IP Addresses and Services Configuration Guide for Cisco 8000 Series Routers, IOS XR Releases

Cisco 8000 Series Routers
Updated August 12, 2026

Topics included

1 Getting Started

Outlines the release history and update summary for the IP Addresses and Services Configuration Guide. It enables users to access the latest IOS XR features and release information through a centralized, continuously updated resource.

This cumulative guide provides a single, continuously updated version that includes all the latest IOS XR features and release updates. It simplifies your experience by letting you bookmark one link and access the complete guide, instead of navigating through multiple release-specific versions.

Specific changes or updates tied to individual releases are clearly called out within the relevant sections. For a list of features introduced in a specific release, refer to the Release Notes.

The table lists the release numbers for which this document has been updated since its initial publication.

Table 1: Changes to this document

Date	Summary
August 2026	First published for Release 26.2.1

2 Network Stack IPv4 and IPv6 Addressing

Topics:

Summarizes IPv4 and IPv6 network stack capabilities, address formats, scaling, interface addressing, protocol processing, virtual addressing, ICMP controls, and flexible source IP configuration.

Use this chapter to understand IPv4 and IPv6 addressing behavior and configure address-related network stack functions on router interfaces.

The chapter organizes IPv4 and IPv6 addressing information into these content areas:

Table 2: IPv4 and IPv6 addressing content areas

Content area	Included information
Network stack capabilities	IPv4 and IPv6 functionality, exceptions, host route scale support, and IPv6 prefix scale expansion guidelines.
IPv4 addressing	Interface and virtual addresses, ICMP rate limiting, unnumbered-interface processing, secondary addresses, and IPv4 packet headers.
IPv6 addressing	Address space and formats, global and link-local unicast addresses, IPv4-compatible addresses, packet headers, multicast groups, virtual addresses, and flexible source IP.

The topics progress from network stack capabilities and scale controls to IPv4 addressing and then IPv6 addressing. Requirement and restriction information appears before the related configuration tasks.

Network stack IPv4 and IPv6

Provides comprehensive instructions for configuring and monitoring IPv4 and IPv6 features on interfaces, covering addressing, dual-stack operation, IPv6 Neighbor Discovery behavior, and scaling and address-management options for IPv4 and IPv6 deployments.

Network stack IPv4 and IPv6 features are software functions that

- configure and monitor Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6)
- support interface addressing, dual-stack operation, and IPv6 neighbor discovery, and
- provide scaling and address-management options for IPv4 and IPv6 deployments.

An interface can use multiple IPv6 global addresses. However, multiple IPv6 link-local addresses on the same interface are not supported.

Network stack IPv4 and IPv6 exceptions

Lists network-stack behaviors that can affect operations, including IPv4 and IPv6 command differences, duplicate address conflict handling, protocol state changes, and point-to-point interface behavior.

Use this reference to confirm exceptions before you operate IPv4 and IPv6 interfaces, neighbor entries, or address conflict policies.

Table 3: Network stack exceptions

Exception	Behavior
IPv6 neighbor commands	The **clear ipv6 neighbors** and **show ipv6 neighbors** commands include the **location** *node-id* keyword. If you specify a location, the command displays only neighbor entries in that location.
Stale IPv6 neighbor entries	The **ipv6 nd scavenge-timeout** command sets the lifetime for neighbor entries in the stale state. When the scavenge timer expires, the entry is cleared.
IPv4 and IPv6 interface commands	The **show ipv4 interface** and **show ipv6 interface** commands include the **location** *node-id* keyword. If you specify a location, the command displays only interface entries in that location.
IP address conflict policy	Cisco IOS XR software allows conflicting IP address entries at configuration time. If a conflict exists between two active interfaces, the software brings down the interface according to the configured conflict policy. By default, the higher interface instance is brought down. For example, if `HundredGigE0/0/0/1` conflicts with `HundredGigE0/0/0/2`, the software brings down the IPv4 protocol on `HundredGigE0/0/0/2`. IPv4 remains active on `HundredGigE0/0/0/1`.

IPv4 and IPv6 functionality

Describes how configured interfaces process IPv4 and IPv6 traffic, including dual-stack forwarding, IPv6 design benefits, simplified packet processing, neighbor discovery, and multicast listener discovery.

IPv4 and IPv6 functionality is dual-stack interface support that

- lets an interface send and receive data on both IPv4 and IPv6 networks

- helps existing IPv4 deployments transition to IPv6 while retaining IPv4 connectivity, and
- uses IPv6 capabilities such as larger address space, simplified headers, prefix aggregation, and neighbor discovery.

IPv6 protocol context

IPv6, formerly named IPng, is a packet-based protocol used to exchange data, voice, and video traffic over digital networks. IPv6 was proposed when the 32-bit IPv4 addressing scheme became insufficient for Internet growth.

IPv6 adds a larger address space, a simplified main header, and extension headers. RFC 2460 describes the initial IPv6 specification from the Internet Engineering Task Force (IETF).

IPv6 design benefits

IPv6 design provides these benefits:

- Larger address space for scale and global reachability.
- Simplified packet header handling for more efficient packet processing.
- Prefix aggregation, simplified network renumbering, and site multihoming for a more efficient addressing hierarchy.
- Support for routing protocols such as Open Shortest Path First (OSPF) and multiprotocol Border Gateway Protocol (BGP).

IPv6 neighbor discovery role

The IPv6 neighbor discovery process uses Internet Control Message Protocol (ICMP) messages and solicited-node multicast addresses to determine the link-layer address of a neighbor on the same local link, verify neighbor reachability, and track neighboring routers.

Host route scale support

Describes connected host-route scaling for IPv4 and IPv6 and includes the configuration workflow used to set maximum host-route scale and verify the hardware resource mode.

Host route scale support is a connected host-route scaling feature that

- increases IPv4 and IPv6 host route scale values
- uses an alternative programming method so that next-hop scale limitations are not applied to connected host routes, and
- lets the router support both IPv4 and IPv6 host routes at a larger scale.

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Host route scale support	Release 25.4.1	This feature provides an alternative programming approach to configure the connected host route, enhancing the scalability for both IPv4 and IPv6. Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8200 [ASIC: P100], 8700 [ASIC: P100], 8700 [ASIC: K100])(select variants only*) *This feature is supported on: • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M The feature introduces the hw-module profile route scale host-route command.

Configure host route scale support

Configures maximum connected host-route scale for IPv4 and IPv6 host routes, reloads the affected location, and verifies the resulting hardware resource mode.

Configure host route scale support when connected IPv4 and IPv6 host routes require higher scale and must avoid next-hop scale limitations.

Procedure

1. Configure maximum scaling for connected host routes.

   Example

   ```
   Router# configure
   Router(config)# hw-module profile route scale host-route
   Router(config)# commit
   ```

2. Reload the router or the affected line cards.

   You must reload the chassis or all affected line cards to activate or deactivate host-route scale mode.

3. Verify the host-route scale status.

Example

```
Router# show hw-module profile route-scale
-------------------------------------------------------------
Knob                  Status        Applied       Action
-------------------------------------------------------------
IPv6-Pfx-Expansion    Unconfigured  N/A           None
LPM-Tcam-Scale        Unconfigured  N/A           None
Wide-Entries-Shortened Unconfigured N/A           None
LPM-CEM-Scale         Unconfigured  N/A           None
LPM-Full-Scale        Unconfigured  N/A           None
Host-Route-Scale      Configured    Yes           None
```

IPv6 prefix scale expansion

Describes how /126 and /127 IPv6 prefixes expand into /128 entries for CEM storage, including supported router scope, scale limits, reload requirements, and configuration steps.

IPv6 prefix scale expansion is a route-scale feature that

- stores long connected IPv6 prefixes in Central Exact Match (CEM) memory
- expands each /126 prefix into four /128 prefixes and each /127 prefix into two /128 prefixes, and
- helps avoid Longest Prefix Match (LPM) memory allocation failures when many /126 and /127 IPv6 prefixes are present.

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
IPv6 prefix scale expansion	Release 25.1.1	ntroduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
IPv6 prefix scale expansion	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-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
IPv6 prefix scale expansion	Release 7.3.5	You can use Central Exact Match (CEM) memory to store /126 and /127 IPv6 prefixes when the prefixes do not fit in Longest Prefix Match (LPM) memory. To store these prefixes in CEM memory, expand each /126 prefix into four /128 prefixes and each /127 prefix into two /128 prefixes. In earlier releases, Cisco IOS XR software stored /126 and /127 IPv6 prefixes only in LPM memory. This feature introduces these commands: • **hw-module profile route scale ipv6-unicast connected-prefix high** • **show hw-module profile route-scale**

Guidelines for IPv6 prefix scale expansion

Provides the supported-router requirement for IPv6 prefix scale expansion, including Q100 and Q200 ASIC-based router scope, CEM entry limits, and the reload requirement.

Supported routers

Use IPv6 prefix scale expansion only on Q100 or Q200 ASIC-based Cisco 8000 routers. To know more about Q100 and Q200 ASIC-based Cisco 8000 routers, see the Cisco 8000 Series Router data sheet.

- Q100 ASIC-based Cisco 8000 routers support 256,000 Central Exact Match (CEM) scale entries.
- Q200 ASIC-based Cisco 8000 routers support 608,000 CEM scale entries.

Reload the router after you configure IPv6 prefix scale expansion for the feature to take effect.

Configure IPv6 prefix scale expansion

Configures IPv6 prefix scale expansion, reloads the affected locations so the setting takes effect, and verifies that expanded prefixes are installed in hardware.

Configure IPv6 prefix scale expansion when many /126 or /127 connected IPv6 prefixes must be stored in CEM memory instead of LPM memory.

Procedure

1. Configure the IPv6 unicast connected-prefix route-scale profile.

   Example

```
Router# configure
Router(config)# hw-module profile route scale ipv6-unicast connected-prefix
high
Router(config)# commit
Router(config)# end
```

2. Reload the router for the feature to take effect.

Example

```
Router# reload location all
Proceed with reload? [confirm] y
```

3. Verify that the route-scale profile is applied.

Example

```
Router# show hw-module profile route-scale
-------------------------------------------------------------
Knob          Status       Applied       Action
-------------------------------------------------------------
Route-Scale   Configured   Yes           None
```

4. Verify the expanded IPv6 prefixes.

Example

```
Router# show cef ipv6 2001:DB8:23:1::/126 hardware egress detail location
0/0/CPU0
leaf npd data:
Specific route may have overwritten NH for expanded prefix.
In that case, use specific route prefix for this command.
Expanded_Prefix0:{v6addr: 2001:0DB8:0023:0001:0000:0000:0000:0000}/128
Expanded_Prefix1:{v6addr: 2001:0DB8:0023:0001:0000:0000:0000:0001}/128
Expanded_Prefix2:{v6addr: 2001:0DB8:0023:0001:0000:0000:0000:0002}/128
Expanded_Prefix3:{v6addr: 2001:0DB8:0023:0001:0000:0000:0000:0003}/128

Router# show cef ipv6 2001:DB8:23:1::/127 hardware egress detail location
0/0/CPU0
leaf npd data:
Specific route may have overwritten NH for expanded prefix.
In that case, use specific route prefix for this command.
Expanded_Prefix0:{v6addr: 2001:0DB8:0023:0001:0000:0000:0000:0000}/128
Expanded_Prefix1:{v6addr: 2001:0DB8:0023:0001:0000:0000:0000:0001}/128
npu: 0, npu_rc: 0
npu: 1, npu_rc: 0
```

How to implement network stack IPv4 and IPv6

Describes IPv4 implementation workflows, including access prerequisites, primary and secondary interface addressing, virtual management addresses, ICMP rate limiting, and unnumbered point-to-point interfaces.

Prerequisites for implementing network stack IPv4 and IPv6

Use this reference to confirm the required authorization before you configure network stack IPv4 and IPv6 features.

Network Stack IPv4 and IPv6 tasks require the correct user group assignment:

- You must be in a user group that is associated with a task group that includes the required task IDs.
- The command reference guides list the task IDs required for each command.
- If user group assignment prevents command use, contact your AAA administrator for assistance.

IPv4 packet headers

Describes IPv4 header fields, basic header size, options, packet data, and the comparison points used to distinguish IPv4 headers from simplified IPv6 headers.

An IPv4 packet header is a network-layer packet header that

- contains 12 basic fields
- has a basic header size of 20 octets, or 160 bits, and
- can include an Options field before the packet data portion.

Basic header size

The basic IPv4 packet header has 12 fields with a total size of 20 octets, or 160 bits.

Options and data portion

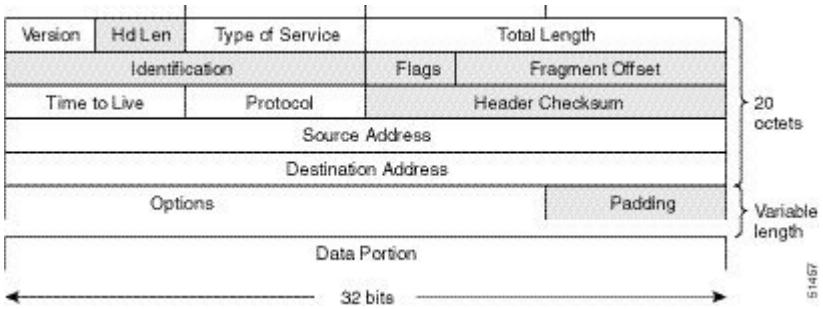
The 12 basic header fields can be followed by an Options field. The Options field is followed by the data portion, which is usually the transport-layer packet.

The variable length of the Options field adds to the total size of the IPv4 packet header.

Header fields

This figure displays IPv4 packet header format.

Figure 1: IPv4 packet header format



The IPv4 packet header format includes these fields and areas:

- Version
- Header Length
- Type of Service
- Total Length
- Identification
- Flags
- Fragment Offset
- Time to Live
- Protocol
- Header Checksum
- Source Address

- Destination Address
- Options
- Padding
- Data Portion

IPv4 addresses

Describes IPv4 interface addressing behavior, including primary address use, secondary address support, leading-zero rejection, contiguous mask requirements, and prefix-length notation.

An IPv4 address is an interface identifier that

- identifies a location where IPv4 datagrams can be sent
- enables communication with hosts on the configured interface, and
- uses a network mask or prefix length to identify the network portion of the address.

An interface can have one primary IPv4 address and multiple secondary IPv4 addresses. Packets generated by the software use the primary IPv4 address. Therefore, all networking devices on a segment should share the same primary network number.

Starting with Cisco IOS XR Release 7.10.2, the CLI does not accept IPv4 addresses with leading zeros. Use canonical dotted-decimal IPv4 addresses, such as `198.51.100.1`. NETCONF and YANG reject IPv4 addresses that contain leading zeros.

Cisco supports only network masks that use contiguous bits that are flush left against the network field.

A network mask can be a four-part dotted decimal address, such as `255.255.255.0`, or a slash and prefix length, such as `/24`. In prefix-length notation, the slash must immediately follow the IPv4 address.

Configure IPv4 addresses

Configures a primary IPv4 address on a network interface and verifies, through interface output, that IPv4 is enabled on the interface.

Configure IPv4 addresses on interfaces to enable IPv4 communication with hosts on those interfaces.

Before you configure IPv4 on an interface, confirm the primary address, prefix length, and network mask requirements.

Procedure

1. Enter interface configuration mode.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/24
   ```

2. Assign the IPv4 address and prefix length to the interface.

   Example

   ```
   Router(config-if)# ipv4 address 198.51.100.1/24
   ```

3. Enable the interface and commit the configuration.

Example

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

4. Verify the running configuration.

Example

```
Router# show running-config interface HundredGigE 0/0/0/24
interface HundredGigE0/0/0/24
 ipv4 address 198.51.100.1 255.255.255.0
!
```

5. Verify that the interface is active and IPv4 is enabled.

Example

```
Router# show ipv4 interface HundredGigE 0/0/0/24
HundredGigE0/0/0/24 is Up, ipv4 protocol is Up
Vrf is default (vrfid 0x60000000)
Internet address is 198.51.100.1/24
MTU is 1514 (1500 is available to IP)
Helper address is not set
Directed broadcast forwarding is disabled
Outgoing access list is not set
Inbound common access list is not set, access list is not set
Proxy ARP is disabled
ICMP redirects are never sent
ICMP unreachables are always sent
ICMP mask replies are never sent
Table Id is 0xe0000000
```

The interface has a primary IPv4 address and IPv4 is operational on the interface.

IPv4 virtual addresses

Describes stable management access across RP failover, including common-subnet requirements for Management Ethernet interfaces, per-VRF virtual addresses, and source-address substitution for management applications.

An IPv4 virtual address is a management address that

- lets you access the router from a single IPv4 address without knowing which route processor is active
- persists across route processor failover, and
- must share a common IPv4 subnet with a Management Ethernet interface on both route processors.

The **vrf** keyword supports virtual addresses on a per-VRF basis.

The **use-as-src-addr** keyword lets management applications use a relevant virtual IPv4 address as the source address when the transport process selects a Management Ethernet address. This behavior works across route processor switchovers. Without the keyword, the selected source address can change after failover and network management software might not handle the change.

Protocol configurations such as **tacacs source-interface**, **snmp-server trap-source**, **ntp source**, and **logging source-interface** do not use the virtual management IPv4 address as the source by default. Use the **ipv4 virtual address use-as-src-addr** command, or configure a loopback address and set it as the source interface for the protocol.

Configure IPv4 virtual addresses

Configures an IPv4 virtual management address and verifies the Management Ethernet interface configuration used for stable access across RP failover.

Configure IPv4 virtual addresses to provide stable management access across route processor failover.

The virtual IPv4 address must share a common IPv4 subnet with a Management Ethernet interface on both route processors.

Procedure

1. Configure the IPv4 virtual address.

   Example

```
Router# configure terminal
Router(config)# ipv4 virtual address 192.0.2.28/24
Router(config)# commit
```

2. Configure the Management Ethernet interface in the same subnet.

   Example

```
Router# configure terminal
Router(config)# ipv4 virtual address 192.0.2.28/24
Router(config)# interface MgmtEth 0/RP0/CPU0/0
Router(config-if)# ipv4 address 192.0.2.28/24
Router(config-if)# no shutdown
Router(config-if)# commit
Router(config-if)# exit
```

3. Verify the Management Ethernet interface configuration.

   Example

```
Router# show running-config interface MgmtEth 0/RP0/CPU0/0
interface MgmtEth0/RP0/CPU0/0
 ipv4 address 192.0.2.28 255.255.255.0
!
```

4. Configure the Management Ethernet interface in a VRF when virtual management access is required for that VRF.

   Example

```
Router# configure terminal
Router(config)# interface MgmtEth 0/RP0/CPU0/0
Router(config-if)# vrf test
Router(config-if)# ipv4 address 192.0.2.29 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# commit
```

5. Verify the VRF interface configuration.

Example

```
Router# show running-config interface MgmtEth 0/RP0/CPU0/0
interface MgmtEth0/RP0/CPU0/0
 vrf test
 ipv4 address 192.0.2.29 255.255.255.0
!
```

The IPv4 virtual address and Management Ethernet interface configuration are available for management access.

IPv4 ICMP rate limiting

Describes how IPv4 ICMP rate limiting controls destination unreachable messages, including general timers, DF timers, message precedence, and global configuration behavior.

IPv4 ICMP rate limiting is a control-plane protection feature that

- limits the rate at which IPv4 ICMP destination unreachable messages are generated
- uses one timer for general destination unreachable messages, and
- uses a separate timer for Don't Fragment (DF) destination unreachable messages when the **DF** keyword is configured.

The general destination unreachable timer and DF destination unreachable timer share the same limits and defaults. When the **DF** keyword is configured, the DF timer remains independent from the general destination unreachable timer.

The optional **DF** keyword limits ICMP destination unreachable messages sent when code 4 fragmentation is needed and Don't Fragment is set in the IP header.

Use the **show ipv4 traffic** command to identify how many ICMP unreachable messages were sent or received.

Configure IPv4 ICMP rate limiting

Configures IPv4 ICMP destination unreachable rate limiting and verifies the resulting ICMP traffic counters after the configured timer is applied.

Configure IPv4 ICMP rate limiting to control the rate at which IPv4 ICMP destination unreachable messages are generated.

Configure the general destination unreachable rate limit and, when required, configure an independent DF destination unreachable rate limit.

Procedure

1. Configure the general IPv4 ICMP destination unreachable rate limit.

   Example

   ```
   Router# configure
   Router(config)# icmp ipv4 rate-limit unreachable 1000
   ```

2. Configure the DF destination unreachable rate limit.

   Example

   ```
   Router(config)# icmp ipv4 rate-limit unreachable DF 1000
   Router(config)# commit
   ```

3. Verify the ICMP rate-limit configuration.

   Example

```
Router# show running-config | in icmp
Building configuration...
icmp ipv4 rate-limit unreachable DF 1000
icmp ipv4 rate-limit unreachable 1000
```

4. Verify that IPv4 is operational on the interface.

   Example

```
Router# show ipv4 interface HundredGigE0/0/0/2
HundredGigE0/0/0/2 is Up, ipv4 protocol is Up
Vrf is default (vrfid 0x60000000)
Internet address is 192.0.2.2/24
MTU is 1514 (1500 is available to IP)
Helper address is not set
Multicast reserved groups joined: 224.0.0.2 224.0.0.1 224.0.0.2
224.0.0.5 224.0.0.6
Directed broadcast forwarding is disabled
Outgoing access list is not set
Inbound common access list is not set, access list is not set
Proxy ARP is disabled
ICMP redirects are never sent
ICMP unreachables are always sent
ICMP mask replies are never sent
Table Id is 0xe0000000
```

5. Verify the ICMP unreachable counters.

   Example

```
Router# show ipv4 traffic
ICMP statistics:
Sent: 0 admin unreachable, 5 network unreachable
0 host unreachable, 0 protocol unreachable
0 port unreachable, 0 fragment unreachable
0 time to live exceeded, 0 reassembly ttl exceeded
0 echo request, 0 echo reply
0 mask request, 0 mask reply
0 parameter error, 0 redirects
5 total
Rcvd: 0 admin unreachable, 0 network unreachable
0 host unreachable, 0 protocol unreachable
0 port unreachable, 0 fragment unreachable
0 time to live exceeded, 0 reassembly ttl exceeded
0 echo request, 0 echo reply
0 mask request, 0 mask reply
0 redirect, 0 parameter error
0 source quench, 0 timestamp, 0 timestamp reply
0 router advertisement, 0 router solicitation
0 total, 0 checksum errors, 0 unknown
```

IPv4 processing on an unnumbered interface

Describes how a point-to-point interface can use another interface address as its IPv4 source address, including source-address selection and interface restrictions.

IPv4 processing on an unnumbered interface is an interface addressing method that

- enables IPv4 on a point-to-point interface without assigning an explicit IPv4 address to that interface
- uses the address of a specified interface as the source address for generated packets, and
- uses the specified interface address to determine which routing processes send updates over the unnumbered interface.

An unnumbered interface has no IPv4 address of its own. You cannot use the **ping** EXEC command to determine whether the unnumbered interface is up. Use SNMP to remotely monitor interface status.

IP security options are not supported on an unnumbered interface.

Enable IPv4 processing on unnumbered interfaces

Enables IPv4 processing on a point-to-point interface without assigning an explicit IPv4 address, using another interface as the source address.

Enable IPv4 processing on an unnumbered interface when a point-to-point interface must use the IPv4 address of another interface as its source address.

Identify the point-to-point interface and the loopback interface whose IPv4 address the unnumbered interface must use.

Procedure

1. Enter interface configuration mode for the point-to-point interface.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/25
   ```

2. Enable IPv4 point-to-point processing and commit the configuration.

   Example

   ```
   Router(config-if)# ipv4 point-to-point
   Router(config-if)# commit
   ```

3. Configure the interface to use the loopback interface IPv4 address and commit the configuration.

   Example

   ```
   Router(config-if)# ipv4 unnumbered loopback 0
   Router(config-if)# commit
   ```

4. Verify the unnumbered interface configuration.

Example

```
Router# show running-config interface HundredGigE 0/0/0/25
interface HundredGigE0/0/0/25
 ipv4 point-to-point
 ipv4 unnumbered Loopback0
!
```

5. Verify that the unnumbered interface is up.

Example

```
Router# show interface HundredGigE 0/0/0/25
HundredGigE0/0/0/25 is up, line protocol is up
Interface state transitions: 5
Hardware is HundredGigE, address is 00e2.2a33.445b (bia 00e2.2a33.445b)
Layer 1 Transport Mode is LAN
Internet address is 192.0.2.1/24
MTU 1514 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
reliability 255/255, txload 194/255, rxload 0/255
Encapsulation ARPA,
Full-duplex, 10000Mb/s, link type is force-up
```

6. Verify the loopback interface configuration.

Example

```
Router# show running-config interface Loopback 0
interface Loopback0
 ipv4 address 192.0.2.1 255.255.255.255
```

Secondary IPv4 addresses on network interfaces

Describes how one network interface can support multiple IPv4 subnets on a physical segment, including configuration steps and consistency guidelines for routers on the segment.

A secondary IPv4 address is an additional interface address that

- supplements the primary IPv4 address on a network interface
- lets one interface support more than one logical IPv4 subnet on the same physical segment, and
- is configured with the **secondary** keyword.

Cisco IOS XR software supports multiple secondary IPv4 addresses per interface.

For IPv6, an interface can have multiple IPv6 addresses without the **secondary** keyword.

Guidelines for secondary IPv4 addresses on network interfaces

Lists use cases and requirements for secondary IPv4 addresses, including host expansion, migration from bridged networks, separated subnet designs, subnet placement, segment consistency, and routing-loop avoidance.

Use these guidelines before you configure secondary IPv4 addresses on a network segment.

Table 6: Secondary IPv4 address guidelines

Guideline	Details
Use secondary addresses when a segment needs more host addresses.	If subnetting allows up to 254 hosts per logical subnet but a physical subnet needs 300 host addresses, secondary IPv4 addresses let routers or access servers support two logical subnets on one physical subnet.
Use secondary addresses during migration from bridged networks.	Older Layer 2 bridged networks that were not subnetted can use secondary IPv4 addresses during migration to a subnetted, router-based network. Routers on the segment can become aware that many subnets are on that segment.
Use secondary addresses to join separated subnets of the same network.	You can create a single network from subnets that are physically separated by another network by using a secondary address. In this design, the first network is extended or layered on top of the second network.
Do not place the same subnet on more than one active interface.	A subnet cannot appear on more than one active interface of the router at a time.
Keep secondary-address use consistent across the segment.	If any router on a network segment uses a secondary IPv4 address, all other routers on that same segment must also use a secondary address from the same network or subnet.
Avoid inconsistent secondary-address use.	Inconsistent use of secondary addresses on a network segment can quickly cause routing loops.

Configure secondary IPv4 addresses on network interfaces

Configures a secondary IPv4 address on a network interface and verifies that the interface shows both primary and secondary addresses.

Configure secondary IPv4 addresses when an interface must support more than one IPv4 subnet on the same physical network segment.

Identify the interface, primary IPv4 address, and secondary IPv4 address before you configure the secondary address.

