preserving headers.

   Result: The overlay packet is delivered to its destination in its original format.

Configure static GUE over IPv6 underlay

This topic describes how to configure a static Generic UDP Encapsulation (GUE) tunnel over an IPv6 underlay on the Cisco 8000 Series Router to transport IPv4 and IPv6 payloads between edge nodes.

Before you begin

Ensure that your environment meets these prerequisites before you configure static GUEv6 tunnels:

- The platform and software version support static GUEv6 tunnel configuration.
- BGP routing is operational and can learn tunnel endpoint addresses.
- Required VRF and interface constructs are present on the routers.
- The local source IPv6 address and remote tunnel endpoint IPv6 address are identified.

Procedure

1. Enter global configuration mode and then the static routing configuration mode.

   Example

   ```
   Router# configure
   Router(config)# router static
   ```

2. Enter IPv4 unicast address family configuration mode to configure an IPv4 payload route over a GUEv6 tunnel.

   Example

   ```
   Router(config-static)# address-family ipv4 unicast
   ```

3. Configure the IPv4 route prefix, IPv6 remote next hop, and GUEv1 tunnel source IPv6 address by using the *ipv4-prefix* **remote-next-hop** *ipv6-destination* **tunnel gue-v1 source-ip** *ipv6-source* command.

   Example

   ```
   Router(config-static-afi)# 200.1.1.1/32 remote-next-hop 2000:75:100:10::16
   tunnel gue-v1 source-ip 64:64::64:64
   ```

4. Exit to static routing configuration mode.

   Example

   ```
   Router(config-static-afi)# exit
   ```

5. Enter IPv6 unicast address family configuration mode to configure an IPv6 payload route over a GUEv6 tunnel.

   Example

   ```
   Router(config-static)# address-family ipv6 unicast
   ```

6. Configure the IPv6 route prefix, IPv6 remote next hop, and GUEv1 tunnel source IPv6 address by using the *ipv6-prefix* **remote-next-hop** *ipv6-destination* **tunnel gue-v1 source-ip** *ipv6-source* command.

   Example

   ```
   Router(config-static-afi)# 2000:1::1/128 remote-next-hop 2000:75:100:10::16
   tunnel gue-v1 source-ip 64:64::64:64
   ```

7. Exit to global configuration mode.

   Example

```
Router(config-static-afi)# exit
Router(config-static)# exit
```

8. Commit the configuration.

   Example

```
Router(config)# commit
```

9. Use these commands to verify the static GUEv6 tunnel configuration:

   Example

```
Router# show tunnel gue summary
Router# show tunnel gue endpoint detail
Router# ping ipv6 2000:75:100:10::16
```

   - Use the **show tunnel gue summary** command to display tunnel summary information.
   - Use the **show tunnel gue endpoint detail** command to display per-endpoint tunnel details.
   - Use the **ping ipv6** *destination* command to verify IPv6 underlay reachability to the tunnel destination.