Procedure

1. Enter interface configuration mode.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/24
   ```

2. Assign the secondary IPv4 address to the interface and commit the configuration.

   Example

   ```
   Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0 secondary
   Router(config-if)# commit
   ```

3. Verify the running configuration.

Example

```
Router# show running-config interface HundredGigE 0/0/0/24
interface HundredGigE0/0/0/24
 ipv4 address 192.0.2.27 255.255.255.0
 ipv4 address 192.0.2.1 255.255.255.0 secondary
!
```

4. Verify that the interface lists the secondary IPv4 address.

Example

```
Router# show ipv4 interface HundredGigE 0/0/0/24
HundredGigE0/0/0/24 is Up, ipv4 protocol is Up
Vrf is default (vrfid 0x60000000)
Internet address is 192.0.2.27/24
Secondary address 192.0.2.1/24
MTU is 1514 (1500 is available to IP)
Helper address is not set
Directed broadcast forwarding is disabled
Outgoing access list is not set
Inbound common access list is not set, access list is not set
Proxy ARP is disabled
ICMP redirects are never sent
ICMP unreachables are always sent
ICMP mask replies are never sent
Table Id is 0xe0000000
```

IPv6 addresses

Describes IPv6 interface addressing, including address-space scale, syntax, unicast types, packet headers, multicast groups, virtual management addresses, ICMP rate limiting, dual-stack configuration, and source-address selection.

An IPv6 address is an interface address that

- enables forwarding of IPv6 traffic on a router interface
- uses hexadecimal fields separated by colons, and
- uses a prefix length to identify the network portion of the address.

By default, IPv6 addresses are not configured on interfaces.

The `ipv6-prefix` argument in the **ipv6 address** command must use hexadecimal values between colons. The `prefix-length` argument is a decimal value that identifies how many high-order contiguous bits compose the network portion of the address. A slash must precede the prefix length.

The maximum number of characters allowed for a prefix name is 128. Invalid prefix names are rejected.

The `ipv6-address` argument in the **ipv6 address link-local** command must use hexadecimal values between colons.

Larger IPv6 address space

Explains how 128-bit IPv6 addressing increases the available address space and supports more globally unique addresses than 32-bit IPv4 addressing.

Use this reference to compare the IPv6 address space with IPv4 and identify the operational effects of the larger IPv6 address space.

IPv6 address space characteristics include the following details:

- IPv6 increases network address bits from 32 bits in IPv4 to 128 bits in IPv6.
- IPv6 supports a much larger supply of globally unique IP addresses.
- Globally unique IPv6 addresses support global reachability and end-to-end security for networked devices.
- The flexibility of the IPv6 address space reduces the need for private addresses and Network Address Translation.
- Mobile Internet-enabled devices, home-area networks, and wireless data services drive demand for globally unique IP addresses.
- The IPv6 address space enables application protocols that do not require special processing by border routers at the edge of networks.

IPv6 address formats

Describes IPv6 notation rules, including hexadecimal fields, compressed formats, prefix notation, special addresses, and IPv4-compatible address representation used in configuration examples.

An IPv6 address format is a text representation that

- uses the format `x:x:x:x:x:x:x:x`
- allows one double colon to compress the longest sequence of zero fields, and
- uses a slash and prefix length to represent contiguous blocks of IPv6 address space.

The hexadecimal letters in IPv6 addresses are not case-sensitive. Two colons, `::`, can be used only once in an IPv6 address.

Table 7: Compressed IPv6 address formats

IPv6 address type	Preferred format	Compressed format
Unicast	`2001:0DB8:0:0:0:800:200C:417A`	`2001:DB8::800:200C:417A`
Multicast	`FF01:0:0:0:0:0:0:101`	`FF01::101`
Loopback	`0:0:0:0:0:0:0:1`	`::1`
Unspecified	`0:0:0:0:0:0:0:0`	`::`

The IPv6 loopback address can be used by a node to send an IPv6 packet to itself. The loopback address cannot be assigned to a physical interface, and IPv6 routers do not forward packets that have the IPv6 loopback address as the source or destination address.

The unspecified IPv6 address indicates the absence of an IPv6 address. The unspecified address cannot be assigned to an interface and must not be used as a destination address in IPv6 packets or the IPv6 routing header.

An IPv6 address prefix uses the format `ipv6-prefix/prefix-length`. For example, `2001:0DB8:8086:6502::/32` is a valid IPv6 prefix.

IPv6 unicast addresses

Describes IPv6 unicast address types, including aggregatable global, link-local, and IPv4-compatible addresses, with their addressing roles and configuration examples for interfaces.

An IPv6 unicast address is an address that

- identifies one interface on one node
- delivers a packet to the interface identified by that address, and
- can be used for global, site-local, link-local, or IPv4-compatible IPv6 addressing.

Cisco IOS XR software supports these IPv6 unicast address types:

- Aggregatable global addresses
- Site-local addresses
- Link-local addresses
- IPv4-compatible IPv6 addresses

Aggregatable global IPv6 addresses

Describes globally routable IPv6 unicast addressing, including routing aggregation, global routing prefixes, subnet IDs, interface ID requirements, EUI-64 construction, PPP negotiation, and fallback behavior.

An aggregatable global IPv6 address is an IPv6 unicast address that

- comes from the aggregatable global unicast prefix
- supports strict aggregation of routing prefixes to limit global routing table entries, and
- is used on links that are aggregated upward through organizations and Internet service providers.

Routing aggregation

The structure of aggregatable global IPv6 addresses enables strict aggregation of routing prefixes. This aggregation limits the number of routing table entries in the global routing table.

Aggregatable global addresses are used on links that are aggregated upward through organizations and eventually to Internet service providers.

Address structure

Aggregatable global IPv6 addresses are defined by these parts:

- Global routing prefix
- Subnet ID
- Interface ID

Except for addresses that start with binary 000, all global unicast addresses have a 64-bit interface ID. The current global unicast address allocation uses the address range that starts with binary value 001, represented as `2000::/3`.

Addresses with a prefix from `2000::/3` through `E000::/3` are required to have 64-bit interface identifiers in the extended universal identifier 64-bit format.

The Internet Assigned Numbers Authority allocates IPv6 address space in the range `2000::/16` to regional registries.

Global routing prefix and subnet ID

The aggregatable global address typically consists of a 48-bit global routing prefix and a 16-bit subnet ID or Site-Level Aggregator.

In the IPv6 aggregatable global unicast address format document, RFC 2374, the global routing prefix included two additional hierarchically structured fields: Top-Level Aggregator and Next-Level Aggregator. The IETF removed the Top-Level Aggregator and Next-Level Aggregator fields from the RFCs because these fields are policy-based.

Some IPv6 networks that were deployed before the change might still use networks based on the older RFC 2374 architecture.

The 16-bit subnet ID can be used by individual organizations to create a local addressing hierarchy and identify subnets. A subnet ID is similar to a subnet in IPv4, except that an organization with an IPv6 subnet ID can support up to 65,535 individual subnets.

Interface ID requirements

An interface ID identifies interfaces on a link. The interface ID must be unique to the link and might also be unique over a broader scope.

In many cases, an interface ID is the same as, or based on, the link-layer address of an interface.

Interface IDs used in aggregatable global unicast and other IPv6 address types must be 64 bits long and constructed in the modified EUI-64 format.

Modified EUI-64 interface ID construction

For IEEE 802 interface types, such as Ethernet and FDDI interfaces, the modified EUI-64 interface ID is constructed from the 48-bit MAC address:

- The first three octets, or 24 bits, are taken from the Organizationally Unique Identifier of the MAC address.
- The fourth and fifth octets, or 16 bits, use the fixed hexadecimal value `FFFE`.
- The last three octets, or 24 bits, are taken from the last three octets of the MAC address.
- The Universal/Local bit, which is the seventh bit of the first octet, is set to indicate whether the IPv6 interface identifier is locally administered or globally unique.

A Universal/Local bit value of 0 indicates a locally administered identifier. A value of 1 indicates a globally unique IPv6 interface identifier.

For tunnel interface types that are used with IPv6 overlay tunnels, the interface ID is the IPv4 address assigned to the tunnel interface with all zeros in the high-order 32 bits of the identifier.

PPP and fallback interface ID behavior

For interfaces that use Point-to-Point Protocol, the interfaces at both ends of the connection might have the same MAC address. In this case, the interface identifiers used at both ends of the connection are negotiated, and reconstructed if necessary, until both identifiers are unique.

The first MAC address in the router is used to construct the identifier for interfaces that use Point-to-Point Protocol.

If no IEEE 802 interface types exist in the router, link-local IPv6 addresses are generated on the interfaces in this sequence:

1. The router is queried for MAC addresses from the pool of MAC addresses in the router.
2. If no MAC address is available, the serial number of the route processor or line card is used to form the link-local address.

Enable IPv6

Enables IPv6 processing on an interface that does not have an explicit IPv6 address and verifies the resulting IPv6 interface state.

Enable IPv6 processing to activate IPv6 on an interface and automatically configure a link-local address.

Procedure

1. Enter interface configuration mode.

   Example

```
Router# configure
Router(config)# interface HundredGigE 0/0/0/33
```

2. Enable IPv6 processing on the interface.

   Example

```
Router(config-if)# ipv6 enable
```

3. Enable the interface and commit the configuration.

   Example

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

4. Verify the running configuration.

   Example

```
Router# show running-config interface HundredGigE 0/0/0/33
interface HundredGigE0/0/0/33
 ipv6 enable
!
```

5. Verify that IPv6 is enabled.

   Example

```
Router# show ipv6 interface HundredGigE 0/0/0/33
HundredGigE0/0/0/33 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
IPv6 is enabled, link-local address is fe80::7ae7:abff:febd:d4e4
No global unicast address is configured
Joined group address(es): ff02::1:ffbd:d4e4 ff02::2 ff02::1
MTU is 1514 (1500 is available to IPv6)
ICMP redirects are disabled
ICMP unreachables are enabled
ND DAD is enabled, number of DAD attempts 1
```

IPv6 is enabled on the interface with an automatically generated link-local address.

Configure IPv6 addresses

Configures an IPv6 global unicast address on an interface and verifies that IPv6 is enabled on the configured interface afterward.

Configure an IPv6 address on an interface to enable IPv6 forwarding and create the associated link-local address.

Procedure

1. Enter interface configuration mode.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/25
   ```

2. Assign the IPv6 address to the interface.

   Example

   ```
   Router(config-if)# ipv6 address 2001:0DB8:0:1::1/64
   ```

3. Enable the interface and commit the configuration.

   Example

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

4. Verify the running configuration.

   Example

   ```
   Router# show running-config interface HundredGigE 0/0/0/25
   interface HundredGigE0/0/0/25
    ipv6 address 2001:db8:0:1::1/64
   !
   ```

5. Verify that the interface is active and IPv6 is enabled.

   Example

   ```
   Router# show ipv6 interface HundredGigE 0/0/0/25
   HundredGigE0/0/0/25 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
   IPv6 is enabled, link-local address is fe80::7ae7:abff:febd:d4c4
   Global unicast address(es):
     2001:db8:0:1::1, subnet is 2001:db8:0:1::/64
     2001:db8:0:1:7ae7:abff:febd:d4c4, subnet is 2001:db8:0:1::/64
   Joined group address(es): ff02::1:ff00:1 ff02::1:ffbd:d4c4 ff02::2
   ff02::1
   MTU is 1514 (1500 is available to IPv6)
   ICMP redirects are disabled
   ICMP unreachables are enabled
   ND DAD is enabled, number of DAD attempts 1
   Table Id is 0xe0800000
   ```

The interface has an IPv6 address, IPv6 is enabled, and the interface has joined the required IPv6 multicast groups.

Configure IPv6 addresses with EUI-64 interface identifiers

Configures an IPv6 address that derives the low-order 64-bit interface identifier from EUI-64 information and verifies the resulting interface address.

Configure the `eui-64` keyword when only the 64-bit network prefix is specified and the interface identifier must be computed automatically.

Procedure

1. Enter interface configuration mode.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/35
   ```

2. Assign the IPv6 prefix with the **eui-64** keyword.

   Example

   ```
   Router(config-if)# ipv6 address 2001:0DB8:0:1::/64 eui-64
   ```

3. Enable the interface and commit the configuration.

   Example

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

4. Verify the running configuration.

   Example

   ```
   Router# show running-config interface HundredGigE 0/0/0/35
   interface HundredGigE0/0/0/35
    ipv6 address 2001:db8:0:1::/64 eui-64
   !
   ```

5. Verify that the interface is active and IPv6 is enabled.

   Example

   ```
   Router# show ipv6 interface HundredGigE 0/0/0/35
   HundredGigE0/0/0/35 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
   IPv6 is enabled, link-local address is fe80::7ae7:abff:febd:d4ec
   Global unicast address(es):
     2001:db8:0:1:7ae7:abff:febd:d4ec, subnet is 2001:db8:0:1::/64
   Joined group address(es): ff02::1:ffbd:d4ec ff02::2 ff02::1
   MTU is 1514 (1500 is available to IPv6)
   ICMP redirects are disabled
   ICMP unreachables are enabled
   ND DAD is enabled, number of DAD attempts 1
   ```

The interface has an IPv6 address whose interface identifier is generated automatically from EUI-64 information.

Link-local IPv6 addresses

Describes local-link IPv6 addressing, including automatic address generation, packet forwarding limits, route resolution, and manual configuration when configured on an interface.

A link-local IPv6 address is an IPv6 unicast address that

- uses the link-local prefix `FE80::/10`
- can be automatically configured on an interface, and
- is used by neighbor discovery and stateless autoconfiguration.

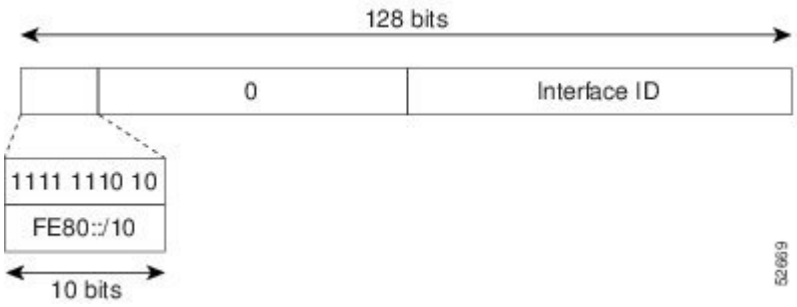
Nodes on a local link can use link-local addresses to communicate without site-local or globally unique addresses.

IPv6 routers must not forward packets that have link-local source or destination addresses to other links.

If no IEEE 802 interface types are in the router, link-local IPv6 addresses are generated by querying the router for MAC addresses. If no MAC address is available, the serial number of the route processor or line card is used to form the link-local address.

This figure displays the address format of a link-local IPv6 address.

Figure 2: Link-local address format



Configure link-local IPv6 addresses

Configures a link-local IPv6 address on an interface and verifies that IPv6 is enabled with the configured link-local address afterward.

Configure the `link-local` keyword when an interface must use a specific link-local IPv6 address instead of the automatically configured link-local address.

Procedure

1. Enter interface configuration mode.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/34
   ```

2. Assign the link-local IPv6 address.

Example

```
Router(config-if)# ipv6 address FE80::260:3EFF:FE11:6770 link-local
```

3. Enable the interface and commit the configuration.

Example

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

4. Verify the running configuration.

Example

```
Router# show running-config interface HundredGigE 0/0/0/34
interface HundredGigE0/0/0/34
 ipv6 address fe80::260:3eff:fe11:6770 link-local
!
```

5. Verify that IPv6 is enabled with the link-local address.

Example

```
Router# show ipv6 interface HundredGigE 0/0/0/34
HundredGigE0/0/0/34 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
IPv6 is enabled, link-local address is fe80::260:3eff:fe11:6770
No global unicast address is configured
Joined group address(es): ff02::1:ff11:6770 ff02::2 ff02::1
MTU is 1514 (1500 is available to IPv6)
ICMP redirects are disabled
ICMP unreachables are enabled
ND DAD is enabled, number of DAD attempts 1
```

IPv4-compatible IPv6 addresses

Describes IPv6 addresses that embed IPv4 addresses in the low-order 32 bits, including their use by dual-stack nodes and automatic tunnels.

An IPv4-compatible IPv6 address is an IPv6 unicast address that

- has zeros in the high-order 96 bits
- contains an IPv4 address in the low-order 32 bits, and
- is assigned to nodes that support both IPv4 and IPv6 protocol stacks.

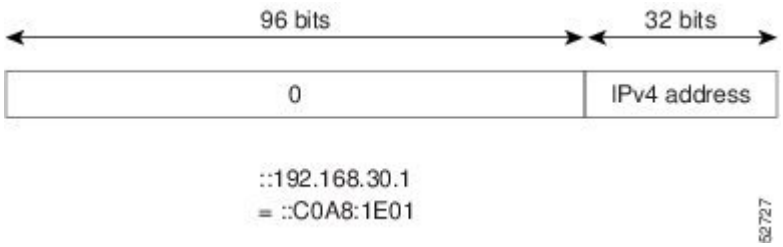
The format of an IPv4-compatible IPv6 address is `0:0:0:0:0:0:A.B.C.D` or `::A.B.C.D`.

The entire 128-bit IPv4-compatible IPv6 address is used as the IPv6 address of a node. The embedded IPv4 address is used as the IPv4 address of the node.

IPv4-compatible IPv6 addresses are used in automatic tunnels.

This figure displays the format of an IPv4-compatible IPv6 address.

Figure 3: IPv4-compatible IPv6 address format



IPv6 packet headers

Describes the simplified IPv6 base header and optional extension headers, including key fields, header chaining, and common extension header types.

An IPv6 packet header is a network-layer header that

- uses eight fields in the basic IPv6 packet header
- has a total basic header size of 40 octets, and
- can be followed by optional extension headers and the packet data.

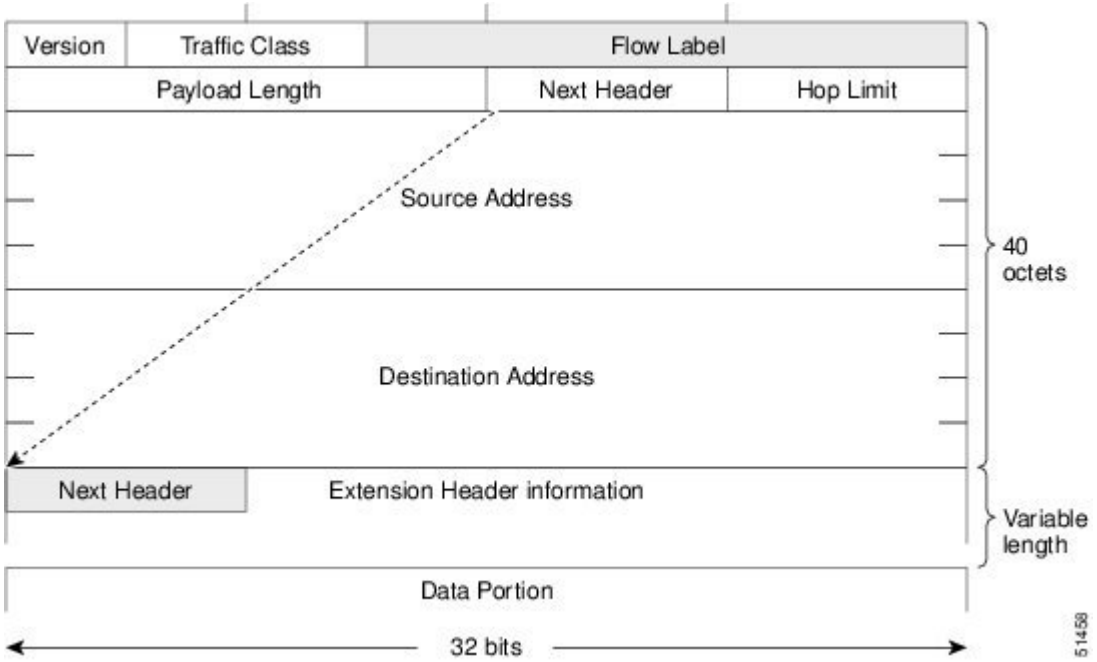
IPv6 removes fields from the IPv4 packet header because IPv6 fragmentation is handled by the packet source and network-layer checksums are not used. Checksums are used at the data-link and transport layers. The basic IPv6 packet header and options fields are aligned to 64 bits.

Table 8: Basic IPv6 packet header fields

Field	Description
Version	Identifies the packet as IPv6 by using the value 6.
Traffic Class	Tags packets with a traffic class used in differentiated services.
Flow Label	Tags packets with a specific flow that differentiates packets at the network layer.
Payload Length	Identifies the total length of the data portion of the packet.
Next Header	Identifies the type of information that follows the basic IPv6 header, such as a transport-layer packet or extension header.
Hop Limit	Specifies the maximum number of routers an IPv6 packet can pass through before the packet is considered invalid.
Source Address	Contains a 128-bit IPv6 source address.
Destination Address	Contains a 128-bit IPv6 destination address.

There is no fixed number of extension headers in an IPv6 packet. Extension headers form a chain, and each extension header is identified by the `Next Header` field of the previous header.

Figure 4: IPv6 packet header format



IPv6 extension header chain

The optional extension headers and the packet data follow the eight fields of the basic IPv6 packet header.

If extension headers are present, each extension header is aligned to 64 bits. An IPv6 packet does not have a fixed number of extension headers. The extension headers form a chain, and each extension header is identified by the `Next Header` field of the previous header.

Typically, the final extension header has a `Next Header` field value for a transport-layer protocol, such as `TCP` or `UDP`.

Figure 5: IPv6 extension header format

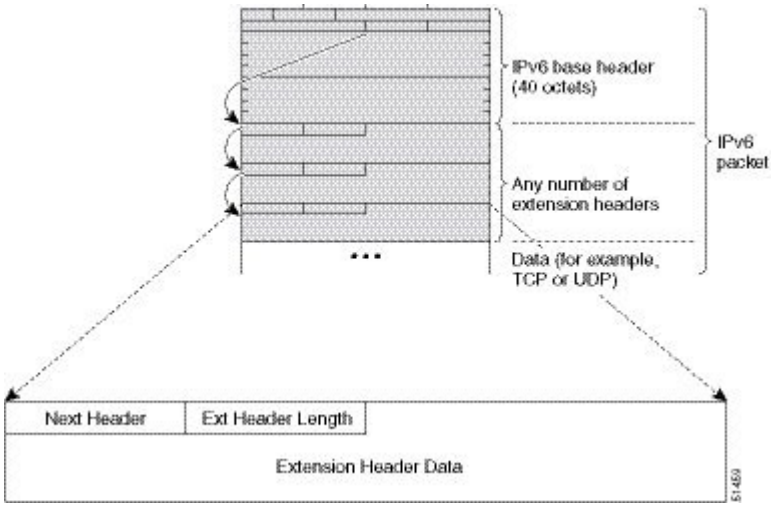


Table 9: IPv6 extension header types

Header type	Next Header value	Description
Hop-by-hop options header	0	Processed by all hops in the packet path.

Header type	Next Header value	Description
Destination options header	60	The destination options header can appear in either of these positions: • After a hop-by-hop options header: The destination options header is processed at the final destination and at each visited address specified by a routing header. • After an Encapsulating Security Payload (ESP) header: The destination options header is processed only at the final destination.
Routing header	43	Used for source routing.
Fragment header	44	Used when a source must fragment a packet that is larger than the path MTU. The Fragment header is used in each fragmented packet.
Authentication header and ESP header	51 and 50	Used by IPSec to provide authentication, integrity, and confidentiality. These headers are identical for both IPv4 and IPv6.
Upper-layer header	6 (TCP) and 17 (UDP)	Identifies TCP or UDP transport headers.
Mobility header	Assigned by IANA	Used by mobile nodes, correspondent nodes, and home agents for mobility binding messages.

IPv6 multicast groups

Describes the IPv6 multicast groups that enabled interfaces join automatically and includes the task for enabling IPv6 processing without an explicit address.

IPv6 multicast groups are link-scoped group addresses that

- are joined automatically when IPv6 is configured on an interface
- support neighbor discovery and router discovery, and
- include solicited-node, all-nodes, and all-routers groups.

An IPv6 address must be configured on an interface for the interface to forward IPv6 traffic. Configuring a global IPv6 address automatically configures a link-local address and activates IPv6 on the interface.

The configured interface automatically joins these required multicast groups for the link:

- Solicited-node multicast group `FF02:0:0:0:0:1:FF00::/104` for each unicast address assigned to the interface
- All-nodes link-local multicast group `FF02::1`
- All-routers link-local multicast group `FF02::2`

The solicited-node multicast address is used in the neighbor discovery process.

Configure IPv4 and IPv6 protocol stacks

Configures one interface with IPv4 and IPv6 addresses and verifies that the interface forwards traffic for both protocol stacks simultaneously.

Configure both IPv4 and IPv6 on an interface when the interface must send and receive data on IPv4 and IPv6 networks.

Procedure

1. Enter interface configuration mode.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/31
   ```

2. Assign the IPv4 address.

   Example

   ```
   Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
   ```

3. Assign the IPv6 address.

   Example

   ```
   Router(config-if)# ipv6 address 2001:0DB8:c18:1::3/64
   ```

4. Enable the interface and commit the configuration.

   Example

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

5. Verify the running configuration.

   Example

   ```
   Router# show running-config interface HundredGigE 0/0/0/31
   interface HundredGigE0/0/0/31
    ipv4 address 192.0.2.1 255.255.255.0
    ipv6 address 2001:db8:c18:1::3/64
   !
   ```

6. Verify that IPv4 is enabled.

   Example

   ```
   Router# show ipv4 interface HundredGigE 0/0/0/31
   HundredGigE0/0/0/31 is Up, ipv4 protocol is Up
   Vrf is default (vrfid 0x60000000)
   Internet address is 192.0.2.1/24
   MTU is 1514 (1500 is available to IP)
   Helper address is not set
   Directed broadcast forwarding is disabled
   ICMP redirects are never sent
   ICMP unreachables are always sent
   Table Id is 0xe0000000
   ```

7. Verify that IPv6 is enabled.

Example

```
Router# show ipv6 interface HundredGigE 0/0/0/31
HundredGigE0/0/0/31 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
IPv6 is enabled, link-local address is fe80::7ae7:abff:febd:d4dc
Global unicast address(es):
  2001:db8:c18:1::3, subnet is 2001:db8:c18:1::/64
Joined group address(es): ff02::1:ff00:3 ff02::1:ffbd:d4dc ff02::2
ff02::1
MTU is 1514 (1500 is available to IPv6)
ICMP redirects are disabled
ICMP unreachables are enabled
ND DAD is enabled, number of DAD attempts 1
```

The interface forwards both IPv4 and IPv6 traffic.

IPv6 virtual addresses

Describes stable IPv6 management access across RP failover, including common-subnet requirements, per-VRF support, transport source-address substitution, and protocol source-address behavior.

An IPv6 virtual address is a management address that

- lets you access the router from a single IPv6 address on a management network without knowing which route processor is active
- persists across route processor failover, and
- must share a common IPv6 subnet with a Management Ethernet interface on both route processors.

VRF support

The `vrf` keyword supports virtual addresses on a per-VRF basis.

Source address selection

The `use-as-src-addr` keyword eliminates the need to configure a loopback interface as the update source for management applications.

When an update source is not configured, management applications allow transport processes such as `TCP`, `UDP`, and `raw_ip` to select a suitable source address. The transport processes consult the FIB to select the source address.

If a Management Ethernet IPv6 address is selected and `use-as-src-addr` is configured, the transport process substitutes the Management Ethernet IPv6 address with the relevant virtual IPv6 address. This behavior works across route processor switchovers.

If `use-as-src-addr` is not configured, the source address selected by the transport process can change after failover, and the network management system might not be able to handle the change.

Protocol source address behavior

Protocol configuration, such as **tacacs source-interface**, **snmp-server trap-source**, **ntp source**, and **logging source-interface**, does not use the virtual management IPv6 address as its source by default.

Use the **ipv6 virtual address use-as-src-addr** command to ensure that the protocol uses the virtual IPv6 address as its source address.

Alternatively, configure a loopback address with the required IPv6 address and set it as the source interface for the protocol, such as TACACS+, by using **tacacs source-interface**.

Configure IPv6 virtual addresses

Configures an IPv6 virtual management address and verifies the Management Ethernet interface configuration used for stable access across RP failover.

Configure IPv6 virtual addresses to provide stable management access across route processor failover.

The virtual IPv6 address must share a common IPv6 subnet with a Management Ethernet interface on both route processors.

Procedure

1. Configure the IPv6 virtual address.

   Example

   ```
   Router# configure terminal
   Router(config)# ipv6 virtual address 2001:0DB8:0:1::1/64
   Router(config)# commit
   ```

2. Configure the Management Ethernet interface in the same subnet.

   Example

   ```
   Router# configure terminal
   Router(config)# ipv6 virtual address 2001:DB8:0:1::1/64
   Router(config)# interface MgmtEth 0/RP0/CPU0/0
   Router(config-if)# ipv6 address 2001:0DB8:0:1::1/64
   Router(config-if)# no shutdown
   Router(config-if)# commit
   Router(config-if)# exit
   ```

3. Verify the Management Ethernet interface configuration.

   Example

   ```
   Router# show running-config interface MgmtEth 0/RP0/CPU0/0
   interface MgmtEth0/RP0/CPU0/0
    ipv6 address 2001:db8:0:1::1/64
   !
   ```

4. Configure the IPv6 virtual address for a VRF.

   Example

   ```
   Router# configure terminal
   Router(config)# ipv6 virtual address vrf Test 2001:0DB8:0:1::1/64
   Router(config)# interface MgmtEth 0/RP0/CPU0/0
   Router(config-if)# ipv6 address 2001:0DB8:0:1::1/64
   Router(config-if)# no shutdown
   Router(config-if)# commit
   ```

5. Verify the VRF interface configuration.

Example

```
Router# show running-config interface MgmtEth 0/RP0/CPU0/0
interface MgmtEth0/RP0/CPU0/0
 ipv6 address 2001:db8:0:1::1/64
!
```

IPv6 ICMP rate limiting

Describes token-bucket control for IPv6 ICMP error messages, including bucket size, token interval, default values, and the configuration task workflow.

IPv6 ICMP rate limiting is an error-message control feature that

- limits the rate at which IPv6 ICMP error messages are sent on the network
- uses tokens to represent permission to send error messages, and
- allows a burst of error messages until the token bucket is empty.

A token bucket scheme lets applications such as traceroute receive replies to a group of requests sent in rapid succession. Each error message removes one token from the bucket. When the bucket is empty, IPv6 ICMP error messages are not sent until a new token is added.

The token bucket algorithm is more flexible than a fixed time interval scheme and does not increase the average rate-limiting time interval.

Configure IPv6 ICMP rate limiting

Configures the token interval and bucket size for IPv6 ICMP error messages, then verifies the applied rate-limiting values in interface output.

Configure IPv6 ICMP rate limiting to control how often IPv6 ICMP error messages are sent.

The `milliseconds` argument specifies the interval between tokens being added to the bucket. The optional `bucketsize` argument defines the maximum number of tokens stored in the bucket.

Procedure

1. Configure the interval and bucket size for IPv6 ICMP error messages.

   Example

   ```
   Router# configure
   Router(config)# ipv6 icmp error-interval 50 20
   Router(config)# commit
   ```

2. Verify the IPv6 ICMP error interval configuration.

   Example

   ```
   Router# show running-config | include ipv6 icmp
   ipv6 icmp error-interval 50 20
   ```

The IPv6 ICMP error interval is configured with a 50 millisecond token interval and a bucket size of 20 tokens.

Flexible source IP

Describes ICMP response source-address selection, including RFC-compliant behavior and the configuration task used to enable flexible source IP for ICMP responses.

Flexible source IP is an ICMP source-address selection behavior that

- selects the source IP address in ICMP response packets
- helps respond to packet-forwarding failures, and
- can be configured for RFC-compliant IPv4 ICMP source address selection.

The related command family includes **icmp ipv4 source vrf**.

Configure flexible source IP for ICMP

Configures RFC-compliant source-address selection for ICMP response packets and verifies that generated responses use the expected source address in command output.

Configure flexible source IP for ICMP when ICMP response packets must use RFC-compliant source address selection.

Procedure

1. Enable RFC-compliant source address selection for IPv4 ICMP.

   Example

   ```
   Router# configure
   Router(config)# icmp ipv4 source rfc
   Router(config)# commit
   ```

2. Verify the source address selection configuration.

   Example

   ```
   Router# show running-config | include source rfc
   icmp ipv4 source rfc
   ```

RFC-compliant source address selection is enabled for IPv4 ICMP response packets.

3 Network Stack Traffic Control and Address Management

Topics:

Summarizes IPv6 path MTU controls, TCP MSS adjustment, neighbor discovery, IP address conflict resolution, VRF fallback, redirects, and local station MAC address configuration.

Use this chapter to control network stack traffic behavior, manage address conflicts, and configure IPv6 neighbor discovery and router-specific addressing functions.

The chapter organizes network stack traffic control and address management into these content areas:

Table 10: Traffic control and address management content areas

Content area	Included information
IPv6 path MTU discovery	Path maximum transmission unit (MTU) discovery behavior and the requirement to use TCP applications for path MTU discovery.
TCP MSS adjustment	TCP maximum segment size (MSS) adjustment and supported egress-interface requirements.
IPv6 neighbor discovery	Neighbor discovery behavior, neighbor solicitation messages, router advertisements, neighbor redirects, router advertisement suppression, and unicast router advertisement messages.
IP address conflict resolution	IP Address Repository Manager (IP ARM) conflict resolution policies, address repository behavior, and route tags for connected routes.
VRF fallback and redirect	Virtual routing and forwarding (VRF) fallback and redirect behavior and operational restrictions.
Local station MAC addresses	Local station media access control (MAC) addresses and the restriction for Cisco Silicon One Q100 ASICs.

The topics progress from packet-size controls to IPv6 neighbor discovery, address conflict resolution, routing-instance fallback, and local station address configuration. Requirement and restriction information appears before the related configuration tasks.

Path MTU discovery for IPv6

Explains how IPv6 path MTU discovery dynamically identifies MTU differences along a path so the source can handle fragmentation, and describes its TCP-only support and key IPv6 MTU requirements.

Path MTU discovery for IPv6 is a host behavior that

- discovers MTU differences for each link along a data path
- lets the IPv6 packet source handle fragmentation when a link MTU is too small, and
- helps save router processing resources by keeping fragmentation at the source.

Path MTU discovery is supported only for applications that use TCP.

In IPv4, the minimum link MTU is 68 octets. In IPv6, the minimum link MTU is 1280 octets. Use an MTU value of 1500 octets for IPv6 links.

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
Path MTU discovery for IPv6	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on: • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Path MTU discovery for IPv6	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 Information	Feature Description
Path MTU discovery for IPv6	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*) The network stack now supports simultaneous IPv4 and IPv6 operations, enhancing protocol flexibility and network efficiency. This dual-stack approach ensures interoperability and smooth transition between IPv4 and IPv6, accommodating diverse application requirements. It allows for more efficient use of IP addresses and prepares networks for future growth, making it easier to handle various network scenarios without service disruption. *Previously this feature was supported on Q200 and Q100. *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

TCP MSS adjustment

Explains how TCP MSS adjustment modifies the maximum segment size of transit IPv4 and IPv6 TCP SYN and SYN-ACK packets to keep flows within the interface MTU and prevent fragmentation during session establishment.

TCP maximum segment size (MSS) adjustment is a traffic management mechanism that

- modifies the maximum segment size value on transit IPv4 and IPv6 packets
- supports TCP SYN and TCP SYN-ACK packets on configured interfaces, and
- helps prevent fragmentation by keeping TCP flows below the interface MTU.

Fragmentation prevention

TCP MSS adjustment modifies the maximum segment size on transit IPv4 and IPv6 TCP SYN and TCP SYN-ACK packets. This behavior helps ensure that TCP segments remain within the interface MTU during session establishment.

Table 12: Feature History Table

Feature Name	Release Information	Feature Description
TCP MSS adjustment	Release 26.2.1	Introduced in this release on: Fixed Systems (8200, 8700, 8010)(select variants only*), Modular Systems (8800 [LC ASIC: P100])(select variants only*) You can now prevent packet fragmentation and ensure TCP segments remain within the interface MTU by using TCP MSS adjustment. This feature modifies the Maximum Segment Size on transit IPv4 and IPv6 TCP SYN and TCP SYN-ACK packets. This helps ensure optimal traffic flow during session establishment. *This feature is now supported on: Line cards: • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM Fixed systems: • 8212-48FH-M • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-24Y8L2FH-I • 8011-24X • 8011-12G12X4Y • 8711-32FH-M • 8711-48Z-M • 8712-MOD-M

Guidelines and restrictions for TCP MSS adjustment

Describes the guidelines and restrictions for TCP MSS adjustment, including supported interfaces, applicable packet types, egress-only configuration, transit IPv4 and IPv6 packet support, and per-network-processor configuration limits.

Use these guidelines and restrictions before you configure TCP MSS adjustment.

Guidelines for TCP MSS adjustment

- Use TCP MSS adjustment on these supported interface types:
  - GRE tunnel interfaces
  - VLAN subinterfaces on physical Ethernet interfaces
- Configure **ipv4 tcp-mss-adjust enable**, **ipv6 tcp-mss-adjust enable**, or both commands on an interface. Any of these configurations has the same effect.

- The feature applies to TCP SYN and TCP SYN-ACK packets that are encapsulated in IPv4 or IPv6 frames.

Restrictions of TCP MSS adjustment

- Use TCP MSS adjustment for transit IPv4 and IPv6 packets only.
- Configure TCP MSS adjustment on the egress interface. If you configure TCP MSS adjustment on the ingress interface, the feature does not work.
- Configure only one TCP MSS adjustment value per network processor (NP).
- To apply the same TCP MSS adjustment value across all locations and all NPs simultaneously, use the **hw-module** command with the `location all` and `np all` options.

Configure TCP MSS adjustment

Configures TCP MSS adjustment on an egress interface for IPv4 packets, IPv6 packets, or both, and sets the network-processor MSS value to keep transit TCP flows below the interface MTU and prevent fragmentation during session establishment.

Configure TCP MSS adjustment to keep transit TCP flows below the GRE tunnel interface or VLAN subinterface MTU and prevent fragmentation during session establishment.

You can configure TCP MSS adjustment for IPv4 packets, IPv6 packets, or both on the egress interface.

Procedure

1. Enable TCP MSS adjustment for IPv4 packets, IPv6 packets, or both on the egress interface.

   Example

   ```
   Router# configure
   Router(config)# interface GigabitEthernet 0/2/0/0.100
   Router(config-if)# ipv4 tcp-mss-adjust enable
   Router(config-if)# ipv6 tcp-mss-adjust enable
   Router(config-if)# commit
   ```

2. Exit interface configuration mode and configure the TCP MSS value for a network processor.

   Example

   ```
   Router(config-if)# exit
   Router(config)# hw-module location 0/0/CPU0 tcp-mss-adjust np 1 value 1300
   Router(config)# commit
   ```

3. Alternatively, apply the same TCP MSS adjustment value across all locations and all network processors.

   Example

   ```
   Router(config-if)# exit
   Router(config)# hw-module location all tcp-mss-adjust np all value 1300
   Router(config)# commit
   ```

IPv6 neighbor discovery

Explains how IPv6 neighbor discovery operates on the local link to resolve neighbor addresses, verify reachability, and track neighboring routers by using ICMP-based discovery, advertisement, solicitation, and redirect messages.

IPv6 neighbor discovery is an IPv6 local-link process that

- determines the link-layer address of a neighbor on the same local link
- verifies neighbor reachability, and
- keeps track of neighboring routers.

Neighbor discovery messages

IPv6 neighbor discovery uses Internet Control Message Protocol (ICMP) messages and solicited-node multicast addresses. Neighbor solicitation, neighbor advertisement, router advertisement, router solicitation, and neighbor redirect messages support the neighbor discovery process.

Table 13: Feature History Table

Feature Name	Release Information	Feature Description
IPv6 Neighbor Discovery	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
IPv6 Neighbor Discovery	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 Information	Feature Description
IPv6 Neighbor Discovery	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*) The integrated network stack allows routers to support both IPv4 and IPv6 protocols, facilitating a seamless transition and coexistence on the same network infrastructure. This feature enhances connectivity options, enabling devices to handle dual-protocol traffic efficiently. By supporting dual stack, the network can leverage the expansive address space of IPv6 while maintaining compatibility with existing IPv4 systems, ensuring smooth operation and future readiness. *Previously this feature was supported on Q200 and Q100. *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

How IPv6 neighbor discovery works

Explains how IPv6 neighbor discovery uses ICMP messages and solicited-node multicast addresses to resolve neighbor link-layer addresses, verify reachability, and track neighboring routers on the local link.

IPv6 neighbor discovery operates on the local link. It supports address resolution, reachability verification, router discovery, and redirect behavior.

Summary

The IPv6 neighbor discovery process uses these components:

- ICMP messages: Carry neighbor solicitation, neighbor advertisement, router advertisement, router solicitation, and neighbor redirect information.
- Solicited-node multicast addresses: Help identify neighbors on the same local link.
- Neighboring routers: Are tracked so IPv6 nodes can select reachable routers on the local link.

Workflow

These stages describe how IPv6 neighbor discovery works.

1. An IPv6 node uses neighbor discovery messages to determine the link-layer address of a neighbor on the same local link.
2. The node verifies whether the neighbor remains reachable after the link-layer address is known.
3. The node tracks neighboring routers and uses router advertisement, router solicitation, and redirect behavior to maintain efficient local-link forwarding.

Result

The IPv6 node can resolve neighbor link-layer addresses, verify neighbor reachability, and track neighboring routers on the local link.

IPv6 neighbor solicitation messages

Describes IPv6 neighbor solicitation messages and explains how they resolve link-layer addresses, verify neighbor reachability, and support duplicate address detection through neighbor advertisement exchanges on the local link.

An IPv6 neighbor solicitation message is an ICMP message that

- uses Type value 135 in the ICMP packet header
- is sent on the local link to determine the link-layer address of another node, and
- can verify neighbor reachability and the uniqueness of tentative unicast IPv6 addresses.

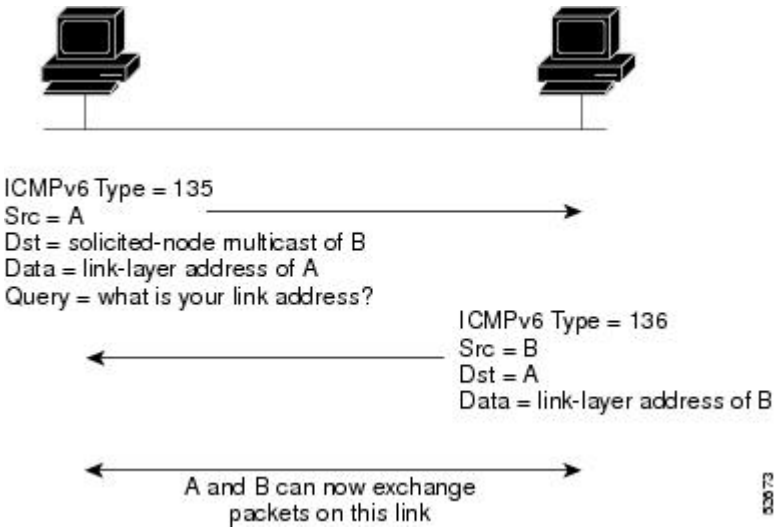
Table 14: Feature History Table

Feature Name	Release Information	Feature Description
IPv6 Neighbor Solicitation Message	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.
IPv6 Neighbor Solicitation Message	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*) The dual protocol stack enables devices to concurrently operate IPv4 and IPv6, allowing seamless communication across both address families. This feature supports gradual network transition strategies and ensures continued compatibility with legacy IPv4 systems while embracing IPv6's broader address capabilities. It optimizes network resource allocation and provides flexibility to accommodate evolving network demands without service disruption. *Previously this feature was supported on Q200 and Q100. *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Address fields

This figure displays the structure of an IPv6 neighbor solicitation message.

Figure 6: IPv6 neighbor discovery—Neighbor solicitation message



When a node sends a neighbor solicitation message to determine a link-layer address, the source address is the IPv6 address of the node that sends the message.

The destination address is the solicited-node multicast address that corresponds to the IPv6 address of the destination node. The neighbor solicitation message also includes the link-layer address of the source node.

Neighbor advertisement response

After a destination node receives a neighbor solicitation message, the destination node replies by sending a neighbor advertisement message on the local link.

A neighbor advertisement message uses Type value 136 in the ICMP packet header. The source address in the neighbor advertisement message is the IPv6 address of the node interface that sends the advertisement. The destination address is the IPv6 address of the node that sent the neighbor solicitation message.

The data portion of the neighbor advertisement message includes the link-layer address of the node that sends the advertisement. After the source node receives the neighbor advertisement, the source node and destination node can communicate.

Reachability and duplicate address detection

Neighbor solicitation messages verify reachability after the link-layer address of a neighbor is known. In this use case, the destination address in the neighbor solicitation message is the unicast address of the neighbor.

Neighbor unreachability detection uses neighbor solicitation messages to identify a neighbor failure or a forward-path failure for paths between hosts and neighboring nodes. This detection is performed for neighbors that receive unicast packets and is not performed for neighbors that receive multicast packets.

A solicited neighbor advertisement message confirms that the forward path is working in both directions. A neighbor advertisement message with a solicited flag value of 0 must not be used as a positive acknowledgment that the forward path is working.

Neighbor solicitation messages are also used in stateless autoconfiguration to verify the uniqueness of unicast IPv6 addresses before the addresses are assigned to an interface. Duplicate address detection is performed first on a new link-local IPv6 address.

During duplicate address detection, the new address remains tentative. The node sends a neighbor solicitation message with an unspecified source address and the tentative link-local address in the message body. If another node is using that address, the node returns a neighbor advertisement message that contains the tentative link-local address.

If no neighbor advertisement messages are received in response to the neighbor solicitation message, and no neighbor solicitation messages are received from other nodes that are verifying the same tentative address, the node considers the tentative link-local address unique and assigns it to the interface.

Address-change advertisements

Neighbor advertisement messages are also sent when the link-layer address of a node changes on the local link. In this case, the destination address for the neighbor advertisement message is the all-nodes multicast address.

Every global or link-local IPv6 unicast address must be checked for uniqueness on the link. Until the uniqueness of the link-local address is verified, duplicate address detection is not performed on other IPv6 addresses that are associated with that link-local address.

The Cisco IOS XR implementation of duplicate address detection does not check the uniqueness of anycast addresses or global addresses that are generated from 64-bit interface identifiers.

How IPv6 neighbor solicitation works

Explains how IPv6 neighbor solicitation resolves link-layer addresses, verifies neighbor reachability, and supports duplicate address detection by using solicitation and advertisement message exchanges on the local link.

Summary

IPv6 neighbor solicitation uses these message fields and responses:

- Source address: Uses the IPv6 address of the node that sends the neighbor solicitation message.
- Destination address: Uses the solicited-node multicast address that corresponds to the IPv6 address of the destination node.
- Source link-layer address: Is included in the neighbor solicitation message.

Workflow

These stages describe how IPv6 neighbor solicitation works.

1. When a node needs the link-layer address of another node on the same local link, the source node sends a neighbor solicitation message to the solicited-node multicast address of the destination node.
2. The destination node replies with a neighbor advertisement message on the local link. The source address in the neighbor advertisement message is the IPv6 address of the interface that sends the advertisement. The destination address is the IPv6 address of the node that sent the neighbor solicitation.
3. The data portion of the neighbor advertisement message includes the link-layer address of the node that sends the advertisement. After the source node receives the neighbor advertisement, the source node and destination node can communicate.
4. After a link-layer address is known, neighbor solicitation messages can verify reachability. In this case, the destination address in the neighbor solicitation message is the unicast address of the neighbor.
5. When the link-layer address of a node changes on a local link, neighbor advertisement messages are sent to the all-nodes multicast address.
6. Neighbor unreachability detection identifies neighbor failures or forward-path failures for neighbors that receive unicast packets. Neighbor unreachability detection is not performed for neighbors that receive multicast packets.
7. A neighbor is considered reachable when a positive acknowledgment is returned. A positive acknowledgment can come from an upper-layer protocol such as TCP or from a neighbor advertisement message sent in response to a neighbor solicitation message.
8. When upper-layer acknowledgments are not available, the node probes the neighbor with unicast neighbor solicitation messages. A solicited neighbor advertisement confirms that the forward path still works in both directions.
9. An unsolicited neighbor advertisement confirms only the one-way path from the source to the destination node. A neighbor advertisement message with the solicited flag set to 0 must not be considered a positive acknowledgment that the forward path still works.

10. During duplicate address detection, a node sends a neighbor solicitation message with an unspecified source address and a tentative link-local address in the message body. If another node already uses the address, that node returns a neighbor advertisement message that contains the tentative link-local address.
11. If another node is verifying the same tentative address at the same time, that node returns a neighbor solicitation message. If no neighbor advertisement messages or competing neighbor solicitation messages are received, the node considers the tentative link-local address unique and assigns it to the interface.

Result

Neighbor solicitation and neighbor advertisement messages resolve link-layer addresses, confirm neighbor reachability, and support duplicate address detection on the local link.

IPv6 router advertisement messages

Explains how IPv6 router advertisement messages provide prefixes, default-router information, and link parameters for host autoconfiguration, and describes message contents, router solicitation responses, configurable interface-specific settings, default behavior, suppression options, and the 64-bit prefix-length requirement for stateless autoconfiguration.

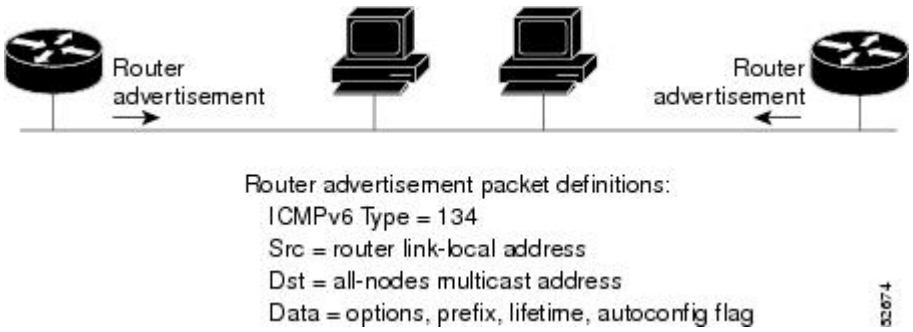
An IPv6 router advertisement message is an ICMP message that

- uses Type value 134 in the ICMP packet header
- is sent periodically from each configured IPv6 router interface, and
- is sent to the all-nodes multicast address.

Message contents

This figure shows the structure of an IPv6 router advertisement message.

Figure 7: IPv6 neighbor discovery—Router advertisement message



Router advertisement messages typically include this information:

- One or more on-link IPv6 prefixes that nodes on the local link can use to automatically configure IPv6 addresses.
- Lifetime information for each prefix included in the advertisement.
- Flags that indicate whether stateless or stateful autoconfiguration can be completed.
- Default-router information, including whether the sending router can be used as a default router and the time, in seconds, for which the router can be used as the default router.
- Additional host information, such as the hop limit and the MTU that a host should use in packets that it originates.

Router solicitation response

Router advertisement messages are also sent in response to router solicitation messages. A router solicitation message uses Type value 133 in the ICMP packet header.

Hosts send router solicitation messages during system startup so that they can autoconfigure immediately instead of waiting for the next scheduled router advertisement message.

Because router solicitation messages are usually sent during startup before a host has a configured unicast address, the source address is usually the unspecified IPv6 address, 0:0:0:0:0:0:0:0. If the host has a configured unicast address, the source address is the unicast address of the interface that sends the router solicitation message.

The destination address in a router solicitation message is the all-routers multicast address with link scope. When a router advertisement is sent in response to a router solicitation message, the destination address in the router advertisement message is the unicast address of the router solicitation source.

Configurable parameters

You can configure these router advertisement message parameters:

- The time interval between periodic router advertisement messages.
- The router lifetime value, which indicates the usefulness of a router as the default router for all nodes on a link.
- The network prefixes in use on a link.
- The time interval between neighbor solicitation message retransmissions on a link.
- The amount of time for which a node considers a neighbor reachable on a link.

Configured router advertisement parameters are specific to an interface.

Default behavior and suppression

Router advertisement transmission with default values is automatically enabled on Ethernet and FDDI interfaces. For other interface types, manually enable router advertisement transmission by using the **no ipv6 nd suppress-ra** command in interface configuration mode.

You can disable router advertisement transmission on individual interfaces by using the **ipv6 nd suppress-ra** command in interface configuration mode.

From Release 25.4.1, the **ipv6 nd suppress-ra** command is deprecated. Use the **ipv6 nd unsolicited-ra** or **ipv6 nd solicited-ra** command to suppress router advertisement messages.

Prefix length requirement

For stateless autoconfiguration to work properly, the advertised prefix length in router advertisement messages must always be 64 bits.

How IPv6 router advertisements work

Explains how IPv6 router advertisements announce prefixes and local-link parameters, respond to router solicitations, and support router discovery and stateless address autoconfiguration on configured interfaces.

A value of 134 in the Type field of the ICMP packet header identifies a router advertisement message. Router advertisement messages are periodically sent out each configured interface of an IPv6 router to the all-nodes multicast address.

Summary

Router advertisement messages can include these parameters:

- One or more on-link IPv6 prefixes that nodes on the local link can use to automatically configure IPv6 addresses.
- Lifetime information for each advertised prefix.
- Flags that indicate the type of autoconfiguration that can be completed.
- Default router information, including whether the advertising router should be used as a default router and how long it should be used.

- Additional host information, such as the hop limit and MTU that a host should use in originated packets.

The router advertisement parameters are specific to an interface.

Workflow

These stages describe how IPv6 router advertisements work.

1. An IPv6 router periodically sends router advertisement messages on configured interfaces.
2. A host can send a router solicitation message, identified by Type value 133, at system startup so that the host can autoconfigure without waiting for the next scheduled router advertisement.
3. If the host does not have a configured unicast address, the source address in the router solicitation message is usually the unspecified IPv6 address. If the host has a configured unicast address, the source address is the unicast address of the interface that sends the router solicitation message.
4. The destination address in a router solicitation message is the all-routers multicast address with link scope. When a router advertisement is sent in response to a router solicitation, the destination address in the router advertisement is the unicast source address of the router solicitation message.
5. Router advertisement messages with default values are automatically enabled on Ethernet and FDDI interfaces. For other interface types, router advertisement messages were manually enabled by using the **no ipv6 nd suppress-ra** command in interface configuration mode.
6. Router advertisement messages were disabled on individual interfaces by using the **ipv6 nd suppress-ra** command in interface configuration mode. From Release 25.4.1, the **ipv6 nd suppress-ra** command is deprecated. Use the IPv6 ND unsolicited and solicited router advertisement commands instead.
7. For stateless autoconfiguration to work properly, the advertised prefix length in router advertisement messages must always be 64 bits.

Result

Hosts receive IPv6 prefixes and local-link parameters that support router discovery and stateless address autoconfiguration.

IPv6 neighbor redirect messages

Describes IPv6 neighbor redirect messages and explains how routers use them to identify a better first-hop node for hosts, including message structure, target address requirements, redirect conditions, restrictions, and rate limiting.

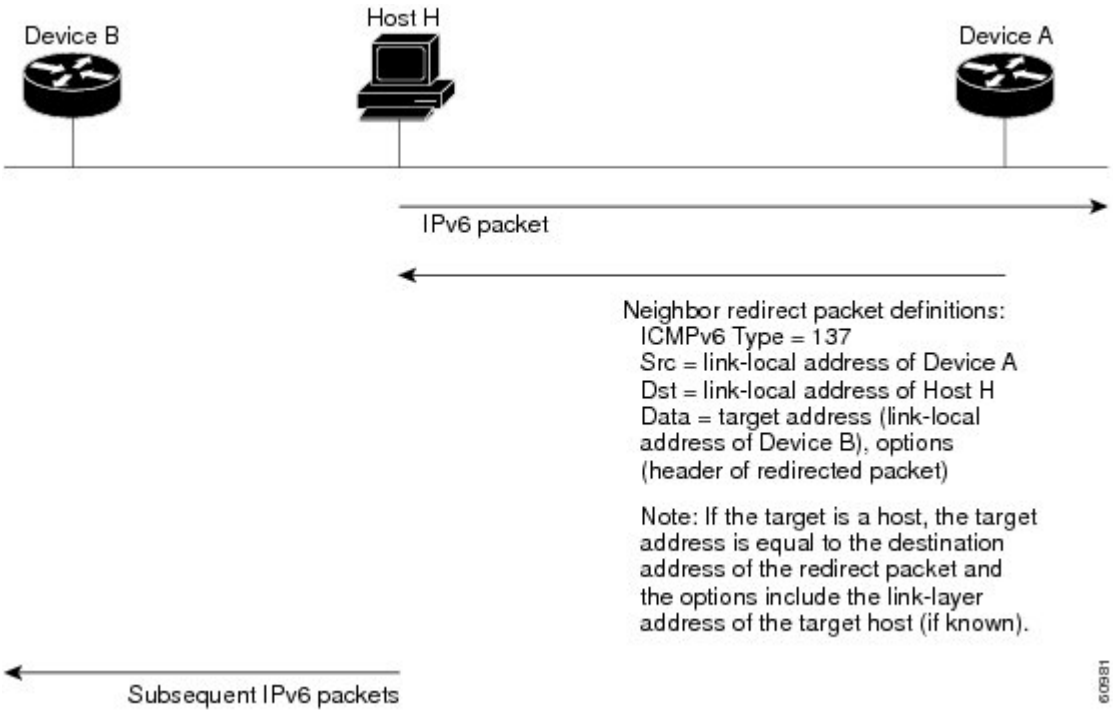
An IPv6 neighbor redirect message is an ICMP message that

- uses Type value 137 in the ICMP packet header
- is sent by routers to hosts, and
- identifies a better first-hop node on the path to a destination.

Structure of an IPv6 neighbor redirect message

This figure displays the structure of an IPv6 neighbor redirect message.

Figure 8: IPv6 neighbor discovery—Neighbor redirect message



Target address requirement

A router must be able to determine the link-local address for each neighboring router so that the target address in a redirect message identifies the neighbor router by its link-local address.

For static routing, specify the next-hop router by using the link-local address of the router. For dynamic routing, IPv6 routing protocols must exchange the link-local addresses of neighboring routers.

Redirect conditions

After forwarding a packet, a router sends a redirect message to the packet source only when these conditions are true:

- The destination address of the packet is not a multicast address.
- The packet was not addressed to the router.
- The packet is about to be sent out the interface on which it was received.
- The router determines that a better first-hop node for the packet resides on the same link as the source of the packet.
- The source address of the packet is a global IPv6 address of a neighbor on the same link or a link-local address.

Redirect restrictions

A router must not update its routing tables after receiving a neighbor redirect message. Hosts must not originate neighbor redirect messages.

Use the **ipv6 icmp error-interval** global configuration command to limit the rate at which the router generates IPv6 ICMP error messages, including neighbor redirect messages.

How IPv6 neighbor redirects work

Explains how IPv6 neighbor redirect messages inform hosts about better first-hop nodes on the local link, and describes redirect conditions, target address usage, host and router behavior, and redirect rate limiting.

A value of 137 in the Type field of the ICMP packet header identifies an IPv6 neighbor redirect message.

Summary

IPv6 neighbor redirects use these elements:

- Router: Sends a redirect message to the source of a packet when a better first-hop node exists on the same link.
- Host: Receives the redirect message and can use the better first-hop node for the destination.
- Neighbor router link-local address: Identifies the target router in the redirect message.

Workflow

These stages describe how IPv6 neighbor redirects work.

1. A router determines the link-local address for each neighboring router so that the target address in a redirect message can identify the neighbor router by its link-local address.
2. After forwarding a packet, a router sends a redirect message to the packet source only when the destination address is not multicast, the packet was not addressed to the router, the packet is about to be sent out the interface on which it was received, and a better first-hop node is on the same link as the source.
3. The router sends the redirect only when the source address of the packet is a global IPv6 address of a neighbor on the same link or a link-local address.
4. For static routing, specify the next-hop router by using the link-local address of the router. For dynamic routing, IPv6 routing protocols exchange the link-local addresses of neighboring routers.
5. A router must not update its routing tables after receiving a neighbor redirect message, and hosts must not originate neighbor redirect messages.

Result

The host can learn a better first-hop node for a destination on the local link.

What's next

Use the **ipv6 icmp error-interval** command to limit the rate at which the router generates IPv6 ICMP error messages, including neighbor redirect messages.

IPv6 ND RA suppression and unicast RA messages

Explains how IPv6 ND router advertisement suppression and unicast response support provides granular control of unsolicited and solicited router advertisement traffic, and describes its purpose, default behavior, new and replacement commands, and support for unicast solicited responses.

IPv6 Neighbor Discovery (ND) router advertisement (RA) control is a traffic-control capability that

- suppresses unsolicited IPv6 ND router advertisement messages
- suppresses solicited IPv6 ND router advertisement messages, and
- enables solicited router advertisement responses to be sent as unicast messages.

Table 15: Feature History Table

Feature Name	Release Information	Feature Description
Suppression of IPv6 ND RA messages and enabling unicast RA messages	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]), 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can granularly control IPv6 ND RA traffic to reduce network traffic and improve control by • independently suppressing unsolicited and solicited IPv6 ND RA messages and • by enabling solicited unicast RA responses. The feature introduces these changes: CLI: • **ipv6 nd unsolicited-ra disable** • **ipv6 nd solicited-ra disable** • **ipv6 nd solicited-ra unicast** YANG Data Model: `openconfig-if-ip.yang` (see GitHub, YANG Data Models Navigator)

Router advertisement role

IPv6 router advertisements inform hosts in the same network about route prefixes, default gateway addresses, router lifetimes, DNS addresses, suggested MTUs, and other network parameters.

Router advertisements allow hosts to obtain network prefixes and program addresses automatically by using stateless address autoconfiguration.

IPv6 router advertisements are part of the Neighbor Discovery Protocol defined in RFC 4861.

Router advertisement message types

Cisco IOS XR software sends these router advertisement message types:

- Unsolicited router advertisements: Routers send these messages periodically to announce their presence and network parameters.
- Solicited router advertisements: Routers send these messages in response to on-demand router solicitation messages from hosts.

By default, Cisco IOS XR software enables router advertisement transmission on all IPv6 interfaces, except management interfaces.

Deprecated and replacement IPv6 ND RA commands

This table lists the deprecated IPv6 ND router advertisement commands and the replacement commands introduced in Cisco IOS XR Release 25.4.1.

Table 16: Deprecated and replacement commands

Deprecated command	Replacement command	Command effect
ipv6 nd suppress-ra	**ipv6 nd unsolicited-ra disable**	Suppresses unsolicited router advertisement messages.
Not applicable	**ipv6 nd solicited-ra disable**	Suppresses solicited router advertisement messages.
ipv6 nd unicast-ra	**ipv6 nd solicited-ra unicast**	Enables solicited unicast router advertisement responses.

Limitations for IPv6 ND RA suppression and unicast RA message support

Describes the limitations for IPv6 ND router advertisement suppression and unicast router advertisement support, including default router advertisement behavior, suppression effects, subscriber-interface restrictions, complete router advertisement disablement, and VRRP or HSRP considerations.

Understand these limitations before you suppress IPv6 ND router advertisements or enable unicast router advertisement messages.

Table 17: IPv6 ND RA limitations

Limitation	Details
Default RA state	By default, router advertisements are enabled on all IPv6 interfaces except management interfaces.
Unsolicited RA suppression	Suppressing unsolicited router advertisements stops periodic router advertisement messages. The interface still responds to router solicitations unless you also suppress solicited router advertisements.
Release 25.4.1 subscriber configuration	In Cisco IOS XR Release 25.4.1, the IPv6 ND unsolicited and solicited router advertisement commands are not available through dynamic templates or RADIUS attributes for subscriber interfaces.
All RA messages disabled	If you disable both unsolicited and solicited router advertisements, the interface does not send router advertisements.
VRRP or HSRP edge routers	For VRRP or HSRP, only the master router sends router advertisements. Backup routers ignore router solicitations and do not send router advertisements.
Subscriber BNG interfaces	The IPv6 ND unsolicited and solicited router advertisement configurations are not yet available for subscriber BNG interfaces configured through dynamic templates.

Configure IPv6 ND RA suppression and unicast RA messages

Configures IPv6 ND router advertisement behavior on an interface by suppressing unsolicited or solicited router advertisements, or by enabling solicited router advertisement responses to be sent as unicast messages.

Configure IPv6 ND router advertisement behavior to control router advertisement traffic on an interface.

If you suppress solicited router advertisement messages, enabling unicast solicited router advertisements has no effect because the interface does not send solicited router advertisements.

Procedure

1. Disable unsolicited router advertisement messages on the interface.

   Example

   ```
   Router# configure terminal
   Router(config)# interface GigabitEthernet0/0/0/1
   Router(config-if)# ipv6 nd unsolicited-ra disable
   Router(config-if)# exit
   ```

2. Disable solicited router advertisement messages on the interface.

   Example

   ```
   Router# configure terminal
   Router(config)# interface GigabitEthernet0/0/0/1
   Router(config-if)# ipv6 nd solicited-ra disable
   Router(config-if)# exit
   ```

3. Enable solicited router advertisement messages to be sent as unicast responses.

   Example

   ```
   Router# configure terminal
   Router(config)# interface GigabitEthernet0/0/0/1
   Router(config-if)# ipv6 nd solicited-ra unicast
   Router(config-if)# exit
   ```

IP address conflict resolution

Describes how IP Address Repository Manager (IPARM) detects address conflicts, applies conflict policies, supports connected-route tags, and uses Address Repository Manager data to select the active address.

IP address conflict resolution is an address-management function that

- detects conflicting IPv4 or IPv6 addresses across interfaces
- uses a configured IP Address Repository Manager (IPARM) conflict policy to select the winning address, and
- coordinates with the Address Repository Manager to keep address state consistent across protocols.

Conflict handling scope

Conflict resolution policies apply to IP address conflicts in the same address family. The policy determines which address remains active when duplicate or overlapping address information is detected.

Connected-route tags can be configured with IPv4 and IPv6 addresses on an interface. IP ARM propagates the tags to routing protocols so route policy language can identify or filter the connected routes.

IPARM conflict resolution policies

Describes the IPARM conflict policies that resolve address conflicts by preserving the running address, selecting the longest prefix, or selecting the highest IP address.

Use this reference to choose the IPARM conflict policy that determines which conflicting address remains active.

Policy summary

Table 18: IPARM conflict policies

Policy	Behavior
Static	Keeps the currently running address active and prevents a later conflicting address from taking precedence.
Longest prefix	Gives precedence to the conflicting address with the longest prefix length.
Highest IP	Gives precedence to the highest IP address among the conflicting addresses.

Static policy resolution

Static policy resolution gives precedence to the IP address that is already active on an interface. If another interface is configured later with a conflicting IP address, the later address is marked down and the original address remains active.

Use static policy resolution when the operational address that is already running must remain stable during later configuration changes.

Longest prefix address conflict resolution

Longest prefix address conflict resolution gives precedence to the address with the longest prefix length in the conflict set. The longest prefix is selected because it represents the most specific address scope among the conflicting entries.

When the policy selects the longest prefix, ARM marks lower-precedence conflicting address state down. Addresses that do not conflict with the selected longest-prefix address can remain active.

Highest IP address conflict resolution

Highest IP address conflict resolution gives precedence to the highest IP address in the conflict set. When duplicate or overlapping address state exists, ARM compares the conflicting addresses and keeps the highest address active.

Use highest IP address conflict resolution when address precedence must be determined by numeric address value instead of configuration order or prefix length.

Related configuration

Use the **ipv4 conflict-policy** command to configure an IPv4 conflict policy. Use the corresponding **ipv6 conflict-policy** command to configure conflict handling for IPv6.

Use the **show arm ipv4 conflicts** command or the **show arm ipv6 conflicts** command to display conflicting address information.

Configure IPARM conflict resolution policies

Configures IPARM conflict resolution policies so IPv4 or IPv6 address conflicts use static, longest-prefix, or highest-IP precedence rules during interface address activation.

Configure an IPARM conflict resolution policy to control which address remains active when an address conflict occurs.

You can configure static, longest-prefix, or highest-IP conflict resolution. Use the corresponding **ipv6 conflict-policy** command when you configure IPv6 conflict handling.

Procedure

1. Configure the static conflict policy.

   Example

```
Router# configure
Router(config)# ipv4 conflict-policy static
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)# exit
Router(config)# commit
```

2. Verify the IPv4 conflict information.

   Example

```
Router# show arm ipv4 conflicts

Address         Interface              State
192.0.2.1       HundredGigE0/0/0/1     UP
192.0.2.1       HundredGigE0/0/0/2     DOWN
```

3. Configure the longest-prefix conflict policy.

   Example

```
Router# configure
Router(config)# ipv4 conflict-policy longest-prefix
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)# exit
Router(config)# commit
```

4. Configure the highest-IP conflict policy.

   Example

```
Router# configure
Router(config)# ipv4 conflict-policy highest-ip
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)# exit
Router(config)# commit
```

Address Repository Manager

Describes how IPARM enforces global IP address uniqueness, disseminates address information, maintains conflict sets, and selects the highest-precedence address deterministically.

IPv4 and IPv6 Address Repository Manager is an IP address repository function that

- enforces uniqueness for global IP addresses configured in the system
- disseminates global IP address information to processes on RPs and LCs through IP address consumer APIs, and
- includes unnumbered interface information when it provides address data to consumers.

Address conflict resolution

Address conflict resolution has two parts: the conflict database and the conflict set definition. IPARM uses both parts to maintain a global view of conflicting IP addresses.

Conflict database

IPARM maintains a global conflict database. IP addresses that conflict with each other are maintained in lists called conflict sets. These conflict sets make up the global conflict database.

A set of IP addresses is part of a conflict set if at least one prefix in the set conflicts with every other IP address in the same set.

Table 19: Conflict set example

Address	Prefix
Address 1	`10.1.1.1/16`
Address 2	`10.2.1.1/16`
Address 3	`10.3.1.1/16`
Address 4	`10.4.1.1/8`

In this example, the four addresses are part of one conflict set because at least one prefix in the set conflicts with each of the other addresses.

Deterministic conflict policy

When a conflicting IP address is added to a conflict set, IPARM runs an algorithm through the set to determine the highest-precedence address.

The conflict policy algorithm is deterministic. You can determine which addresses on an interface are enabled or disabled. The enabled address is declared as the highest-precedence IP address for that conflict set.

How address conflict resolution works

Describes how ARM detects duplicate address state, records the conflict, applies the configured policy, and notifies clients about the winning address.

Address conflict resolution starts when ARM receives address state from address producers and detects a duplicate or overlapping address condition.

Summary

Address conflict resolution uses these elements:

- Address producers: Provide address state to ARM.
- ARM conflict database: Stores the conflict state and candidate addresses.
- Conflict policy: Determines which candidate address has precedence.

- Address clients: Receive the resolved address state from ARM.

Workflow

These stages describe how address conflict resolution works.

1. An address producer sends address state to ARM.
2. ARM compares the new address state with existing address state and detects whether a conflict exists.
3. When a conflict exists, ARM records the conflicting entries and applies the configured conflict resolution policy.
4. The policy selects the address with precedence and marks the other conflicting address state down.
5. ARM notifies address clients about the resolved address state so dependent protocols use the active address.

Result

The router keeps one active address state for the conflict and prevents conflicting address state from being used by clients.

Route-tag support for connected routes

Describes how route tags are added to connected IPv4 and IPv6 routes and propagated through IPARM to routing protocols for route policy control.

Route-tag support for connected routes is an address-management function that

- adds a route tag as an attribute of an IPv4 or IPv6 interface address
- passes the route tag from IP master address processing to IP ARM, and
- makes the tag available to routing protocols for route policy language processing.

Route policy use

Routing protocols can use connected-route tags to identify, match, or control distribution of connected routes through route policy language.

Configure the tag with the interface address by using the **route-tag** keyword. The route tag becomes part of the connected route information that IP ARM provides to routing protocols.

Configure route tags for connected routes

Configures route tags on connected IPv4 routes so IPARM can propagate tag information to routing protocols for route policy processing.

Configure a route tag on a connected route so routing protocols can use the tag in route policy language.

Route tags configured with IPv4 and IPv6 addresses on an interface are propagated from the IP master address component to IPARM, and then to routing protocols.

Procedure

1. Configure an IPv4 address with a route tag on the interface.

Example

```
Router# configure
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# ipv4 address 198.51.100.1 255.255.255.0 route-tag 50
```

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

2. Verify the running configuration for the route-tagged address.

Example

```
Router# show running-config interface HundredGigE 0/0/0/1

interface HundredGigE0/0/0/1
 ipv4 address 198.51.100.1 255.255.255.0 route-tag 50
!
```

3. Verify the parameters of the route.

Example

```
Router#show route 198.51.100.1
Routing entry for 198.51.100.1/24
  Known via "local", distance 0, metric 0 (connected)
  Tag 50
Routing Descriptor Blocks
    directly connected, via HundredGigE0/0/0/24
      Route metric is 0
  No advertising protos.
```

VRF fallback and redirect

Describes how VRF fallback and redirect forward unmatched traffic through another VRF and use ABF rules to redirect IPv4 or IPv6 packets.

VRF fallback and redirect is a forwarding function that

- uses a fallback VRF when a route lookup in the primary VRF does not find a match
- uses ACL-based forwarding to redirect selected IPv4 and IPv6 packets, and
- supports traffic steering across VRF boundaries when the configured restrictions are met.

Table 20: Feature History Table

Feature Name	Release Information	Feature Description
VRF Fallback and Redirect	Release 7.3.16	The VRF fallback functionality of this feature allows you to configure a fallback VRF route, which is a better alternative than a default route from the global routing table that requires an explicit next hop. If the destination prefix of a data packet does not have a match in the VRF table, the fallback VRF table is used. The VRF redirect functionality of this feature allows the forwarding of IPv4 and IPv6 packets into a configured next hop VRF.

Fallback behavior

When a lookup in the primary VRF does not produce a route, the router can perform another lookup in the configured fallback VRF. The resolved route can then identify the next-hop VRF used for forwarding.

Redirect behavior

ACL-based forwarding can redirect IPv4 or IPv6 packets to a specified next-hop address in another VRF. The redirect behavior is configured in IPv4 or IPv6 access lists and applied to the ingress interface.

Restrictions of VRF fallback and redirect limits

Provides the VRF fallback and redirect limits for PPS impact, unsupported adjacencies, scale, and subscriber interfaces.

These restrictions apply when you configure fallback VRF or ACL-based forwarding redirect behavior:

- Configure only one fallback VRF route for each address family in each primary VRF.
- Do not use ping, traceroute, or other slow-path applications with fallback VRF. LPTS receive trap support is not available for fallback VRF.
- Use a maximum of 512 VRFs and one global table on the router.
- If a static default route is configured for a VRF, the static default route takes precedence over the fallback VRF. The global routing table is used for route lookup, and the default route is always directed to the configured next hop.
- If route lookup for a packet fails in the primary VRF, the router retries the lookup in the fallback VRF.
- If both ABF VRF redirect and VRF fallback are configured for a packet, forwarding performance can have up to 25 percent PPS impact because both functions are processed at the full throughput rate of the system.
- If a route for a packet is found in the fallback VRF, only Glean IPv4 and Glean IPv6 adjacency packets are punted successfully.
- Avoid looped configurations. If a route for a packet is not found in both the primary VRF and fallback VRF, the packet loops in the recycle path and is eventually dropped in the recycle egress queue. Because the recycle queue has the highest priority, high-rate looped traffic can cause other valid recycled packets to be dropped.

Configure VRF fallback and redirect

Configures fallback VRF lookup and ACL-based forwarding redirect rules for IPv4 and IPv6 traffic that must be steered to another VRF.

Configure VRF fallback and redirect to forward selected traffic through another VRF when the configured lookup and ACL conditions are met.

Configure fallback VRF under the source VRF. Configure ACL-based forwarding redirect rules in IPv4 or IPv6 access lists and apply the access group to the ingress interface.

Procedure

1. Configure a fallback VRF.

Example

```
Router# configure
Router(config)# vrf vrf100
Router(config-vrf)# fallback-vrf vrf200
Router(config-vrf)# commit
```

2. Verify that the lookup resolves through the fallback VRF.

   Example

```
Router# show cef vrf vrf100 192.0.2.1 detail

192.0.2.1/32, version 4, internal 0x5000001 0x0 (ptr 0xa147eab8) [1], 0x0
(0xa13e1b20), 0x0
 Updated Jun 25 10:05:53.930
 fallback VRF: vrf200
 Prefix Len 32, traffic index 0, precedence n/a, priority 4
 next hop VRF - 'vrf200', table - 0xe0000004
```

3. Configure an IPv6 access list that redirects matching packets to a next-hop VRF.

   Example

```
Router# configure
Router(config)# ipv6 access-list tms-bypass-v6
Router(config-ipv6-acl)# 10 permit ipv6 any any nexthop1 vrf tms-bypass-l3vrf
 2001:DB8:1:1::1
Router(config-ipv6-acl)# exit
Router(config)# commit
```

4. Configure an IPv4 access list that redirects matching packets to a next-hop VRF.

   Example

```
Router# configure
Router(config)# ipv4 access-list tms-bypass-v4
Router(config-ipv4-acl)# 10 permit ipv4 any any nexthop1 vrf tms-bypass-l3vrf
 192.0.2.1
Router(config-ipv4-acl)# exit
Router(config)# commit
```

5. Apply the IPv4 and IPv6 access groups to the ingress interface.

   Example

```
Router# configure
Router(config)# interface Bundle-Ether 100.100
Router(config-subif)# ipv4 access-group tms-bypass-v4 ingress
Router(config-subif)# ipv6 access-group tms-bypass-v6 ingress
Router(config-subif)# commit
```

6. Verify the route that is resolved through the selected next-hop VRF.

   Example

```
Router# show route vrf tms-bypass-l3vrf

B    198.51.100.0/24 [20/2219] via 203.0.113.1 (nexthop in vrf default)
```

Local station MAC addresses for routers

Describes how a router-wide local station MAC address supports static ARP and dynamic ARP packets received across router interfaces on supported routers.

A local station MAC address for a router is a configured MAC address that

- represents the router for local station processing
- can be added for static ARP and dynamic ARP packets received on all router interfaces, and
- is configured at the router level using the **hw-module local-station-mac** command..

Table 21: Feature History Table

Feature Name	Release Information	Feature Description
Local Station MAC Address for Router	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
Local Station MAC Address for Router	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Local Station MAC Address for Router	Release 7.9.1	Local Station MAC Address is the MAC address for the router that applies to all interfaces, including VRFs. You can configure Local Station MAC Address for a router using the **hw-module local-station-mac** command. This feature allows the router and its interfaces to have one network-wide identity and helps in identifying the neighboring devices in the network while using static ARP. The feature introduces these changes: • CLI: New hw-module local-station-mac and show hw-module local-station-mac commands • YANG Data Model: `Cisco-IOS-XR-um-8000-local-mac-cfg.yang` Cisco native model (see GitHub, YANG Data Models Navigator)

Reduced static ARP entries with system-wide MAC addresses

When a router receives Layer 3 traffic, the destination MAC address is the MAC address of the router interface that receives the packet.

For dynamic ARP, the router learns the interface MAC address through ARP response exchanges.

For static ARP, the router needs a system-wide MAC address to identify the router. This system-wide MAC address must be available in the static ARP table.

In earlier implementations, the router maintained the MAC address of each interface in the static ARP table. This approach used more router resources because the router had to maintain a pool of static ARP entries for all interfaces. The current implementation uses one static ARP entry for the system.

Restrictions for local station MAC addresses

Provides the platform restriction for local station MAC addresses and identifies Cisco Silicon One Q100 ASIC-based routers as unsupported for this feature.

Local station MAC address support is not available on Cisco Silicon One Q100 ASIC-based routers.

For a list of Cisco 8000 Series hardware using Cisco Silicon One Q100 ASIC, refer Cisco 8000 Series Routers Data Sheet.

Configure local station MAC addresses for routers

Configures a router-wide local station MAC address and verifies that the configured address is stored in the local station MAC table.

Configure a local station MAC address for static ARP and dynamic ARP packets received on all router interfaces.

Procedure

1. Configure the local station MAC address.

   Example

```
Router# configure
Router(config)# hw-module local-station-mac B03F.C98C.B948
Router(config)# commit
```

2. Verify the local station MAC address.

   Example

```
Router# show hw-module local-station-mac
-------------------------------------------------------------
Knob                          Status          Applied   Action
-------------------------------------------------------------
Local-Station-MAC             Configured      Yes       None
```

4 Address Resolution

Topics:

Summarizes ARP address resolution, cache entries, proxy behavior, cache management controls, duplicate packet policing, and direct-attached gateway redundancy configuration guidance.

Use this chapter to understand Address Resolution Protocol (ARP) behavior and configure ARP cache, proxy, traffic-control, and gateway-redundancy functions.

The chapter organizes ARP information into these content areas:

Table 22: ARP content areas

Content area	Included information
ARP fundamentals	ARP restrictions, address resolution, and the address resolution process.
ARP cache entries	Dynamic and static ARP cache entries and their use on router interfaces.
Proxy ARP	Proxy ARP and local proxy ARP behavior.
ARP cache management	Cache purge delays, entry timeouts, per-interface entry limits, and duplicate-packet policers.
Direct-attached gateway redundancy	Direct-attached gateway redundancy (DAGR) behavior and operational restrictions.

The topics progress from ARP behavior to cache configuration, proxy functions, cache controls, duplicate-packet policing, and DAGR. Restrictions appear before the related configuration tasks.

ARP fundamentals

Explains how ARP maps IP addresses to MAC addresses, how address resolution operates on local and routed networks, and which restrictions apply to ARP configuration.

Address Resolution Protocol (ARP) is an address-mapping protocol that

- takes an IP address as input and determines its associated media or MAC address
- stores the IP-to-MAC address association in an ARP cache for rapid retrieval, and
- enables an IP datagram to be encapsulated in a link-layer frame and sent over the network.

Cisco IOS XR software supports ARP, proxy ARP, and local proxy ARP. ARP is defined in RFC 826, and proxy ARP is defined in RFC 1027.

The system typically resolves addresses dynamically. You can also configure a static ARP entry when a permanent mapping is required.

ARP restrictions

Provides restrictions that must be considered before you configure ARP features in Cisco IOS XR software during network design and feature deployment.

The software does not support these functions:

- Reverse Address Resolution Protocol (RARP)
- ARP throttling, which is the rate limiting of ARP packets in the Forwarding Information Base (FIB)

Address resolution

Explains how address resolution maps an IP address to the 48-bit MAC address that identifies a device on its local Ethernet segment.

Address resolution is a process that

- determines a device's 48-bit media access control (MAC) address from its IP address
- supports communication with the device on a local Ethernet segment, and
- uses local addressing information that the data link layer processes.

Local and network addresses

An IP network device can have a local address and a network address. The local address uniquely identifies the device on its local segment or LAN. The network address identifies the network to which the device belongs.

The local address is also called a data link address. It appears in the data link layer, or Layer 2, section of the packet header. The data link layer is part of the Open Systems Interconnection (OSI) model.

Data-link devices, such as bridges and device interfaces, read the local address. The MAC sublayer processes the address, which is commonly called a MAC address.

Address resolution on Ethernet

To communicate with an Ethernet device, Cisco IOS XR software determines the device's 48-bit MAC address. Address resolution maps the device's IP address to this local data link address.

How address resolution works

Describes how devices resolve IP addresses to MAC addresses when communicating on one LAN or across LANs through a router that has proxy ARP enabled.

A device can have a local data-link address that identifies it on its LAN and a network address that identifies its network. On Ethernet, Cisco IOS XR software resolves an IP address to a 48-bit MAC address before sending traffic.

Communication between different LANs uses the router process only when proxy ARP is enabled.

Summary

The address-resolution process involves these components:

- The source device broadcasts an ARP request and stores the received mapping.
- The destination device replies when the source and destination are on the same LAN.
- A router replies on behalf of a destination on another LAN when proxy ARP is enabled.

The resulting ARP cache entry lets the source send later frames without repeating the initial request.

Workflow

These stages describe address resolution on a single LAN and across LANs interconnected by a router.

1. Address resolution on a single LAN

   The source device broadcasts an ARP request on its LAN to learn the destination device's MAC address.

   Every device on the LAN receives and processes the request.

2. When the destination is on the same LAN, only that destination sends an ARP reply that contains its MAC address.

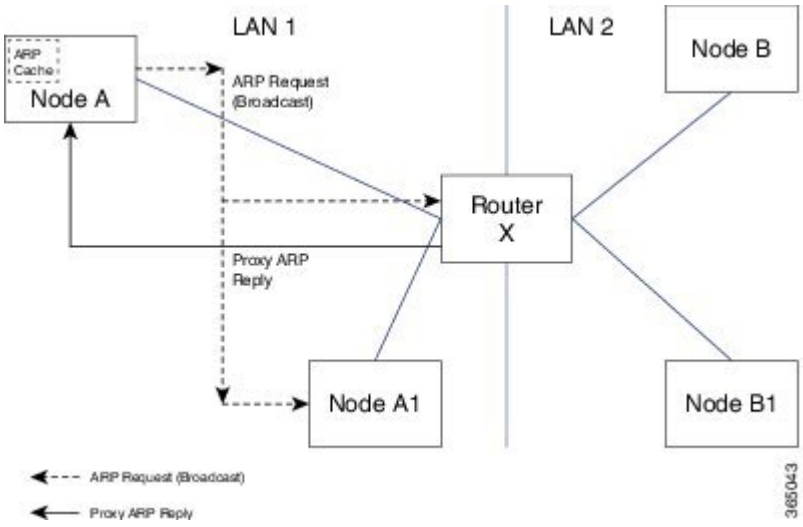
   The source saves the mapping between the destination's IP address and MAC address in its ARP cache.

   The source programs the learned adjacency in hardware, which forwards subsequent packets to the destination.

3. Address resolution when interconnected by a router

   In this topology, the source and destination are on different LANs that are interconnected by a router. The router must have proxy ARP enabled for this process.

   Figure 9: Address resolution when interconnected by a router



   The source broadcasts an ARP request for the remote destination. Devices on the source LAN, including the router, receive and process the request.

The router checks its routing table to determine whether it can reach the destination and replies with its own MAC address.

The source stores the router's MAC address in the ARP cache entry for the remote destination.

4. The source sends subsequent frames for the remote destination to the router's MAC address without broadcasting another ARP request.

   The router receives the traffic, resolves the destination on the other LAN as needed, and forwards the traffic to the destination.

Result

The source obtains a usable data-link destination and can forward traffic to a local device or through a router to a remote device.

ARP cache entries

Explains how ARP cache entries retain IP-to-hardware-address mappings and when to define a permanent static entry for consistent address translation.

An ARP cache entry is an address correspondence that

- maps a network address, such as an IP address, to an Ethernet hardware address
- lets the system retrieve an existing mapping without repeating address resolution, and
- remains in the cache dynamically for a predetermined time or permanently when configured as a static entry.

Static ARP cache entries

Describes when to configure a static ARP cache entry and how Cisco IOS XR software uses the entry for permanent address translation.

Use this reference to understand the purpose and behavior of static ARP cache entries.

Most hosts support dynamic address resolution, so static ARP entries are generally unnecessary.

When a permanent mapping is required, you can configure a global static entry. The entry persists until you explicitly remove it.

Cisco IOS XR software uses the static entry to translate a 32-bit IP address into a 48-bit hardware address.

Define a static ARP cache entry

Configures a permanent association between an IP address and a MAC address, and verifies that the resulting ARP cache entry has a static state.

Use a static ARP cache entry when the system must retain a specific IP-to-MAC address mapping until you explicitly remove it.

The example maps IP address 203.0.113.2 to MAC address 0010.9400.000c.

Before you begin

Identify the IP address and MAC address for the permanent mapping.

Follow these steps to define a static ARP cache entry.

Procedure

1. Define the static mapping between the IP address and the MAC address in global configuration mode.

   Example

   ```
   Router# configure
   Router(config)# arp 203.0.113.2 0010.9400.000c ARPA
   Router(config)# commit
   ```

2. Display the running ARP configuration.

   Example

   ```
   Router# show running-config arp
   arp vrf default 203.0.113.2 0010.9400.000c ARPA
   ```

3. Verify that the ARP entry state is `Static`.

   Example

   ```
   Router# show arp location 0/0/CPU0
   Address         Age  Hardware Addr    State      Type  Interface
   203.0.113.1     -    ea28.5f0b.8024  Interface  ARPA  HundredGigE0/0/0/9
   203.0.113.2     -    0010.9400.000c  Static     ARPA  HundredGigE0/0/0/9
   ```

Proxy ARP and local proxy ARP

Explains when the router answers ARP requests on behalf of another device and how to enable proxy ARP or local proxy ARP for different network relationships.

Proxy ARP functions are address-resolution functions that

- let the router answer eligible ARP requests with its own local data-link address
- help hosts reach destinations without requiring the hosts to understand routing, and
- support either destinations on other physical networks or destinations on the same Layer 3 network, depending on the configured function.

Proxy ARP behavior

When proxy ARP is disabled, the router responds only when the target is the receiving interface address or has a statically configured ARP alias.

When proxy ARP is enabled, the router also responds when the target is on another physical network, the router has a route to the target, and every route to the target uses an interface other than the receiving interface.

Local proxy ARP behavior

Local proxy ARP applies when the target, source, and receiving interface addresses are on the same Layer 3 network and the next hop to the target uses the receiving interface.

Local proxy ARP commonly resolves addresses for devices on the same Layer 3 network that are separated at Layer 2, such as devices in private VLANs. It supports the interface types supported by ARP, including unnumbered interfaces.

Enable proxy ARP

Enables proxy ARP on an interface so that the router can answer eligible requests for hosts on other networks or subnets.

Enable proxy ARP when hosts without routing knowledge must reach devices through the router.

Proxy ARP is disabled by default. The example enables it on interface HundredGigE0/0/0/0.

Before you begin

Identify the interface on which the router must answer eligible proxy ARP requests.

Follow these steps to enable proxy ARP.

Procedure

1. Enter interface configuration mode.

   Example

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

2. Enable proxy ARP and commit the configuration.

   Example

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

3. Display the running interface configuration.

   Example

   ```
   Router# show running-config interface HundredGigE 0/0/0/0
   interface HundredGigE0/0/0/0
    mtu 4000
    ipv4 address 192.0.2.1 255.255.255.0
    proxy-arp
   !
   ```

4. Verify the operational proxy ARP state.

   Example

   ```
   Router# show arp idb HundredGigE 0/0/0/0 location 0/0/CPU0
   interface HundredGigE0/0/0/0 (0x08000038):
   IPv4 address 192.0.2.1, Vrf ID 0x60000000
   VRF Name default
   Dynamic learning: Enable
   Dynamic entry timeout: 14400 secs
   Purge delay: off
   ```

```
Proxy arp is configured, is enabled
Local Proxy arp not configured
```

Enable local proxy ARP

Enables local proxy ARP so that the router can resolve addresses for devices on the same Layer 3 network that are separated at Layer 2.

Enable local proxy ARP when devices share a Layer 3 network but cannot resolve each other directly at Layer 2.

A typical use case involves private VLANs that separate devices at Layer 2. The example enables local proxy ARP on interface HundredGigE0/0/0/0.

Before you begin

Confirm that the source, target, and receiving interface addresses are on the same Layer 3 network and that the next hop to the target uses the receiving interface.

Follow these steps to enable local proxy ARP.

Procedure

1. Enter interface configuration mode.

   Example

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

2. Enable local proxy ARP and commit the configuration.

   Example

   ```
   Router(config-if)# local-proxy-arp
   Router(config-if)# commit
   ```

3. Display the running interface configuration.

   Example

   ```
   Router# show running-config interface HundredGigE 0/0/0/0
   interface HundredGigE0/0/0/0
    ipv4 address 192.0.2.1 255.255.255.0
    local-proxy-arp
   !
   ```

4. Verify the operational local proxy ARP state.

   Example

   ```
   Router# show arp idb HundredGigE0/0/0/0 location 0/0/CPU0
   HundredGigE0/0/0/0 (0x08000038):
   IPv4 address 192.0.2.1, Vrf ID 0x60000000
   VRF Name default
   ```

```
Dynamic learning: Enable
Dynamic entry timeout: 14400 secs
Purge delay: off
Proxy arp not configured, not enabled
Local Proxy arp is configured
```

ARP cache management

Explains how four ARP controls preserve entries, expire stale mappings, limit cache growth, and police duplicates and provides procedures for configuring and verifying each control.

Address Resolution Protocol (ARP) cache management uses controls that

- preserve dynamic entries during short interface flaps and expire stale entries
- limit dynamic entries on each interface to protect cache resources, and
- police duplicate requests from the same IP address or MAC address.

ARP purge delays

Explains how ARP purge delay preserves dynamic ARP entries during brief interface outages, restores them if the interface recovers before the timer expires, and supports forwarding recovery after interface flaps and ECMP path changes.

An ARP purge delay is an interface recovery control that

- caches existing dynamic ARP entries when an interface goes down
- restores cached entries when the interface returns before the purge timer expires, and
- deletes cached entries when the purge timer expires before the interface returns.

Entry refresh after an interface flap

The system temporarily reduces the normal entry timeout after the interface state changes.

The reduced timeout starts new ARP resolution after the interface state stabilizes.

Use with equal-cost multipath

Equal-cost multipath (ECMP) sends traffic across paths that have equal cost.

A short interface flap can move traffic to another path and then return it to the restored path.

The purge delay preserves entries until ARP resolution and adjacency installation resume forwarding.

Configure the ARP purge delay

Configures an ARP purge delay on an interface to preserve dynamic ARP entries during a short interface flap and verifies that the configured delay is applied.

Configure a purge delay to preserve dynamic ARP entries during a short interface flap.

This example configures a 100-second purge delay on interface HundredGigE 0/0/0/34.

Before you begin

Identify the interface and select a purge-delay value.

Follow these steps to configure the ARP purge delay.

Procedure

1. Set the purge delay to 100 seconds in interface configuration mode.

   Example

```
Router# configure
Router(config)# interface HundredGigE 0/0/0/34
Router(config-if)# arp purge-delay 100
Router(config-if)# commit
```

2. Display the running interface configuration.

   Example

```
Router# show running-config interface HundredGigE 0/0/0/34
interface HundredGigE0/0/0/34
 arp purge-delay 100
 shutdown
!
```

3. Verify that the interface data reports a 100-second purge delay.

   Example

```
Router# show arp idb HundredGigE 0/0/0/34 location 0/RP0/CPU0
HundredGigE0/0/0/34:
  IDB Client: default
  IPv4 address 192.0.2.1, Vrf ID 0x00000000
  VRF Name unknown
  Dynamic learning: Enable
  Dynamic entry timeout: 14400 secs
  Drop adjacency timeout: Disable
  Purge delay: 100 seconds
  Proxy arp not configured, not enabled
  Local Proxy arp not configured
  Packet IO layer is NetIO
  Srg Role: DEFAULT
  Total entries: 1
```

The interface retains dynamic ARP entries for up to 100 seconds after it goes down.

ARP timeouts

Explains how ARP timeout controls the aging of dynamic ARP entries by removing stale learned entries at the default interval, while preserving local-interface and statically configured entries.

An ARP timeout is an entry-aging control that

- applies to dynamic entries that the system learns from valid ARP replies
- removes stale dynamic entries every four hours by default, and
- does not expire local-interface entries or statically configured entries.

Configure the ARP timeout

Configures the timeout for dynamic ARP entries on an interface and verifies that cached entries expire after the specified interval.

Configure how long dynamic ARP entries remain in the cache before they expire.

This example configures a 100-second timeout on interface HundredGigE 0/0/0/35.

Before you begin

Identify the interface and select a timeout value.

Follow these steps to configure the ARP timeout.

Procedure

1. Set the dynamic entry timeout to 100 seconds in interface configuration mode.

   Example

```
Router(config)# interface HundredGigE 0/0/0/35
Router(config-if)# arp timeout 100
Router(config-if)# commit
```

2. Display the running interface configuration.

   Example

```
Router# show running-config interface HundredGigE 0/0/0/35
interface HundredGigE0/0/0/35
 arp timeout 100
 shutdown
!
```

3. Verify that the interface data reports a 100-second dynamic entry timeout.

   Example

```
Router# show arp idb HundredGigE 0/0/0/35 location 0/RP0/CPU0
HundredGigE0/0/0/35:
  IDB Client: default
  IPv4 address 192.0.2.1, Vrf ID 0x00000000
  VRF Name unknown
  Dynamic learning: Enable
  Dynamic entry timeout: 100 secs
  Drop adjacency timeout: Disable
  Purge delay: off
  Proxy arp not configured, not enabled
  Local Proxy arp not configured
  Packet IO layer is NetIO
  Srg Role: DEFAULT
  Total entries: 1
```

Dynamic ARP entries on the interface expire after 100 seconds.

ARP cache entry limits

Explains how ARP cache entry limits protect interface ARP caches from overflow and memory overuse by restricting dynamic entry learning, and describes supported values, limit behavior, exclusions, and monitoring through syslog and interface statistics.

An Address Resolution Protocol (ARP) cache entry limit is an overflow-protection control that

- sets the maximum number of dynamic ARP entries that an interface can learn
- prevents cache entries from consuming excessive router memory, and
- drops new requests after the cache reaches the configured limit.

Table 23: Feature History Table

Feature Name	Release Information	Feature Description
Limit Address Resolution Protocol (ARP) Cache Entries per Interface	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
Limit Address Resolution Protocol (ARP) Cache Entries per Interface	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
Limit Address Resolution Protocol (ARP) Cache Entries per Interface	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Limit Address Resolution Protocol (ARP) Cache Entries per Interface	Release 7.5.4	In this feature, you can configure the maximum limit for the number of entries of dynamic mapping between IP addresses and media addresses by ARP per interface. Limiting the number of entries provides overflow protections in ARP cache and protects the routers from DOS attacks by preventing memory overuse by cache entries. This feature introduces the **arp cache-limit** command.

Supported values and resources

- You can configure a value from 0 to 127,999 on each interface.
- The maximum cache capacity is 127,999 entries.
- Available ARP cache resources depend on router hardware. Select a limit that fits the available resources.

Cache-limit behavior

The cache limit has these effects:

- The router drops new ARP requests when the cache reaches or exceeds the configured limit.
- The router stops learning from ARP packets after the cache exceeds the configured limit.
- The limit does not apply to static ARP entries.
- The router does not enforce the limit on entries that an ARP client triggers.
- The router retains dynamic entries that it learned before a lower limit was configured.
- The router generates a syslog message when the cache reaches the limit.
- The router generates another syslog message for every 1,000 entries beyond the limit.
- The **show arp idb** command displays ARP entry statistics.

Limit ARP cache entries per interface

Configures a per-interface ARP cache entry limit to restrict dynamic ARP learning and verifies the configured limit and entry counters.

Limit dynamic Address Resolution Protocol (ARP) entries to protect the interface cache from overflow and memory overuse.

This example sets the cache limit to 3900 on interface HundredGigE 0/0/0/0.

Before you begin

Select a value from 0 to 127999 that fits the available router resources.

Follow these steps to limit ARP cache entries per interface.

Procedure

1. Set the ARP cache limit to 3900 entries in interface configuration mode.

   Example

```
Router# configure
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# arp cache-limit 3900
Router(config-if)# commit
```

2. Display the running interface configuration.

   Example

```
Router# show running-config interface HundredGigE 0/0/0/0
interface HundredGigE0/0/0/0
 arp cache-limit 3900
!
!
```

3. Verify the cache limit and entry counters.

   Example

```
Router# show arp idb HundredGigE 0/0/0/0 location RP0
HundredGigE0/0/0/0:
  IDB Client: default
  IPv4 address 192.0.2.1, Vrf ID 0x60000000
  VRF Name default
  Dynamic learning: Enable
  Dynamic entry timeout: 14400 secs
  Drop adjacency timeout: Disable
  Purge delay: off
  Cache limit: 3900
  Incomplete glean count: 0
  Complete glean count: 0
  Complete protocol count: 0
  Dropped glean count: 0
  Dropped protocol count: 0
```

ARP duplicate policers

Describes how the ARP duplicate policer drops repeated ARP requests from the same IP or MAC address within a configured interval.

An ARP duplicate policer is a request-control mechanism that

- checks requests from the same sender IP address within the configured interval
- drops a new request when a response already exists for that IP address, and
- checks the MAC-address policer when no response exists for that IP address.

Duplicate request handling

The router drops ARP requests from the same IP address or MAC address during the configured interval.

Police duplicate ARP packets

Configures the interval that the router uses to identify duplicate ARP requests.

Set an interval for policing duplicate Address Resolution Protocol (ARP) packets from the same IP address or MAC address.

This example sets the global ARP policing interval to 34.

Procedure

Set the ARP policing interval to 34 in global configuration mode.

Example

```
Router# configure
Router(config)# arp police-interval 34
Router(config)# commit
```

Direct-attached gateway redundancy

Explains how direct-attached gateway redundancy provides Layer 3 failover by using gratuitous ARP signals and RIB route advertisement for OSPF distribution, enabling subsecond redundancy with directly connected devices.

Direct-attached gateway redundancy (DAGR) is a Layer 3 redundancy feature that

- uses gratuitous Address Resolution Protocol (ARP) messages as failover signals
- advertises routes in the Routing Information Base (RIB) for Open Shortest Path First (OSPF) distribution, and
- supports subsecond failover without MAC learning, MAC flooding, or an intervening Ethernet switch.

Use with connected redundant devices

Some connected devices use proprietary Layer 2 mechanisms instead of routing protocols.

Examples include redundant base station controllers and multimedia gateways with aggressive failover requirements.

These mechanisms can resemble Virtual Router Redundancy Protocol (VRRP).

Interoperability qualification

One-to-one Layer 2 redundancy mechanisms can differ because they are proprietary.

Qualify interoperability before you deploy DAGR with IP mobile equipment.

Restrictions of direct-attached gateway redundancy

Describes the unsupported capabilities and deployment restrictions for direct-attached gateway redundancy, including lack of support for IPv6, Ethernet bundles, non-Ethernet interfaces, and hitless recovery behaviors.

Plan the direct-attached gateway redundancy (DAGR) deployment without these unsupported capabilities:

- DAGR does not support IPv6.

- DAGR does not support Ethernet bundles.
- DAGR does not support non-Ethernet interfaces.
- DAGR does not support hitless ARP process restart.
- DAGR does not support hitless route switch processor failover.

Configure DAGR

Configures a DAGR peer with route preferences, query and standby timers, and a priority timeout and verifies the operational state of DAGR groups.

Configure direct-attached gateway redundancy (DAGR) for a peer that uses gratuitous Address Resolution Protocol (ARP) as a failover signal.

This example configures peer 192.0.2.1 on interface HundredGigE 0/0/0/1.

Before you begin

Confirm that the deployment complies with all DAGR restrictions.

Identify the interface, peer IPv4 address, route preferences, and timer values.

Follow these steps to configure DAGR.

Procedure

1. Enter DAGR configuration mode.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/0/0/1
   Router(config-if)# arp dagr
   ```

2. Configure peer IPv4 address 192.0.2.1.

   Example

   ```
   Router(config-if-dagr)# peer ipv4 192.0.2.1
   ```

3. Set the normal and priority route distances.

   Example

   ```
   Router(config-if-dagr-peer)# route distance normal 140 priority 3
   ```

4. Set the normal and priority route metrics.

   Example

   ```
   Router(config-if-dagr-peer)# route metric normal 84 priority 80
   ```

5. Set the query timer to 2 and the standby timer to 19.

Example

```
Router(config-if-dagr-peer)# timers query 2 standby 19
```

6. Set the priority timeout to 25.

Example

```
Router(config-if-dagr-peer)# priority-timeout 25
Router(config-if-dagr-peer)# commit
```

7. Display the running DAGR configuration.

Example

```
Router# show running-config interface HundredGigE 0/0/0/1
interface HundredGigE 0/0/0/1
 arp dagr
  peer ipv4 192.0.2.1
   priority-timeout 25
   route distance normal 140 priority 3
   route metric normal 84 priority 80
   timers query 2 standby 19
!
```

8. Verify the operational state of the DAGR groups.

Example

```
Router# show arp dagr
Interface              Virtual IP   State   Query-pd  Dist  Metr
HundredGigE0/0/0/1     192.0.2.1    Active  None      150   100
HundredGigE0/0/0/1     192.0.2.45   Query   1         None  None
```

5 Dynamic Host Configuration Protocol

Topics:

This chapter groups the concepts, tasks, and reference material required to implement Dynamic Host Configuration Protocol (DHCP) on the Cisco 8000 Series Router.

Introduction to DHCP relay

This topic describes the DHCP relay agent on the Cisco 8000 Series Router, which forwards DHCP packets between clients and servers that do not reside on a shared physical subnet.

A DHCP relay agent is a host that

- forwards DHCP packets between clients and servers that do not reside on a shared physical subnet, and
- differs from a normal IP router because it processes and forwards DHCP messages instead of transparently switching IP datagrams between networks.

Table 24: Feature History Table

Feature Name	Release Information	Feature Description
Extending DHCP Relay agent support	Release 26.3.1	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.
Extending DHCP Relay agent support	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.
Extending DHCP Relay agent support	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: • 8712-MOD-M • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Extending DHCP Relay agent support	Release 25.2.1	Introduced in this release on: Fixed Systems ((8010 [ASIC: A100], 8200 [ASIC: P100]), 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) * This feature is supported on: • 8011-4G24Y4H-I • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH

Feature Name	Release Information	Feature Description
Extending DHCP Relay agent support	Release 25.1.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) This release introduces support for DHCP Relay Agent and DHCPv6 Relay Over BVI for IANA Address Allocation to Cisco Silicon One P100 ASIC-based systems. The feature, enabled by default, efficiently allocates IP addresses from a centralized DHCP server to clients across multiple subnets. It ensures that devices can receive IP addresses even when they are not on the same local network as the DHCP server. *The feature is supported on: • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

How DHCP relay works

This topic describes how a DHCP relay agent forwards a DHCP client broadcast to a DHCP server on another network segment on the Cisco 8000 Series Router.

DHCP clients use User Datagram Protocol (UDP) broadcasts to send DHCPDISCOVER messages when they lack information about the network to which they belong.

If a client is on a network segment that does not include a server, the network segment needs a relay agent. Presence of the relay agent ensures that DHCP packets reach the servers on another network segment. Routers do not forward UDP broadcast packets, because most routers are not configured to forward broadcast traffic. Configure a DHCP relay agent to forward DHCP packets to a remote server. Configure a DHCP relay profile on the DHCP relay agent and configure one or more helper addresses in it. Assign the profile to an interface or a VRF.

Summary

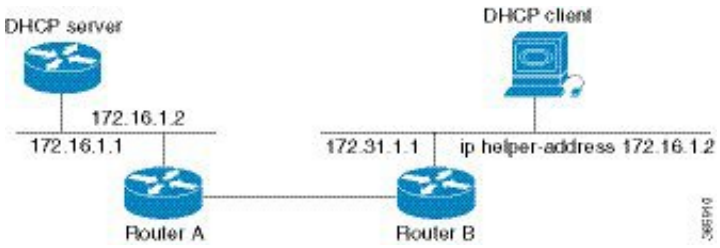
The key components involved in DHCP relay are:

- DHCP client: Broadcasts a request for an IP address and other configuration parameters on its local LAN.
- DHCP relay agent: A router that picks up the client broadcast, inserts the gateway address (giaddr) in the DHCP packet, and unicasts the message to the DHCP server specified by the helper address in the relay profile.
- DHCP server: Receives the relayed message and uses the giaddr to determine which subnet receives the offer and to identify the appropriate IP address range.

The relay agent unicasts messages to the server address specified by the helper address in the relay profile, so that DHCP packets can traverse routed segments between the client and the server.

Workflow

Figure 10: Forward UDP broadcasts to a DHCP server using a helper address



These stages describe how a DHCP relay agent forwards packets between the client and the server:

1. The DHCP client broadcasts a request for an IP address and other configuration parameters on its local LAN.
2. Acting as a DHCP relay agent, Router B picks up the broadcast and inserts the IP address of the interface that receives the DHCP client's packets into the gateway address (giaddr) field of the DHCP packet.
3. The relay agent changes the destination address to the DHCP server's address and sends the message out on another interface. It unicasts the message to the server address (in this case, 172.16.1.2), which is specified by the helper address in the relay profile.
4. The DHCP server uses the giaddr to determine which subnet receives the offer and to identify the appropriate IP address range.

Prerequisites for DHCP relay agent

This topic lists the prerequisites for configuring a DHCP relay agent on the Cisco 8000 Series Router.

These prerequisites apply before you configure a DHCP relay agent:

- You must be in a user group that is 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 that a user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- A configured and running DHCP client and DHCP server.
- Connectivity between the relay agent and DHCP server.

Limitations for DHCP relay

This topic lists the limitations that apply to the DHCP relay feature on the Cisco 8000 Series Router.

These limitations apply to the DHCP relay feature:

- The DHCP relay profile does not support multicast addresses. The **helper-address** command in a DHCP relay profile submode supports a global unicast IP address only as the helper address.
- Relay agents add only interface-id and remote-id DHCP option code while forwarding the packet to a DHCP server.
- DHCP relay profile submode does not support the configuration of DHCP option code.

Configure the DHCP relay agent

This topic describes how to configure and enable the DHCP relay agent on the Cisco 8000 Series Router.

Configure a DHCP relay profile and attach it to a BVI interface to forward DHCP packets between clients and DHCP servers.

Procedure

1. Enter the global configuration mode, and enter the DHCP IPv4 configuration submode.

   Example

   ```
   Router# configure
   Router(config)# dhcp ipv4
   ```

2. Enable the DHCP relay profile.

   Example

   ```
   Router(config-dhcpv4)# profile r1 relay
   ```

3. Configure VRF addresses for forwarding UDP broadcasts, including DHCP.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# helper-address vrf A 10.10.7.1 giaddr
   10.10.7.2
   Router(config-dhcpv4-relay-profile)# broadcast-flag policy check
   ```

4. Insert the DHCP relay agent information option (option-82 field) in forwarded BOOTREQUEST messages to a DHCP server.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# relay information option vpn
   Router(config-dhcpv4-relay-profile)# relay information option vpn-mode rfc
   ```

5. (Optional) Configure the DHCP IPv4 Relay not to discard BOOTREQUEST packets that have an existing relay information option and the `giaddr` set to zero.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# relay information option allow-untrusted
   Router(config-dhcpv4-relay-profile)# exit
   ```

6. Configure DHCP relay on a BVI interface and commit the configuration.

   Example

   ```
   Router(config-dhcpv4)# interface BVI 1 relay profile r1
   Router(config-dhcpv4)# commit
   ```

Enable a DHCP relay agent on an interface

This topic describes how to enable the Cisco IOS XR DHCP relay agent on an interface on the Cisco 8000 Series Router.

The DHCP relay agent is disabled by default. Attach a relay profile to an interface to enable the DHCP relay agent on that interface.

Procedure

1. Enter the global configuration mode and enter the DHCPv4 configuration submode.

   Example

   ```
   Router# configure
   Router(config)# dhcp ipv4
   ```

2. Attach a relay profile to an interface. To disable the DHCP relay on the interface, use the **no interface HundredGigE 0/2/0/2 none** command.

   Example

   ```
   Router(config-dhcpv4)# interface HundredGigE 0/2/0/2 relay profile client
   ```

3. Commit the configuration.

   Example

   ```
   Router(config-dhcpv4-if)# commit
   ```

Disable DHCP relay on an interface

This topic describes how to disable DHCP relay on an interface on the Cisco 8000 Series Router by using the **no** keyword.

Use the **no interface** command with the relay profile name, or with the **none** keyword, to disable DHCP relay on an interface.

Procedure

1. Enter the global configuration mode and enter the DHCPv6 configuration submode.

   Example

   ```
   Router# configure terminal
   Router(config)# dhcp ipv6
   ```

2. Disable DHCP relay on the interface by removing the relay profile attachment.

   Example

   ```
   Router(config-dhcpv6)# no interface type name relay profile profile-name
   Router(config-dhcpv6)# no interface type name none
   ```

3. Commit the configuration.

Example

```
Router(config-dhcpv6-if)# commit
```

Configure and enable the DHCP relay agent with MAC address verification

This topic describes how to configure and enable the DHCP relay agent with DHCP MAC address verification on the Cisco 8000 Series Router.

Enable MAC address verification so that if the MAC address in the DHCPv4 protocol header does not match the L2 header source MAC address in the DHCPv4 relay profile, the frame is dropped.

Procedure

1. Enter the global configuration mode and the DHCPv4 configuration submode, and enable the DHCP relay profile.

   Example

   ```
   Router# configure
   Router(config)# dhcp ipv4
   Router(config-dhcpv4)# profile client relay
   ```

2. Enable MAC address verification.

   Example

   ```
   Router(config-dhcpv4)# client-mac-mismatch action drop
   ```

3. Insert the DHCP relay agent information option (option-82 field) in forwarded BOOTREQUEST messages to a DHCP server.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# relay information option
   ```

4. (Optional) Configure DHCP to check the validity of the relay agent information option in forwarded BOOTREPLY messages.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# relay information check
   ```

5. (Optional) Configure the reforwarding policy for the DHCP relay agent. This example drops the relay information; use the **keep** keyword to keep it.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# relay information policy drop
   ```

6. (Optional) Configure the DHCP IPv4 relay agent not to discard BOOTREQUEST packets that have an existing relay information option and the giaddr set to zero.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# relay information option allow-untrusted
   ```

7. Drop the packet that has an existing nonzero giaddr value. Use the **replace** keyword to replace the existing giaddr value with a generated value (the default behavior).

   Example

   ```
   Router(config-dhcpv4-relay-profile)# giaddr policy drop
   ```

8. Forward UDP broadcasts, including DHCP, to the helper address for the VRF.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# helper-address vrf vrf1 10.1.1.1
   Router(config-dhcpv4-relay-profile)# commit
   Router(config-dhcpv4-relay-profile)# exit
   ```

9. Configure DHCP relay on a VRF and commit the entire configuration.

   Example

   ```
   Router(config-dhcpv4)# vrf vrf1 relay profile client
   Router(config-dhcpv4)# commit
   ```

   Use **show dhcp ipv4 relay statistics raw all** command to verify that DHCP MAC addresses are verified on the router.

   ```
   Router# show dhcp ipv4 relay statistics raw all
   packet_drop_mac_mismatch                    :         0
   ```

Configure the DHCPv6 stateless relay agent

This topic describes how to configure the DHCPv6 stateless relay agent on the Cisco 8000 Series Router by specifying a destination address for client messages and enabling DHCP IPv6 relay service on the interface.

To configure the DHCPv6 stateless relay agent, enable the DHCP IPv6 configuration mode, configure the DHCPv6 relay profile, configure helper addresses, and specify the interface for the relay profile.

Procedure

1. Enter the global configuration mode and enter the DHCP IPv6 configuration mode.

   Example

   ```
   Router# configure terminal
   Router(config)# dhcp ipv6
   ```

2. Configure the DHCPv6 relay profile.

   Example

```
Router(config-dhcpv6)# profile test relay
```

3. Configure the helper address for the relay profile.

   Example

```
Router(config-dhcpv6-relay-profile)# helper-address vrf default 2001:1::1
Router(config-dhcpv6-relay-profile)# !
```

4. Specify the interface for the relay profile.

   Example

```
Router(config-dhcpv6-relay-profile)# interface TenGigE0/0/0/0 relay profile
test
Router(config-dhcpv6)# !
```

Enable DHCP relay on a VRF

This topic describes how to enable DHCP relay on a VRF on the Cisco 8000 Series Router.

Associate an existing DHCP relay profile with a VRF so that DHCP packets that arrive in that VRF are relayed using the profile.

Procedure

1. Enter the global configuration mode and enter the DHCPv4 configuration submode.

   Example

```
Router# configure terminal
Router(config)# dhcp ipv4
```

2. Associate the relay profile with the VRF.

   Example

```
Router(config-dhcpv4)# vrf vrf-name relay profile profile-name
```

3. Commit the configuration.

   Example

```
Router(config-dhcpv4)# commit
```

Configure a DHCP relay profile with multiple helper addresses

This topic describes how to configure a DHCP relay profile with multiple helper addresses on the Cisco 8000 Series Router. You can configure up to 16 IPv4 and IPv6 helper addresses for a DHCPv4 or DHCPv6 relay profile.

Configure the DHCPv6 relay profile with multiple helper addresses and verify the configuration.

Procedure

1. Enter the DHCPv4 or DHCPv6 configuration mode.

   Example

   ```
   Router(config)# dhcp ipv6
   ```

2. Configure the DHCPv4 or DHCPv6 relay profile.

   Example

   ```
   Router(config-dhcpv6)# profile helper relay
   ```

3. Configure the helper addresses.

   > **Note**
   >
   > You can configure up to 16 IPv4 and IPv6 addresses.

   Example

   ```
   Router(config-dhcpv6-relay-profile)# helper-address vrf default 2001:1:1::2
   ```

4. Confirm your configuration.

   Example

   ```
   Router(config-dhcpv6-relay-profile)# show configuration
   !! IOS XR Configuration 0.0.0
   dhcp ipv6
    profile helper relay
     helper-address vrf default 2001:1:1::2
    !
   !
   end
   ```

5. Commit your configuration.

   Example

   ```
   Router(config-dhcpv6-relay-profile)# commit
   ```

6. Exit the configuration mode.

   Example

   ```
   Router(config-dhcpv6-relay-profile)# exit
   ```

   You have successfully configured the DHCPv6 relay helper address.

DHCP relay agent notification for prefix delegation

This topic describes how the DHCP relay agent notification for prefix delegation feature enables the Cisco 8000 Series Router, working as a DHCPv6 relay agent, to find prefix delegation options and manage subscriber routes.

DHCP relay agent notification for prefix delegation allows the router working as a DHCPv6 relay agent to

- find prefix delegation options,
- review the contents of a DHCP RELAY-REPLY packet that is sent to the client,
- extract information about the delegated prefix, and
- insert an IPv4 or IPv6 subscriber route that matches the prefix delegation information onto the relay agent.

A relay agent forwards future packets that are destined to that prefix based on the information contained in the prefix delegation. The relay agent automatically does the subscriber route management.

The IPv4 or IPv6 routes are added when the relay agent relays a RELAY-REPLY packet. The IPv4 or IPv6 routes are deleted when the prefix delegation lease time expires or the relay agent receives a release message. An IPv4 or IPv6 subscriber route in the routing table of the relay agent is updated when the prefix delegation lease time is extended.

This feature leaves an IPv4 or IPv6 route on the routing table of the relay agent. This registered IPv4 or IPv6 address allows a Unicast Reverse Path Forwarding (uRPF) to work by allowing the router to do the reverse lookup. The reverse lookup enables you to confirm that the IPv4 or IPv6 address on the relay agent is not malformed or spoofed. The IPv6 route in the routing table of the relay agent can be redistributed to other routing protocols to advertise the subnets to other nodes. When the client sends a DHCP_DECLINE message, the routes are removed.

Relay agent Option 82 per Ethernet Flow Points

This topic describes DHCP relay agent Option 82 support on the Cisco 8000 Series Router, which lets you insert Circuit ID and Remote ID suboptions per Ethernet Flow Point (EFP) or per ingress Layer 2 interface into forwarded DHCP packets.

In forwarded BOOTREQUEST messages to a DHCP server, you can configure the relay agent to insert Option 82 suboptions in the DHCP packet. The Option 82 suboptions available for configuration are Circuit ID and Remote ID. When a DHCP relay profile is attached to a Bridge Virtual Interface (BVI), you can assign the Option 82 Circuit ID and Remote ID per EFP or per ingress Layer 2 interface. The relay agent then sends the DHCP packet to the server with the included Option 82 Circuit ID or Remote ID. DHCP relay agent Option 82 provides security when DHCP is used to allocate network addresses. Thus, you can enable the DHCP relay agent to prevent DHCP client requests from untrusted sources.

Table 25: Feature History Table

Feature Name	Release	Description
Option 82 support in DHCP packets	Release 26.3.1	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.
Option 82 support in DHCP packets	Release 26.2.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature support is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8011-4G24Y4H-I
Option 82 support in DHCP packets	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.
Option 82 support in DHCP packets	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) * This feature is supported on: • 8712-MOD-M • 8711-48Z-M
Option 82 support in DHCP packets	Release 25.1.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) By utilizing DHCP relay agent Option 82 functionality, you can now enhance the network security by ensuring that DHCP allocates network addresses only to trusted sources. This reduces the risk of unauthorized devices gaining network access. You can achieve this by assigning Option 82 Circuit ID and Remote ID per Ethernet Flow Points (EFP) or per ingress Layer 2 interface in the DHCP packet. *This feature is supported on: • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Benefits of configuring relay agent Option 82

This topic lists the benefits of configuring relay agent Option 82 on the Cisco 8000 Series Router.

Configuring relay agent Option 82 provides these benefits:

- Option 82 allows for better tracking of where DHCP requests originate, making it easier to manage and troubleshoot network connections.
- By identifying the origin of DHCP requests, network resources such as IP addresses can be allocated more efficiently and appropriately based on the source location.

Configure relay agent Option 82 per Ethernet Flow Points

This topic describes how to configure the DHCP relay agent to insert Option 82 suboptions (Circuit ID and Remote ID) per Ethernet Flow Point (EFP) or per ingress Layer 2 interface on the Cisco 8000 Series Router.

Procedure

1. Configure DHCP for IPv4 and enter the DHCPv4 configuration submode.

   Example

   ```
   Router# configure
   Router(config)# dhcp ipv4
   ```

2. Enable the DHCP relay profile.

   Example

   ```
   Router(config-dhcpv4)# profile bvi1_profile relay
   ```

3. Configure VRF addresses for forwarding UDP broadcasts, including DHCP.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# helper-address 192.0.2.1
   ```

4. Insert the DHCP relay agent information option (option-82 field) in forwarded BOOTREQUEST messages to a DHCP server.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# relay information option
   Router(config-dhcpv4-relay-profile)# exit
   ```

5. Enable DHCP for IPv4 on a BVI interface and attach the profile as the relay profile for the BVI interface.

   Example

   ```
   Router(config-dhcpv4)# interface BVI1 relay profile bvi1_profile
   ```

6. Enable the DHCP relay agent to add the Option 82 Circuit ID field in ASCII or hex format per EFP to the DHCP packet.

   Example

   ```
   Router(config-dhcpv4)# interface Bundle-Ether50.103 relay information option
    circuit-id format-type ascii 110
   ```

7. Enable the DHCP relay agent to add the Option 82 Remote ID field in ASCII or hex format per EFP to the DHCP packet.

Example

```
Router(config-dhcpv4)# interface Bundle-Ether50.103 relay information option
 remote-id format-type ascii 110
Router(config-dhcpv4)# commit
```

DHCPv6 relay over BVI for IANA address allocation

This topic describes DHCPv6 relay over a Bridge Virtual Interface (BVI) for Internet Assigned Numbers Authority (IANA) address allocation on the Cisco 8000 Series Router, which lets a DHCPv6 relay agent act as a stateful relay and maintain client bindings and route entries.

DHCPv6 relay agents relay all packets that are coming from DHCPv6 clients over the access-interfaces toward external DHCPv6 servers. DHCPv6 relay agents request IP addresses (::/128) through IANA allocation for the DHCPv6 clients. DHCPv6 relay agents also receive response packets from the DHCPv6 servers and forward the packets toward DHCPv6 clients over BVI interfaces. DHCPv6 relay agents act as stateless, by default, for DHCPv6 clients. DHCPv6 clients do not maintain any DHCPv6 binding and respective route entry for the allocated IP addresses. You can enable a DHCPv6 client to get a particular IPv6 address assigned by the DHCPv6 server over a BVI through IANA address allocation. Therefore, the DHCPv6 relay agent acts as a stateful relay agent and maintains DHCPv6 binding and respective route entry for the allocated IPv6 addresses.

Table 26: Feature History Table

Feature Name	Release	Description
DHCPv6 Relay Over BVI for IANA Address Allocation	Release 26.2.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is supported on: • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
DHCPv6 Relay Over BVI for IANA Address Allocation	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.

Restrictions for DHCPv6 relay over BVI for IANA address allocation

This topic lists the restrictions for DHCPv6 relay over a Bridge Virtual Interface (BVI) for IANA address allocation on the Cisco 8000 Series Router.

These restrictions apply to DHCPv6 relay over BVI for IANA address allocation:

- You can configure up to 500 client sessions over a BVI interface for DHCP relay.
- You can configure up to 8 DHCPv6 server addresses for each DHCPv6 relay profile.

Configure DHCPv6 relay over BVI for IANA address allocation

This topic describes how to configure DHCPv6 relay over a Bridge Virtual Interface (BVI) for Internet Assigned Numbers Authority (IANA) address allocation on the Cisco 8000 Series Router.

Procedure

1. Enter the interface configuration mode and configure a BVI interface.

   Example

   ```
   Router# configure
   Router(config)# interface BVI1
   ```

2. Assign an IPv6 address to the BVI interface.

   Example

   ```
   Router(config-if)# ipv6 address 2001:db8::2/64
   Router(config-if)# commit
   Router(config-if)# exit
   ```

3. Route the L2 access interface to the L3 BVI interface of the relay agent.

   Example

   ```
   Router(config)# l2vpn bridge group 1
   Router(config-l2vpn-bg)# bridge-domain 1
   Router(config-l2vpn-bg-bd)# interface hundredGigE 0/0/0/1.100
   Router(config-l2vpn-bg-bd-ac)# commit
   Router(config-l2vpn-bg-bd-ac)# exit
   Router(config-l2vpn-bg-bd)# routed interface BVI1
   Router(config-l2vpn-bg-bd)# exit
   Router(config-l2vpn-bg)# exit
   Router(config-l2vpn)# exit
   ```

4. Enter the DHCP IPv6 configuration mode and then create a DHCP IPv6 stateful relay profile.

   Example

   ```
   Router(config)# dhcp ipv6
   Router(config-dhcpv6)# profile RELAY1 relay
   ```

5. Attach the relay profile to a server address.

   Example

   ```
   Router(config-dhcpv6-relay-profile)# helper-address vrf default 2001:DB8::1
   ```

6. Configure a stateful relay agent by enabling route allocation through IANA.

Example

```
Router(config-dhcpv6-relay-profile)# iana-route-add
Router(config-dhcpv6-relay-profile)# exit
```

7. Attach the BVI interface to the DHCPv6 relay profile and commit the configuration.

Example

```
Router(config-dhcpv6)# interface BVI1 relay profile RELAY1
Router(config-dhcpv6)# commit
```

8. Verify that more than one DHCP client is bridged over the BVI.

Example

```
Router# show dhcp ipv6 relay binding
Summary:
Total number of clients: 500

IPv6 Address: 2000::418f/128 (BVI31)
    Client DUID: 000100015dcf28de001094003295
    MAC Address: 0010.9400.3295
    IAID: 0x0
    VRF: default
    Lifetime: 600 secs (00:10:00)
    Expiration: 533 secs (00:08:53)
    L2Intf AC: Bundle-Ether3.1
    SERG State: NONE
    SERG Intf State: SERG-NONE
IPv6 Address: 2000::4190/128 (BVI31)
Client DUID: 000100015dcf28de001094003296
MAC Address: 0010.9400.3296
IAID: 0x0
VRF: default
Lifetime: 600 secs (00:10:00)
Expiration: 531 secs (00:08:51)
L2Intf AC: Bundle-Ether3.1
SERG State: NONE
SERG Intf State: SERG-NONE
IPv6 Address: 2000::4191/128 (BVI31)
Client DUID: 000100015dcf28de001094003297
MAC Address: 0010.9400.3297
IAID: 0x0
VRF: default
Lifetime: 600 secs (00:10:00)
Expiration: 448 secs (00:07:28)
L2Intf AC: Bundle-Ether3.1
SERG State: NONE
SERG Intf State: SERG-NONE
IPv6 Address: 2000::4192/128 (BVI31)
Client DUID: 000100015dcf28de001094003298
MAC Address: 0010.9400.3298
IAID: 0x0
VRF: default
Lifetime: 600 secs (00:10:00)
Expiration: 439 secs (00:07:19)
L2Intf AC: Bundle-Ether3.1
SERG State: NONE
SERG Intf State: SERG-NONE
```

9. Verify that a unique IPv6 address is assigned to a client due to IANA allocation.

Example

```
Router# show route ipv6
Mon Oct 21 06:16:43.617 UTC
Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
U - per-user static route, o - ODR, L - local, G - DAGR, l - LISP
A - access/subscriber, a - Application route
M - mobile route, r - RPL, t - Traffic Engineering, (!) - FRR Backup path
Gateway of last resort is not set
A 2000::/64
[1/0] via fe80::1, 00:00:37, BVI700
A 2000::1/128
[1/0] via fe80::210:94ff:fe00:8, 00:00:12, BVI700
C 2007:3019::/64 is directly connected,
00:00:37, Loopback1
L 2007:3019::1/128 is directly connected,
00:00:37, Loopback1
C 7001:6018::/64 is directly connected,
00:00:37, BVI700
L 7001:6018::1/128 is directly connected,
00:00:37, BVI700
C 7001:6019::/64 is directly connected,
00:00:37, TenGigE0/0/0/2.2
L 7001:6019::1/128 is directly connected,
00:00:37, TenGigE0/0/0/2.2
```

Configure the relay agent information feature

This topic describes how to configure the DHCP relay agent information option on the Cisco 8000 Series Router.

A DHCP relay agent may receive a message from another DHCP relay agent that already contains relay information. By default, the relay information from the previous relay agent is replaced (using the replace option).

Procedure

1. Enter the DHCP IPv4 configuration submode.

Example

```
Router# configure
Router(config)# dhcp ipv4
```

2. Enter the DHCP IPv4 profile relay submode.

Example

```
Router(config-dhcpv4)# profile RELAY relay
```

3. Configure the system to insert the DHCP relay agent information option (option-82 field) in forwarded BOOTREQUEST messages to a DHCP server.

   Note

   - This option is injected by the relay agent while forwarding client-originated DHCP packets to the server. Servers recognizing this option can use the information to implement IP address or other parameter assignment policies. When replying, the DHCP server echoes the option back to the relay agent. The relay agent removes the option before forwarding the reply to the client.
   - The relay agent information is organized as a single DHCP option that contains one or more suboptions. These options contain the information known by the relay agent. The supported suboptions are Remote ID and Circuit ID.
   - This function is disabled by default. The port field of the default circuit-ID denotes the configured bundle-ID of the bundle. If circuit IDs require that bundles be unique, and because the port field is 8 bits, the low-order 8 bits of configured bundle IDs must be unique. To achieve this, configure bundle IDs within the range from 0 to 255.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# relay information option
   ```

4. (Optional) Configure DHCP to check the validity of the relay agent information option in forwarded BOOTREPLY messages.

   Note

   - If an invalid message is received, the relay agent drops the message. If a valid message is received, the relay agent removes the relay agent information option field and forwards the packet.
   - By default, DHCP does not check the validity of the relay agent information option field in DHCP reply packets received from the DHCP server.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# relay information check
   ```

5. (Optional) Configure the reforwarding policy for the DHCP relay agent. That is, whether the relay agent drops or keeps the relay information.

> **Note**
>
> By default, the DHCP relay agent replaces the relay information option.

Example

```
Router(config)# dhcp ipv4 profile TEST relay relay information policy drop
Router(config)# commit
```

Configure the relay agent giaddr policy

This topic describes how to configure the DHCP relay agent's processing policy for BOOTREQUEST packets that already contain a nonzero giaddr attribute on the Cisco 8000 Series Router.

Use the **giaddr policy replace** command to replace the existing giaddr value with a value that the relay agent generates. Use the **giaddr policy drop** command to drop the packet that has an existing nonzero giaddr value.

Procedure

1. Enter the DHCP IPv4 configuration submode.

   Example

   ```
   Router# configure
   Router(config)# dhcp ipv4
   ```

2. Enter the relay profile submode.

   Example

   ```
   Router(config-dhcpv4)# profile client relay
   ```

3. Configure the giaddr policy and commit the configuration.

   Example

   ```
   Router(config-dhcpv4-relay-profile)# giaddr policy drop
   Router(config-dhcpv4-relay-profile)# commit
   ```

DHCP snooping

This topic describes DHCP snooping on the Cisco 8000 Series Router, which enhances network security by preventing unauthorized DHCP servers from distributing IP addresses.

DHCP snooping is a Layer 2 security feature that

- maintains a list of approved servers so that only trusted DHCP messages are processed,
- safeguards against IP address spoofing and man-in-the-middle attacks by inspecting and filtering DHCP packets, and
- integrates with existing network configurations to provide protection while maintaining efficient IP address management.

Table 27: Feature History Table

Feature Name	Release Information	Feature Description
DHCP Snooping	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
DHCP Snooping	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is supported on Cisco 8011-4G24Y4H-I routers.
DHCP Snooping	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*) DHCP snooping enhances network security and prevents unauthorized or malicious DHCP servers from distributing IP addresses. By maintaining a list of approved servers, DHCP snooping ensures that only trusted DHCP messages are processed. This feature safeguards against IP address spoofing and man-in-the-middle attacks by inspecting and filtering DHCP packets. This reliable security measure integrates seamlessly with existing network configurations, offering robust protection for your network infrastructure while maintaining efficient IP address management. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • Cisco 8712-MOD-M routers • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

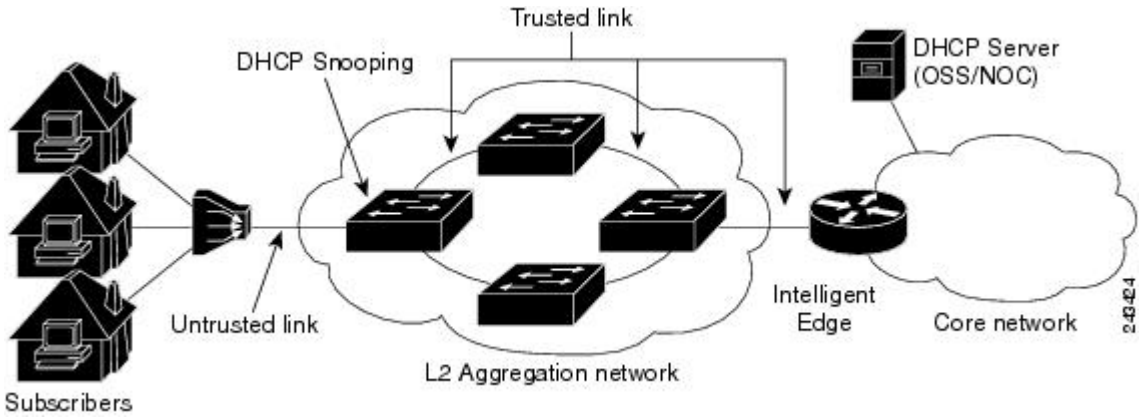
DHCP snooping features are focused on the edge of the aggregation network. Security features are applied at the first point of entry for subscribers. The relay agent information option is used to identify the subscriber's line, which is either the DSL line to the subscriber's home or the first port in the aggregation network.

The central concept for DHCP snooping is that of trusted and untrusted links. A trusted link provides secure access for traffic on that link. On an untrusted link, subscriber identity and subscriber traffic cannot be determined. DHCP snooping runs on untrusted links to provide subscriber identity. The following figure shows an aggregation network. The link from the DSLAM to the aggregation network is untrusted and is the point of presence for DHCP snooping. The links that connect

the switches in the aggregation network and the link from the aggregation network to the intelligent edge are considered trusted.

> **Note**
>
> Enabling both DHCP relay on a BVI and DHCP snooping in a bridge domain that has a BVI can result in duplicate DHCP messages from the DHCP client to the DHCP server.

Figure 11: DHCP snooping in an aggregation network



Prerequisites for DHCP snooping

This topic lists the prerequisites for configuring DHCP snooping on the Cisco 8000 Series Router.

These prerequisites apply before you configure DHCP snooping:

- You must be in a user group that is 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 that user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- A Cisco 8000 Series Router running Cisco IOS XR software.
- A configured and running DHCP client and DHCP server.

Configuration guidelines and restrictions for DHCP snooping

This topic lists the configuration guidelines and restrictions that apply to DHCP snooping on the Cisco 8000 Series Router.

These configuration guidelines and restrictions apply to DHCP snooping:

- Supported on AC bridge-ports, Access Pseudowire (PW), and core PW.
- Not supported on bridge-port types, such as Ethernet VPN Instance, which is used in Ethernet VPN (EVPN) configurations.

Trusted and untrusted ports

This topic describes how DHCP snooping treats packets on trusted and untrusted ports on the Cisco 8000 Series Router.

On trusted ports, DHCP BOOTREQUEST packets are forwarded by DHCP snooping. The client's address lease is not tracked and the client is not bound to the port. DHCP BOOTREPLY packets are forwarded.

When the first DHCP BOOTREQUEST packet from a client is received on an untrusted port, DHCP snooping binds the client to the bridge port and tracks the client's address lease. When that address lease expires, the client is deleted from the database and is unbound from the bridge port. Packets from this client received on this bridge port are processed and forwarded as long as the binding exists. Packets that are received on another bridge port from this client are dropped while the binding exists. DHCP snooping only forwards DHCP BOOTREPLY packets for this client on the bridge port that the client is bound to. DHCP BOOTREPLY packets that are received on untrusted ports are not forwarded.

DHCP snooping in a bridge domain

This topic describes how DHCP snooping profiles are applied in a bridge domain on the Cisco 8000 Series Router.

To enable DHCP snooping in a bridge domain, there must be at least two profiles, a trusted profile and an untrusted profile. The untrusted profile is assigned to the client-facing ports, and the trusted profile is assigned to the server-facing ports. In most cases, there are many client-facing ports and few server-facing ports. The simplest example is two ports, a client-facing port and a server-facing port, with an untrusted profile explicitly assigned to the client-facing port and a trusted profile assigned to the server-facing port.

Assign profiles to a bridge domain

This topic describes how to assign DHCP snooping profiles to a bridge domain on the Cisco 8000 Series Router.

Because there are normally many client-facing ports and a small number of server-facing ports, the operator assigns the untrusted profile to the bridge domain. This configuration effectively assigns an untrusted profile to every port in the bridge domain. This action saves the operator from explicitly assigning the untrusted profile to all of the client-facing ports. Because there also must be server-facing ports that have trusted DHCP snooping profiles, in order for DHCP snooping to function properly, this untrusted DHCP snooping profile assignment is overridden on server-facing ports by specifically configuring trusted DHCP snooping profiles on the server-facing ports. For ports in the bridge domain that do not require DHCP snooping, all should have the **none** profile assigned to them to disable DHCP snooping on those ports.

Create DHCP snooping profiles

This topic describes how to create an untrusted DHCP snooping profile for the client port and a trusted DHCP snooping profile for the server port on the Cisco 8000 Series Router.

These steps create an untrusted DHCP snooping profile for the client port and a trusted DHCP snooping profile for the server port, and then enter the l2vpn configuration mode in preparation for attaching the profiles to bridge ports.

Procedure

1. Enter the DHCP IPv4 profile configuration submode.

   Example

   ```
   Router(config)# dhcp ipv4
   ```

2. Configure an untrusted DHCP snooping profile for the client port using the **profile** *untrusted-profile-name* **snoop** command.

   Example

   ```
   Router(config-dhcpv4)# profile untrustedClientProfile snoop
   ```

3. Exit the DHCP IPv4 profile configuration mode.

Example

```
Router(config-dhcpv4)# exit
```

4. Enable DHCP for IPv4 and enter the DHCP IPv4 profile configuration mode.

   Example

   ```
   Router(config)# dhcp ipv4
   ```

5. Configure a trusted DHCP snooping profile for the server port using the **profile** *profile-name* **snoop** command.

   Example

   ```
   Router(config-dhcpv4)# profile trustedServerProfile snoop
   ```

6. Configure the DHCP snoop profile as trusted using the **trusted** command.

   Example

   ```
   Router(config-dhcpv4)# trusted
   ```

7. Exit the DHCP IPv4 profile configuration mode.

   Example

   ```
   Router(config-dhcpv4)# exit
   ```

8. Enter the l2vpn configuration mode.

   Example

   ```
   Router(config)# l2vpn
   ```

What to do next
Attach snooping profiles to bridge ports.

Attach snooping profiles to bridge ports

This topic describes how to create the client-facing and server-facing bridge ports in a bridge domain and attach the corresponding DHCP snooping profiles on the Cisco 8000 Series Router.

These steps create the bridge group and bridge domain, add the client-facing and server-facing interfaces as bridge ports, and attach the untrusted DHCP snooping profile to the client bridge port and the trusted DHCP snooping profile to the server bridge port.

Procedure

---

1. Create a bridge group to contain bridge domains and enter the l2vpn bridge group configuration submode.

Example

```
Router(config-l2vpn)# bridge group ccc
```

2. Establish a bridge domain.

Example

```
Router(config-l2vpn-bg)# bridge-domain ddd
```

3. Identify the client-facing interface.

Example

```
Router(config-l2vpn-bg-bd)# interface gigabitethernet 0/1/0/0
```

4. Attach an untrusted DHCP snoop profile to the bridge port.

Example

```
Router(config-l2vpn-bg-bd-ac)# dhcp ipv4 snoop profile untrustedClientProfile
```

5. Identify the server-facing interface.

Example

```
Router(config-l2vpn-bg-bd-ac)# interface gigabitethernet 0/1/0/1
```

6. Attach a trusted DHCP snoop profile to the bridge port.

Example

```
Router(config-l2vpn-bg-bd-ac)# dhcp ipv4 snoop profile trustedServerProfile
```

7. Exit the l2vpn bridge group bridge-domain interface configuration submode.

Example

```
Router(config-l2vpn-bg-bd-ac)# exit
```

8. Exit the l2vpn bridge group bridge-domain configuration submode.

Example

```
Router(config-l2vpn-bg-bd)# exit
```

9. Commit the configuration changes on the router.

Example

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

Disable DHCP snooping on a specific bridge port

This topic describes how to disable DHCP snooping on a specific bridge port in a bridge domain on the Cisco 8000 Series Router.

The following configuration enables DHCP to snoop packets on all bridge ports in the bridge domain ISP1 except for bridge ports GigabitEthernet 0/1/0/1 and GigabitEthernet 0/1/0/2. DHCP snooping is disabled on bridge port GigabitEthernet 0/1/0/1. Bridge port GigabitEthernet 0/1/0/2 is the trusted port that connects to the server. In this example, no additional features are enabled, so only DHCP snooping is running.

Procedure

---

1. Enter the l2vpn configuration submode.

   Example

   ```
   Router(config)# l2vpn
   ```

2. Create a bridge group to contain bridge domains and enter the l2vpn bridge group configuration submode.

   Example

   ```
   Router(config-l2vpn)# bridge group GRP1
   ```

3. Establish a bridge domain and enter the l2vpn bridge group bridge-domain configuration submode.

   Example

   ```
   Router(config-l2vpn-bg)# bridge-domain ISP1
   ```

4. Attach the untrusted DHCP snooping profile to the bridge domain.

   Example

   ```
   Router(config-l2vpn-bg-bd)# dhcp ipv4 snoop profile untrustedClientProfile
   ```

5. Identify the bridge port where DHCP snooping is to be disabled.

   Example

   ```
   Router(config-l2vpn-bg-bd)# interface gigabitethernet 0/1/0/1
   ```

6. Disable DHCP snooping on the port.

   Example

   ```
   Router(config-l2vpn-bg-bd-if)# dhcp ipv4 none
   ```

7. Identify the trusted server bridge port.

   Example

   ```
   Router(config-l2vpn-bg-bd)# interface gigabitethernet 0/1/0/2
   ```

8. Attach the trusted DHCP snooping profile to that port.

   Example

   ```
   Router(config-l2vpn-bg-bd)# dhcp ipv4 snoop profile trustedServerProfile
   ```

9. Exit the l2vpn bridge-domain bridge group interface configuration submode.

   Example

   ```
   Router(config-l2vpn-bd-bg)# exit
   ```

10. Exit the l2vpn bridge-domain submode.

    Example

    ```
    Router(config-l2vpn-bg)# exit
    ```

11. Commit the configuration changes on the router.

    Example

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

Relay information options

This topic describes the DHCP snooping relay information option (option 82) handling on the Cisco 8000 Series Router.

You can configure a DHCP snooping profile to insert the relay information option (option 82) into DHCP client packets only when it is assigned to a client port. The **relay information option allow-untrusted** command addresses what to do with DHCP client packets when there is a null giaddr and a relay-information option is already in the client packet when it is received. This is a different condition than a DHCP snooping trusted or untrusted port. The **relay information option allow-untrusted** command determines how the DHCP snooping application handles untrusted relay information options.

Use the relay information option

This topic describes how to use the relay information commands to insert the relay information option (option 82) into DHCP client packets and forward DHCP packets with untrusted relay information options on the Cisco 8000 Series Router.

Procedure

1. Enter the DHCP IPv4 profile configuration submode.

Example

```
Router# configure
Router(config)# dhcp ipv4
```

2. Configure an untrusted DHCP snooping profile for the client port.

Example

```
Router(config-dhcpv4)# profile untrustedClientProfile snoop
```

3. Enable the system to insert the DHCP relay information option field in forwarded BOOTREQUEST messages to a DHCP server.

Example

```
Router(config-dhcpv4-snoop-profile)# relay information option
```

4. Commit the configuration changes on the router.

Example

```
Router(config-dhcpv4-snoop-profile)# commit
```

DHCPv6 relay source address

This topic describes the DHCPv6 relay source address feature on the Cisco 8000 Series Router.

The DHCPv6 relay source address is a configuration feature that

- allows the selection of a specific IPv6 address from the relay source-interface,
- ensures consistent source address selection for DHCPv6 server communication, and
- eliminates the need to modify firewall rules when new IPv6 addresses are added to an interface.

Table 28: Feature History Table

Feature Name	Release Information	Feature Description
Configure DHCPv6 relay source address	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]) You can now select an IPv6 address from the configured relay source-interface to be used as the source address for forwarding packets to a server. By selecting a fixed source address, the need to frequently update firewall rules when new, lower-value IPv6 addresses are added is minimized. Previously, the router automatically used the lowest numbered IPv6 address configured on that interface as the source address.

Benefits of DHCPv6 relay source address

This topic lists the benefits of the DHCPv6 relay source address feature on the Cisco 8000 Series Router.

The DHCPv6 relay source address feature provides these benefits:

- Flexibility – You can now choose a specific IPv6 address from the relay source-interface, allowing more control over which address is used for packet forwarding.
- Reduced firewall updates – By selecting a fixed source address, the need to frequently update firewall rules when new, lower-value IPv6 addresses are added is minimized.
- Enhanced security – With a stable relay source address, security policies can be more consistently applied, reducing the risk of misconfigurations.
- Improved troubleshooting – Having a predictable relay source address makes it easier to track and troubleshoot network traffic issues.

Configuration guidelines and restrictions for DHCPv6 relay source address

This topic lists the configuration guidelines and restrictions that apply to the DHCPv6 relay source address feature on the Cisco 8000 Series Router.

These configuration guidelines and restrictions apply to the DHCPv6 relay source address feature:

- Removing the configured DHCPv6 relay source-interface IP disables the feature and retains the old behavior of automatically using the lowest numbered IPv6 address configured on that interface as the source address.
- You can configure the DHCPv6 relay source address under **helper-address** and in the **source-interface** as well.

Configure DHCPv6 relay source address

This topic describes how to configure a specific IPv6 address from the relay source-interface as the source address for forwarding packets to a DHCPv6 server on the Cisco 8000 Series Router.

Procedure

1. Enter the global configuration mode and then enter the DHCP IPv6 configuration mode.

   Example

   ```
   Router# configure terminal
   Router(config)# dhcp ipv6
   ```

2. Create a DHCPv6 relay profile.

   Example

   ```
   Router(config-dhcpv6)# profile test relay
   ```

3. Configure the helper address, specifying the interface and source address to use for packets forwarded to that server.

Example

```
Router(config-dhcpv6-relay-profile)# helper-address vrf default 2011::3
TenGigE0/0/0/0 1001::10
Router(config-dhcpv6-relay-profile)# helper-address vrf default 2011::4
```

4. Configure the source interface and source address for the relay profile.

Example

```
Router(config-dhcpv6-relay-profile)# source-interface TenGigE0/1/0/0 1001::1
```

5. Attach the relay profile to an interface and commit the configuration.

> **Note**
>
> To disable this feature, use the **no** form of the command.

Example

```
Router(config-dhcpv6-relay-profile)# interface Bundle-Ether1 relay profile
test
Router(config-dhcpv6-relay-profile)# commit
```

6. Verify the DHCPv6 relay source address configuration.

Example

```
Router# show running-config interface TenGigE0/0/0/0
dhcp ipv6
 profile test relay
  helper-address vrf default 2011::3 TenGigE0/0/0/0 1001::10
  helper-address vrf default 2011::4
  source-interface TenGigE0/1/0/0 1001::1
!
interface Bundle-Ether1 relay profile test
```

In this example, when a packet is received on the **profile test relay**:

- The address `1001::10`, specified as the DHCPv6 relay source address on interface `TenGigE0/0/0/0`, is used to send packets to server `2011::3`.
- The address `1001::1`, specified as the DHCPv6 relay source address on interface `TenGigE0/1/0/0`, is used to send packets to server `2011::4`.

DHCP client

This topic describes the DHCP client functionality on the Cisco 8000 Series Router, which allows router interfaces to dynamically acquire IP address and server information.

The DHCP client functionality

- allows router interfaces to dynamically acquire the IPv4 or IPv6 address and DHCPv4 or DHCPv6 server information, and
- forwards responses to the correct Layer 2 address, ensuring that each device receives the appropriate configuration information.

Table 29: Feature History Table

Feature Name	Release	Description
Support for DHCP client	Release 26.3.1	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.
Support for DHCP client	Release 26.2.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature support is now supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8011-4G24Y4H-I
Support for DHCP client	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.
Support for DHCP client	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) * This feature is supported on: • 8712-MOD-M • 8711-48Z-M
Support for DHCP client	Release 25.2.1	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH

Feature Name	Release	Description
Support for DHCP client	Release 25.1.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) This release introduces the Dynamic Host Configuration Protocol (DHCP) client functionality. DHCP client is crucial for efficiently managing IP address allocation in a network. It enables router interfaces to dynamically obtain server information for DHCPv4 or DHCPv6, and ensures that the configuration responses are sent to the appropriate Layer 2 addresses. This process ensures that each device on the network receives the correct configuration data. *This feature is supported on: • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

DHCP can allocate IP addresses only for a specified duration, known as the lease period. If a client needs to retain an IP address for a longer period beyond the lease period, the lease period must be renewed before the IP address expires. The client renews the lease period based on the server's configuration by unicasting a REQUEST message to the server's IP address. Upon receiving the REQUEST, the server responds with an ACK message, extending the client's lease period as specified in the ACK.

Configuration guideline for DHCP clients

Either DHCPv4, DHCPv6, static IPv4, or static IPv6 can be configured on an interface.

Enable DHCP client on an interface

This topic describes how to enable a DHCPv4 or DHCPv6 client at the interface level and configure DHCPv6 client options on the Cisco 8000 Series Router.

The DHCPv4 or DHCPv6 client can be enabled at an interface level. The DHCP component receives a notification when DHCPv4 or DHCPv6 is enabled or disabled on an interface.

```
Router# configure
Router(config)# interface bvi
Router(config)# interface bvi_interface_name ipv6 address dhcp
```

You can configure DHCPv6 client on BVI interfaces. You can configure different DHCPv6 client options to differentiate between clients as required. The different DHCPv6 client options are also configured to differentiate how a DHCPv6 client communicates with a DHCPv6 server. The different DHCPv6 client options that you can configure are:

- **DUID**: If the DUID DHCPv6 client option is configured on an interface, DHCPv6 client communicates with the DHCPv6 server through the link layer address.
- **Rapid Commit**: If the Rapid Commit DHCPv6 client option is configured on an interface, DHCPv6 client can obtain configuration parameters from the DHCPv6 server through a rapid two-step exchange (solicit and reply) instead of the default four-step exchange (solicit, advertise, request, and reply).
- DHCP Options: The various other DHCPv6 options that can be configured on a DHCPv6 client are:
  - **Option 15**: Option 15 is also known as the User Class option and it is used by a DHCPv6 client to identify the type or category of users or applications it represents.

- **Option 16**: Option 16 is also known as the Vendor ID option and it is used by a DHCPv6 a client to identify the vendor that manufactured the hardware on which the client is running.
- **Option 23**: Option 23 is also known as the Domain name Server (DNS) option provides a list of one or more IPv6 addresses of DNS recursive name servers to which a client's DNS resolver can send DNS queries.
- **Option 24**: Option 24 is also known as the Domain List option and it specifies the domain search list that the client uses to resolve hostnames with the DNS.

- **DHCP Timers**: This option is used to set different timer value for DHCP client configurations. The various DHCP timer options are:
  - **Release-timeout**: It is used to set retransmission timeout value for the initial release message.
  - **Req-max-rt**: It is used to set the maximum retransmission timeout value for the request message.
  - **Req-timeout**: It is used to set the initial request timeout value of the request message.
  - **Sol-max-delay**: It is used to set the maximum delay time of the first solicit message.
  - **Sol-max-rt**: It is used to set the maximum solicit retransmission time.
  - **Sol-time-out**: It is used to set the intial timeout value of the solicit message.

Procedure

1. Enter the global configuration mode, and then configure a BVI interface.

   Example

   ```
   Router# configure
   Router(config)# interface BVI 1
   ```

2. Enter the DHCPv6 client mode and commit the configuration.

   Example

   ```
   Router(config)# ipv6 address dhcp
   Router(config)# commit
   ```

To verify the DHCPv6 client mode, use the **show dhcp ipv6 client** command:

```
Router# show dhcp ipv6 client
Interface name IPv6 Address State Lease Time Rem
---------------------- -------------- --------------- --------------------
BVI1 2001::1:2/128 BOUND 2591973
```

What to do next

Associated commands:

- ipv6 address dhcp-client-options
- clear dhcp ipv6 client
- show dhcp ipv6 client

- show dhcp ipv4 client
- show tech-support dhcp ipv6 client

6 Host Services and Applications

Topics:

Describes the router services and applications used to diagnose network paths, resolve hostnames, transfer files, provide network access, identify syslog traffic, and configure HTTP client behavior.

Host services and applications are Cisco IOS XR functions that

- test reachability and identify the paths that packets take through a network
- resolve hostnames and transfer files between local and remote systems, and
- provide and configure router-based network application services.

Install the optional RPM package for a service before you use that service. For example, install the Telnet RPM before you enable Telnet.

Network diagnostic services

Explains how ping, traceroute, enhanced ICMP diagnostics, and ICMP probes test reachability, expose packet paths, and report interface or neighbor status, with tasks for running and configuring each utility.

Network diagnostic services are utilities that

- test whether one or more destinations respond
- identify Layer 3 hops and routing failures, and
- provide interface, next-hop, and neighbor details for troubleshooting.

Table 30: Feature History Table

Feature Name	Release Information	Feature Description
Network Connectivity Tools: Ping and Traceroute	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
Network Connectivity Tools: Ping and Traceroute	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.
Network Connectivity Tools: Ping and Traceroute	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*) The **ping** command can be used to verify connectivity between devices, the **traceroute** command can be used to discover the paths packets take to a remote destination and where routing breaks down. *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Ping utilities

Explains ping utilities, how ICMP echo exchanges determine reachability, and the basic, extended, and bulk procedures used to check one or multiple IPv4 destinations.

A ping utility is a network diagnostic tool that

- exchanges Internet Control Message Protocol (ICMP) echo request and echo reply messages with a destination
- measures the time required to receive each reply, and
- supports basic, extended, and bulk reachability tests.

A basic ping tests a destination by using the outgoing interface address as the packet source.

An extended ping lets you select the source address, timeout, datagram size, and other parameters.

A bulk ping accepts multiple destinations directly from the CLI and supports IPv4 destinations only.

How ping verifies reachability

Describes how ping exchanges ICMP echo messages, measures reply time, and determines whether a destination is reachable within the configured timeout interval.

Ping uses Internet Control Message Protocol (ICMP) echo messages to evaluate path reliability, path delay, and destination availability.

Summary

The process uses these messages.

- ICMP echo request: The source sends a query to the destination.
- ICMP echo reply: The destination returns a response to the source.

Workflow

These stages describe how ping verifies reachability.

1. The source sends an ICMP echo request to the specified address and starts the reply timer.
2. The request traverses the network to the destination. A basic ping uses the outgoing interface address as its source unless you select another source with extended ping.
3. The destination returns an ICMP echo reply. The test succeeds when the reply reaches the source within the configured timeout.

Result

The output reports the success rate and can report round-trip timing statistics.

Extended ping connectivity tests

Explains how selecting a source address with extended ping isolates bidirectional routing between two router interfaces and helps distinguish routing failures from host default-gateway problems.

An extended ping connectivity test is a diagnostic method that

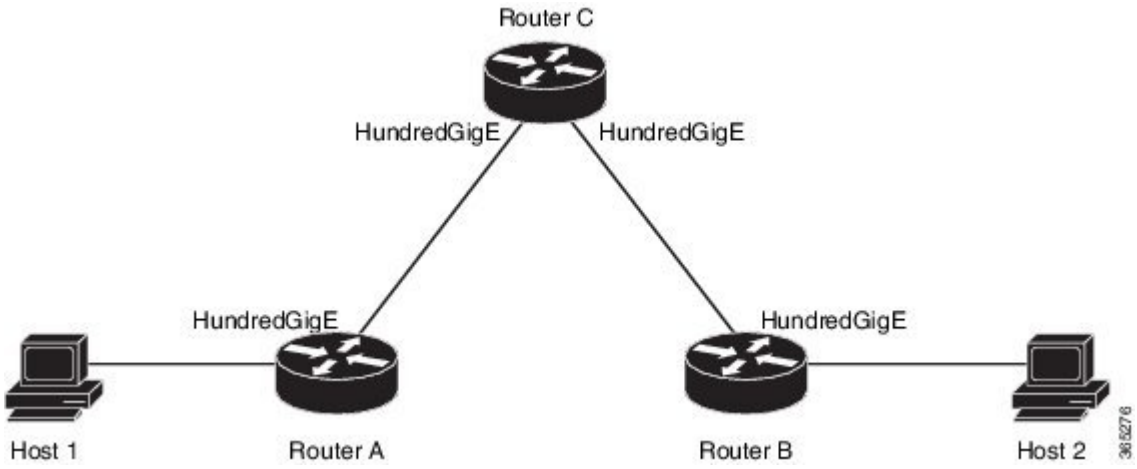
- uses a selected router interface address as the source of an ICMP echo request
- tests the forward and return routes between two specified interfaces, and
- isolates routing problems before you troubleshoot the attached hosts.

A basic ping normally uses the address of the outgoing interface as its source. Extended ping lets you select the source address that represents the path you want to test.

Extended ping connectivity scenario

The example tests connectivity from the HundredGigE interface on Router A at 192.0.2.2 to the interface on Router B at 192.0.2.1.

Figure 12: Extended ping connectivity path



A successful extended ping confirms these routing conditions.

- Router A has a route to the Router B interface.
- Router B has a return route to the Router A source interface.
- The intermediate routers can forward traffic between the two interface subnets.

Failed test interpretation

A failed extended ping can indicate a missing route at any router in the path.

Table 31: Potential routing failures

Router	Potential missing route
Router A	A route to the Router B interface subnet or to the subnet between Router C and Router B.
Router B	A route to the Router A interface subnet or to the subnet between Router C and Router A.
Router C	A route to the Ethernet segment attached to Router A or Router B.

Correct the routing problem before Host 1 pings Host 2. If the hosts still cannot communicate after the extended ping succeeds, verify the default gateway on each host.

Check network connectivity with ping

Checks reachability between two addresses and uses an extended ping when you must select the source address, timeout, datagram size, or other test parameters.

Use ping to confirm that the destination responds and that return routing is available.

An extended ping can test connectivity between specific router interfaces instead of using the outgoing interface address automatically.

Procedure

1. Run a basic ping to the destination.

   Example

   ```
   Router# ping 192.0.2.1
   Type escape sequence to abort.
   Sending 5, 100-byte ICMP Echos to 192.0.2.1, timeout is 2 seconds:
   !!!!!
   Success rate is 100 percent (5/5)
   ```

2. Run an interactive ping when you must specify additional parameters.

   Example

   ```
   Router# ping
   Protocol [ipv4]: ipv4
   Target IP address: 192.0.2.1
   Repeat count [5]: 5
   Datagram size [100]: 36
   Timeout in seconds [2]: 1
   Extended commands? [no]: yes
   Source address or interface: 192.0.2.2
   ```

3. Review the success rate and round-trip timing values.

A successful test confirms reachability between the selected source and destination addresses.

Check connectivity to multiple destinations

Checks IPv4 reachability for multiple CLI-supplied destinations in one batch and reports the success or failure of the ping test for each address.

Use bulk ping when you need to test several IPv4 destinations.

Before you begin

Prepare the IPv4 destination addresses. The bulk option supports IPv4 destinations only.

Procedure

1. Start a bulk IPv4 ping that accepts destinations from the CLI.

   Example

   ```
   Router# ping bulk ipv4 input cli batch
   ```

2. Enter one destination per line, and then press Ctrl-D.

Example

```
1: 192.0.2.1
2: 198.51.100.1
3: 203.0.113.1
```

3. Respond to the prompts for each destination and review each success rate.

Example

```
Starting pings...
Target IP address: 192.0.2.1
Repeat count [5]: 5
Datagram size [100]: 1
% A decimal number between 36 and 18024.
Datagram size [100]: 1
% A decimal number between 36 and 18024.
Datagram size [100]: 1000
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 10
Extended commands? [no]: no
Sweep range of sizes? [no]:
Type escape sequence to abort.
Sending 5, 1000-byte ICMP Echos to 192.0.2.1, vrf is default, timeout is 1
seconds:
!!!!!
Success rate is 100 percent (5/5),
Target IP address: 198.51.100.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Interval in milliseconds [10]:
Extended commands? [no]:
Sweep range of sizes? [no]:
Sending 5, 100-byte ICMP Echos to 192.0.2.1, vrf is default, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5),
Target IP address:  203.0.113.1
Repeat count [5]: 4
Datagram size [100]: 100
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 10
Extended commands? [no]: no
Sweep range of sizes? [no]: no
Sending 4, 100-byte ICMP Echos to 192.0.2.1, vrf is default, timeout is 1
seconds:
!!!!!
Success rate is 100 percent (4/4),
```

The router runs the configured ping test for every supplied IPv4 destination.

Traceroute utilities

Explains traceroute utilities, how TTL values and ICMP responses reveal Layer 3 hops, and the procedure for tracing paths with basic or interactive command parameters.

A traceroute utility is a network diagnostic tool that

- identifies the path that packets take toward a remote destination
- displays the responding Layer 3 devices in hop order, and
- helps locate the point at which routing stops.

A basic traceroute uses default probe, timeout, TTL, port, and display settings.

An interactive traceroute lets you select the source address, timeout, probe count, TTL range, UDP port, and display options.

How traceroute discovers packet paths

Describes how traceroute varies the IP TTL value and interprets ICMP responses to identify Layer 3 hops, locate routing failures, and recognize the destination host.

Traceroute records the source of each Internet Control Message Protocol (ICMP) time-exceeded message to identify the path that a packet takes toward a remote destination.

Summary

The process relies on these fields and messages.

- TTL: Limits how many Layer 3 hops process a datagram.
- ICMP time-exceeded message: Identifies an intermediate hop.
- ICMP port-unreachable message: Indicates that the datagram reached the destination.

Workflow

These stages describe how traceroute discovers a packet path.

1. Traceroute sends a UDP datagram with a TTL value of 1. The first router discards the datagram and returns an ICMP time-exceeded message.
2. Traceroute increases the TTL for each successive set of probes. Each router that reduces the TTL to zero identifies itself in an ICMP response.
3. The destination receives a datagram addressed to an unused high-numbered UDP port and returns an ICMP port-unreachable message.

Result

The output lists the discovered Layer 3 hops in path order.

Trace packet routes

Traces the Layer 3 path to a destination and uses interactive parameters when you must specify the source address, timeout, probe count, TTL range, port, or display format.

Use traceroute to identify the hops toward a destination and the point at which routing stops.

Procedure

1. Trace the route to the destination.

   Example

```
Router# traceroute 198.51.100.1
Type escape sequence to abort.
```

```
Tracing the route to 198.51.100.1
1 192.0.2.1 39 msec * 3 msec
```

2. Run interactive traceroute when you must specify additional parameters.

   Example

```
Router# traceroute
Protocol [ipv4]: ipv4
Target IP address: 198.51.100.1
Source address: 192.0.2.1
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:

Type escape sequence to abort.
Tracing the route to 198.51.100.1
 1  192.0.2.1.1 3 msec  *  3 msec
```

3. Review the hop addresses and response times to locate a routing failure.

The output identifies each responding hop between the source and destination.

Enhanced ICMP diagnostics for traceroute

Explains the interface, MTU, and next-hop details added to traceroute ICMP errors, identifies supported configuration and models, documents the SRv6 restriction, and introduces the enablement task.

Enhanced ICMP diagnostics are traceroute extensions that

- add incoming and outgoing interface names, addresses, and MTU values to ICMP error messages
- identify the next-hop address at each supported hop, and
- improve path analysis when interface addressing or topology makes basic traceroute output insufficient.

The implementation complies with RFC 5837 and shares its permitted-remote-address configuration with the RFC 8335 ICMP probe utility.

Table 32: Feature History Table

Feature Name	Release Information	Feature Description
Enhanced ICMP error messages for traceroute	Release 25.4.1	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: Q100, Q200, P100]) You now get detailed insights into each step of the network path. Building on the existing traceroute utility, this feature adds more diagnostic information to ICMP error messages. At each hop, the router shows not only the IP address, but also the incoming and outgoing interfaces and the next hop IP address. This makes it easier to find and fix network issues. This feature introduces these changes: CLIs: • **icmp ipv4 mpls extended-diagnostics** • **icmp ipv6 mpls extended-diagnostics** • **icmp ipv4 vrf extended-diagnostics permitted-remote-address** • **icmp ipv6 vrf extended-diagnostics permitted-remote-address** • The **extended** keyword is added to the **traceroute** command. YANG Data Model: • `Cisco-IOS-XR-um-ip-icmp-cfg` • `Cisco-IOS-XR-traceroute-act` (see GitHub, YANG Data Models Navigator)

Restriction for enhanced ICMP diagnostics for traceroute

SRv6 does not support enhanced ICMP messages for traceroute.

Benefits of enhanced ICMP diagnostics for traceroute

Enhanced ICMP error messages for traceroute provide several benefits.

- These messages improve visibility into network paths, which allows network operators to diagnose issues more effectively.
- Enhanced ICMP error messages enable identification of specific interfaces and their characteristics at each hop along the network path.
- They help to solve limitations encountered in complex environments, such as those involving unnumbered interfaces or networks that use only loopback IPv4 addresses.
- These messages offer a clearer understanding of how packets traverse the network, supporting more accurate network analysis and troubleshooting.

Enable enhanced ICMP diagnostics for traceroute

Use this procedure to provide extended diagnostic information in ICMP error messages, improving the effectiveness of traceroute for network troubleshooting.

By default, extended ICMP diagnostics are disabled. You must configure which remote addresses or prefixes are permitted to receive enhanced diagnostics. Optionally, enable diagnostics for MPLS traffic or use a wildcard to cover all destinations. This configuration is shared with the ICMP PROBE Utility (RFC 8335). For information on how to configure ICMP PROBE utility, see ICMP probe utilities on page 134. The implementation of enhanced ICMP error messages for traceroute complies with RFC 5837.

Before you begin

- Identify the IP address prefixes (IPv4 or IPv6) for which you want to enable extended diagnostics (up to 1024 prefixes are supported).
- If your network uses MPLS, determine whether extended diagnostics for MPLS traffic should also be enabled.

Procedure

1. Enable extended ICMP diagnostics for the desired VRF and IP address or prefix.

   Example

   ```
   Router# configure
   Router(config)# icmp ipv4 vrf default extended-diagnostics
   permitted-remote-address 192.168.0.0/16
   Router(config-icmp-vrf-diag)# commit
   ```

2. Optional: Enable extended diagnostics for MPLS traffic.

   Example

   ```
   Router# configure
   Router(config)# icmp ipv4 mpls
   Router(config-icmp-mpls)# extended-diagnostics
   Router(config-icmp-mpls-diag)# commit
   ```

   When MPLS support is enabled, incoming MPLS packets are checked against the list of permitted remote prefixes for the default VRF.

   The same configuration can also be repeated in IPv6.

3. Repeat step 1 and 2 on each intermediate router along the traceroute path.

   This configuration should be applied to the routers that are expected to generate the enhanced ICMP error messages.

4. Verify the configuration.

   Example

```
Router# show running-config icmp ipv4 vrf extended-diagnostics
```

5. Display enhanced diagnostic information, if configured and available.

   Example

```
Router# traceroute ipv4 192.168.1.10 source Loopback0 extended
ype escape sequence to abort.
Tracing the route to 192.168.1.10

 1  192.168.1.1  16 msec  15 msec  14 msec
     incoming GigabitEthernet0/0/0/1:
       Address: 192.168.1.1
       MTU: 1500
     outgoing GigabitEthernet0/0/0/2:
       Address: 192.168.1.2
       MTU: 1500
     next hop:
       Address: 192.168.1.10

 2  192.168.1.10  21 msec
     incoming GigabitEthernet0/0/0/2:
       Address: 192.168.1.10
       MTU: 1500
```

The routers include extended diagnostic information in ICMP error messages for specified remote address prefixes. Enhanced diagnostics are displayed in traceroute output for supported destinations.

What to do next

Review traceroute output to confirm that diagnostic information is present. Adjust permitted prefixes or enable MPLS diagnostics on additional routers as needed.

ICMP probe utilities

Explains how RFC 8335 probes report local interface or remote-neighbor status through a proxy, share permitted prefixes, expose asymmetric failures, and support configured IPv4, IPv6, local, and remote tests.

An ICMP probe utility is a diagnostic tool that

- queries the operational status of local interfaces or directly connected remote neighbors
- reports interface activity and IPv4 or IPv6 reachability through a proxy, and
- supports troubleshooting when direct bidirectional connectivity is unavailable.

The utility supports same-family and cross-address-family probes and can expose partial-path, asymmetric-routing, and interface-specific problems that a traditional ping might not identify.

Table 33: Feature History Table

Feature Name	Release Information	Feature Description
ICMP probe utility	Release 25.4.1	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: Q100, Q200, P100]) This feature introduces a new network diagnostic tool called `probe`, similar to `ping` but with enhanced capabilities. It allows users to query the operational status of local interfaces or directly connected remote neighbors, providing details like probe can report whether the interface is active, and whether IPv4 and IPv6 are running. probe operates through a proxy device, making it suitable for troubleshooting scenarios where direct bidirectional connectivity to the probed interface is not available. This feature introduces these changes: CLI: • **probe**

ICMP probe utilities expand upon the traditional ping diagnostic tool by delivering enhanced information about network interfaces and neighbors. First introduced in Release 25.4.1, these utilities allow network administrators to diagnose connectivity and status even in complex environments where direct access may not be possible.

Benefits of ICMP probe utilities

Describes the benefits of the ICMP probe utility, including IPv4 and IPv6 testing, cross-address-family probes, detailed interface and neighbor visibility, improved detection of asymmetric or partial-path issues, and flexible source and destination selection for troubleshooting.

ICMP probe utilities provide these diagnostic benefits.

- The utilities support IPv4 and IPv6 operations, including cross-address-family tests that use an IPv4 probe to assess an IPv6 destination.
- The utilities report the state and reachability of each tested interface or neighbor.
- The utilities identify asymmetric-routing, partial-path, and interface-specific problems that a traditional ping might not detect.
- The utilities summarize successful and failed probe attempts so that you can isolate connectivity problems quickly.
- The utilities let you specify source and destination addresses for complex or multihomed networks.

Configure the ICMP probe utility

Configures the permitted reply prefixes for RFC 8335 probes and runs local-interface or remote-neighbor tests to report activity, reachability, and response details.

Configure ICMP probes to diagnose interface or directly connected neighbor reachability.

The probe feature is disabled by default and reuses the RFC 5837 permitted-remote-address configuration. Probe replies are sent only when the source matches that list.

Procedure

1. Define a permitted remote prefix for the required VRF.

   Example

   ```
   Router# configure
   Router(config)# icmp ipv4 vrf default extended-diagnostics
   permitted-remote-address 192.168.1.0/24
   Router(config-icmp-vrf-diag)# commit
   ```

   You can configure up to 1,024 IPv4 and IPv6 prefixes.

2. Run a local probe for an interface on the proxy device.

   Example

   ```
   Router# probe 192.168.1.10 source Loopback0 by-name BVI1
   ```

3. Run a remote probe to test a directly connected neighbor.

   Example

   ```
   Router# probe 2001:db8::1 source GigabitEthernet0/0/0/0 by-remote-address
   fe80::1
   ```

The device reports interface activity or neighbor reachability for each permitted probe.

Domain name services

Explains how Cisco IOS XR resolves hostnames and addresses through static and dynamic cache entries, configured domain names, and DNS servers, and provides a task for configuring and verifying the resolver.

Domain name services are BSD-compatible resolver functions that

- maintain static and dynamic hostname-to-address cache entries
- complete unqualified hostnames with a default domain or domain list, and
- query configured DNS servers when the local cache does not contain a requested mapping.

Applications such as Telnet and commands such as ping and traceroute request hostname or address resolution through a remote procedure call.

The domain service checks the local cache first, which reduces the time required to resolve previously stored mappings. If no entry exists, the domain service sends a DNS query to a name server and stores the response as a dynamic entry.

DNS name servers maintain distributed hostname-to-address mappings. You can configure one or more name servers for the resolver.

Cache and domain settings

The resolver uses these entry and configuration types.

- Static entry: Created with a **domain ipv4 host** or **domain ipv6 host** command.
- Dynamic entry: Received from a configured name server.
- Name server: Configured with the **domain name-server** command. You can configure one or more name servers.
- Default domain: Configured with the **domain name** command. The resolver appends this domain to an unqualified hostname before adding the completed hostname to the host table.
- Domain list: Configured with the **domain list** command. The list supplies candidate domains for an unqualified hostname.

Configure domain name services

Configures static host mappings, a default domain, and primary and secondary name servers, and then verifies the resolver settings and cached host entries.

Configure the router to resolve hostnames and IP addresses for applications and diagnostic commands.

DNS-based hostname-to-address translation is enabled by default. If it was disabled, use **no domain lookup disable** to enable it.

Procedure

1. Define static IPv4 host mappings.

   Example

   ```
   Router# configure
   Router(config)# domain ipv4 host host1 192.168.7.18
   Router(config)# domain ipv4 host host2 10.2.0.2 192.168.7.33
   Router(config)# commit
   ```

2. Define the default domain name.

   Example

   ```
   Router(config)# domain name sampledomain.com
   Router(config)# commit
   ```

3. Configure the primary and secondary name server addresses.

   Example

   ```
   Router(config)# domain name-server 192.168.1.111
   Router(config)# domain name-server 192.168.1.2
   Router(config)# commit
   ```

4. Verify the domain settings and host cache.

Example

```
Router# show hosts
Default domain is sampledomain.com
Name/address lookup uses domain service
Name servers: 192.168.1.111, 192.168.1.2

Host                      Flags      Age(hr) Type Address(es)
host2                   (perm, OK)    0       IP  10.2.0.2
                                                  192.168.7.33
host1                   (perm, OK)    0       IP  192.168.7.18
```

The router uses the configured cache, domain name, and name servers for name and address resolution.

File transfer services

Explains how FTP, TFTP, rcp, and SCP locate and transfer files, and provides tasks for configuring clients, operating a TFTP server, and copying individual files securely.

File transfer services are resource managers and file-system interfaces that

- use URLs to identify files on routers or remote servers
- move configuration files, software images, core dumps, and other files between systems, and
- provide different authentication, security, and server behaviors through FTP, TFTP, rcp, and SCP.

Protocol characteristics

This table compares the file transfer services.

Table 34: File transfer protocols

Protocol	Primary characteristic	Use case
FTP	TCP/IP file transfer defined in RFC 959	Transfers files and supports configurable client behavior.
TFTP	Simplified file transfer, usually without client authentication	Transfers files and can serve files from router storage.
rcp	Remote copy client	Can save a core dump to a remote server.
SCP	Authenticated transfer over SSHv2	Securely transfers one file between a local and a remote device.

FTP connections

Describes FTP file transfers and the client settings for passive mode, an anonymous-user password, and the source interface used for network connections.

An FTP connection is a TCP/IP file transfer session that

- moves files between network nodes
- can use passive client behavior and an anonymous-user password, and
- can use a configured router interface as its source.

FTP is defined in RFC 959.

Configure FTP client connections

Configures passive FTP behavior, the anonymous-user password, and the router interface that supplies the source address, and then verifies the running configuration.

Set the FTP client characteristics used for file transfers.

Procedure

1. Configure the FTP client settings and commit the configuration.

   Example

```
Router# configure
Router(config)# ftp client passive
Router(config)# ftp client anonymous-password xxxx
Router(config)# ftp client source-interface HundredGigE 0/0/0/0
Router(config)# commit
```

2. Verify each FTP client setting.

   Example

```
Router# show running-config ftp client passive
Router# show running-config ftp client anonymous-password xxxx
Router# show running-config ftp client source-interface HundredGigE 0/0/0/0
```

TFTP connections

Describes simplified file transfers that usually omit client authentication and introduces the client source-interface and router-based server configurations used for TFTP connections.

A TFTP connection is a simplified file transfer session that

- moves files between networked systems
- usually operates without a username and password, and
- can use either a router client or a router-based server.

Configure TFTP client connections

Configures the router interface that supplies the source address for TFTP client connections and verifies the setting and registered TFTP service.

Select a consistent source interface for TFTP client traffic.

Procedure

1. Configure and commit the TFTP client source interface.

   Example

```
Router# configure
Router(config)# tftp client source-interface HundredGigE 0/0/0/0
Router(config)# commit
```

2. Verify the client configuration and service registration.

   Example

```
Router# show running-config tftp client source-interface HundredGigE 0/0/0/0
tftp client source-interface HundredGigE 0/0/0/0
Router# show cinetd services
Vrf Name Family  Service  Proto Port    ACL max_cnt  curr_cnt  wait   Program
 Client Option
default  v4      tftp     udp    69      unlimited  0       wait   tftpd
sysdb disk0:
default  v4      telnet   tcp    23      10         0       nowait telnetd
sysdb
```

TFTP servers

Explains how a router-based TFTP server provides files such as system images and configurations to reachable TFTP clients, and describes its connectivity and write-request requirements.

A router-based TFTP server is a file service that

- responds to requests from reachable TFTP client routers
- provides files such as system images or configurations from router storage, and
- reduces the need for a dedicated server on each network segment.

Table 35: Feature History Table

Feature Name	Release Information	Feature Description
TFTP Server support in Router	Release 7.5.4	You can now configure the Cisco 8000 Series Router as a TFTP server to serve requests from client routers. This capability reduces costs and time delays in your network by eliminating the redundancy of having a machine that acts only as a server on every network segment. This feature introduces the following commands: • **tftp server** • **show cinetd services**

Configure a router as a TFTP server

Configures a router to serve TFTP read and write requests from a selected home directory and verifies both the running configuration and service registration.

Provide files to TFTP clients from a router storage directory.

Before you begin

- Verify reachability between the server and client routers with ping.
- Create any destination file before a client sends a write request.

Procedure

1. Configure the TFTP server home directory and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# tftp ipv4 server homedir disk0:
   Router(config)# commit
   ```

2. Verify the server configuration.

   Example

   ```
   Router# show running-config tftp ipv4 server homedir disk0:
   tftp vrf default ipv4 server homedir disk0:
   ```

3. Verify that Cisco inetd registered the TFTP service.

Example

```
Router# show cinetd services
Vrf Name Family Service     Proto Port ACL  max_cnt  curr_cnt wait  Program
Client Option
default  v4       tftp        udp   69      unlimited 0       wait   tftpd
 sysdb disk0:
default  v4       telnet      tcp   23      10        0       nowait telnetd
 sysdb
```

SCP transfers

Explains SCP security, client and server roles, single-file limitations, source and sink modes, and the procedure for copying and verifying a file over SSHv2.

Secure Copy Protocol (SCP) is a file transfer protocol that

- uses SSHv2 to provide secure and authenticated transfers
- supports both client and server operations on Cisco IOS XR devices, and
- copies one file between a local device and a remote device.

A device acts as an SCP server when it receives an SCP request. The SSH server starts a new SCP server instance for each incoming subsystem request.

A device acts as an SCP client when it sends a file transfer request to another device.

SCP transfer limitations

SCP transfers have these limitations.

- SCP transfers one file at a time.
- SCP cannot copy a file directly between two remote devices.
- The SSH server process must run on a destination that acts as the SCP server.

How SCP transfers files

Describes how SCP uses SSHv2 client and server processes and source or sink modes to transfer one file between a local device and a remote device.

SCP uses an authenticated SSHv2 connection to transfer a file between a local device and a remote device.

Summary

The process uses these roles.

- Client: Starts the SSH connection and requests the file transfer.
- Server: Starts a new SCP instance for each incoming subsystem request.
- Source: Reads the file from its local directory and sends it to the destination.
- Sink: Receives the file at the destination.

Workflow

These stages describe how SCP transfers a file.

1. The client starts an SSHv2 connection to the remote device and requests a file transfer.

2. The SSH server starts an SCP server instance for the incoming request.
3. The device that provides the file operates in source mode, and the receiving device operates in sink mode.
4. The source reads the file from its local directory and sends it through the authenticated connection to the sink.

Result

The file is available on the selected local or remote destination.

Transfer a file with SCP

Copies one file between a local router and a remote device over SSHv2 and verifies that the destination contains the transferred file.

Securely copy an individual file to or from a remote device.

Before you begin

Ensure that the SSH server process runs on a destination that acts as the SCP server.

SCP cannot transfer multiple files or copy a file directly between two remote devices.

Procedure

1. Copy the local file to the remote destination and enter the remote password when prompted.

   Example

   ```
   Router# scp /harddisk:/test123.txt user@192.0.2.10:/remote/test123.txt
   Connecting to 192.0.2.10...
   Password:
   ```

2. Verify that the destination contains the file.

   Example

   ```
   remote-host:/remote> ls -altr test123.txt
   ```

The specified file exists at the remote destination.

Router application services

Explains how the router starts network daemons, accepts Telnet sessions, assigns a stable source to syslog traffic, and controls HTTP client connections and protocol behavior.

Router application services are functions that

- start configured server programs for incoming network requests
- identify outbound management traffic through selected source interfaces, and
- control client connection, security, response, and protocol settings.

Cisco inetd services

Explains the multithreaded Cisco inetd daemon, its configuration-dependent Telnet and TFTP listeners, the Telnet RPM requirement, and how it starts a server program for each request.

Cisco Internet services process daemon, or Cisco inetd, is a system server process that

- starts after the system boots
- listens for configured network services such as Telnet and TFTP, and
- starts the server program associated with an incoming service request.

Cisco inetd is multithreaded. The router configuration determines which services it monitors.

For example, the **tftp server** command causes Cisco inetd to listen for TFTP requests.

Cisco inetd service requirement

Important

Install the Telnet RPM before you use the Telnet service or the **show cinetd services** command.

How Cisco inetd services work

Describes how Cisco inetd starts after system boot, registers configured Telnet or TFTP services, listens for incoming requests, and starts the associated server program.

The system manager starts the multithreaded Cisco inetd process after the system boots.

Summary

The process uses these components.

- Service configuration: Determines which network services Cisco inetd monitors.
- Service listener: Waits for incoming requests to a registered service.
- Server program: Handles a request for the associated service.

Workflow

These stages describe how Cisco inetd handles a service request.

1. A server command, such as **tftp server**, registers a network service.
2. Cisco inetd starts listening for requests to the registered service.
3. A client sends a request to the service.
4. Cisco inetd starts the server program associated with the requested service.

Result

The server program handles the client request.

What's next

Use **show cinetd services** to display registered services and their current connection counts.

Telnet services

Describes inbound Telnet access, the required optional RPM package, and the configurable maximum number of simultaneous users accepted by the router.

A Telnet service is an inbound access function that

- accepts Telnet connections to the router
- uses Cisco inetd to start the Telnet server program, and
- can limit the number of simultaneous users.

Install the Telnet RPM before you enable Telnet or use **show cinetd services** for the Telnet service.

Enable the Telnet server

Enables the IPv4 Telnet server, limits simultaneous access to the configured number of users, and verifies that Cisco inetd registered the service.

Allow inbound Telnet sessions while limiting concurrent connections.

Before you begin

Install the Telnet RPM.

Procedure

1. Enable the IPv4 Telnet server, set the maximum number of users, and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# telnet ipv4 server max-servers 10
   Router(config)# commit
   ```

2. Verify the registered Telnet service and maximum connection count.

   Example

   ```
   Router# show cinetd services
   ```

The router accepts up to 10 simultaneous IPv4 Telnet sessions.

Syslog source interfaces

Describes how a configured logging source interface gives syslog traffic from a router and VRF a consistent source identity at the remote server.

A syslog source interface is a logging setting that

- selects the interface address used for remote syslog traffic
- associates traffic from a VRF with one router identity, and
- helps the remote server identify messages from the same device.

Configure a syslog source interface

Configures a loopback interface as the source of remote syslog traffic and verifies the source-interface setting in the running logging configuration.

Give remote syslog traffic a consistent router source address.

Procedure

1. Configure and commit the logging source interface.

   Example

   ```
   Router# configure
   Router(config)# logging source-interface Loopback2
   Router(config)# commit
   ```

2. Verify the logging configuration.

   Example

   ```
   Router# show running-config logging
   logging source-interface Loopback2
   ```

Remote syslog traffic uses the address assigned to Loopback2.

HTTP client applications

Describes the default HTTP client used to retrieve files and the configurable connection, response, certificate, source-interface, SSL, TCP window-scale, HTTP version, and VRF settings.

An HTTP client application is a router function that

- retrieves files from an HTTP server
- supports connection, response, identity, security, and source settings, and
- selects the HTTP version and VRF for requests.

The HTTP client application is available by default and is configured in XR configuration mode.

Configuration options

This table describes the HTTP client settings.

Table 36: HTTP client settings

Setting	Function
connection	Sets retry or timeout behavior.
response	Sets how long the client waits for a server response.
secure-verify-host	Verifies that the peer certificate identifies the server. Use **http client secure-verify-host disable** to disable this check.

Setting	Function
secure-verify-peer	Verifies the authenticity of the peer certificate.
source-interface	Selects the source interface for outgoing HTTP connections and supports an IPv4 address, an IPv6 address, or both.
ssl	Selects the SSL configuration for HTTPS requests.
tcp-window-scale	Sets the TCP window-scale factor for high-latency links.
version	Selects HTTP version to be used in HTTP request • 1.0: HTTP1.0 will be used for all HTTP requests. • 1.1: HTTP1.1 will be used for all HTTP requests. • default libcurl: will use HTTP version automatically.
vrf	Specifies the VRF name for HTTP requests.

Configure the HTTP client

Configures the TCP window-scale factor and HTTP protocol version used for outgoing requests and shows how to display the available HTTP client settings.

Adjust HTTP client behavior for the network and server requirements.

The HTTP client is available by default.

Procedure

1. Display the available HTTP client settings.

   Example

   ```
   Router# configure
   Router(config)# http client ?
   ```

2. Configure a TCP window-scale factor for high-latency links when required.

   Example

   ```
   Router(config)# http client tcp-window-scale 8
   ```

3. Configure the HTTP version when the server requires a specific protocol version.

   Example

   ```
   Router(config)# http client version 1.0
   Router(config)# commit
   ```

Subsequent HTTP requests use the configured client settings.

7 Introduction to Access Control Lists

Topics:

This chapter introduces access control lists (ACLs), covering how sequence numbering works, how to apply ACLs, and the tasks to configure IPv4, IPv6, and extended ACLs on the Cisco 8000 Series Router.

Access Control Lists

This topic describes access control lists (ACLs) on the Cisco 8000 Series Router, which perform packet filtering to control which packets move through the network and where.

The Access Control List (ACL) is a security mechanism that

- controls packet movement through the network
- restricts user and device access, and
- defines network traffic profiles through collections of entries.

An ACL consists of one or more access control entries (ACE) that collectively define the network traffic profile. Access control entries (ACE) are entries in an ACL that describe the access rights related to a particular security identifier or user. This profile can then be referenced by Cisco IOS XR software features such as traffic filtering, route filtering, QoS classification, and access control. There are two types of ACLs:

- Standard ACLs—Verify only the source IP address of the packets. Traffic is controlled by the comparison of the address or prefix configured in the ACL, with the source address found in the packet.
- Extended ACLs—Verify more than just the source address of the packets. Attributes such as destination address, specific IP protocols, User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) port numbers, Differentiated Services Code Point (DSCP), and so on are validated. Traffic is controlled by a comparison of the attributes stated in the ACL with those in the incoming or outgoing packets.

Cisco IOS XR does not differentiate between standard and extended access lists. Standard access list support is provided for backward compatibility.

How an IP ACL works

This topic describes how an IP ACL processes packets on the Cisco 8000 Series Router.

An ACL (Access Control List) is a sequential list consisting of permit and deny statements that apply to IP addresses and possibly upper-layer IP protocols. The ACL has a name by which it is referenced. Many software commands accept an access list as part of their syntax.

An ACL can be configured and named, but it is not in effect until the ACL is referenced by a command that accepts an access list. Multiple commands can reference the same access list. An ACL can control traffic arriving at the router or leaving the router, but not traffic originating at the router.

Summary

Source address and destination addresses are two of the most typical fields in an IP packet on which to base an access list. The key fields on which an ACL filters packets are:

- Source address: Specify source addresses to control packets from certain networking devices or hosts.
- Destination address: Specify destination addresses to control packets being sent to certain networking devices or hosts.
- Transport layer information: You can also filter packets on the basis of transport layer information, such as whether the packet is a TCP, UDP, Internet Control Message Protocol (ICMP), or Internet Group Management Protocol (IGMP) packet.

This image illustrates the workflow of an ACL.

Workflow

ACL process and rules

This topic lists the process and rules to use when you configure an ACL on the Cisco 8000 Series Router.

Use the these process and rules when configuring an IP ACL:

- The software tests the source or destination address or the protocol of each packet being filtered against the conditions in the access list, one condition (permit or deny statement) at a time.
- If a packet does not match an access list statement, the packet is then tested against the next statement in the list.
- If a packet and an access list statement match, the remaining statements in the list are skipped and the packet is permitted or denied as specified in the matched statement. The first entry that the packet matches determines whether the software permits or denies the packet. That is, after the first match, no subsequent entries are considered.
- If the access list denies the address or protocol, the software discards the packet and returns an ICMP Host Unreachable message. ICMP is configurable in the Cisco IOS XR software.

  ICMP type and code such as ECHO, ECHO-REPLY, MASK-REPLY, MASK-REQUEST, and so on are supported.
- If no conditions match, the software drops the packet because each access list ends with an unwritten or implicit deny statement. That is, if the packet has not been permitted or denied by the time it was tested against each statement, it is denied.
- The access list should contain at least one permit statement or else all packets are denied.
- Because the software stops testing conditions after the first match, the order of the conditions is critical. The same permit or deny statements specified in a different order could result in a packet being passed under one circumstance and denied in another circumstance.
- Only one access list per interface, per protocol, per direction is allowed.
- Inbound access lists process packets arriving at the router. Incoming packets are processed before being routed to an outbound interface. An inbound access list is efficient because it saves the overhead of routing lookups if the packet is to be discarded because it is denied by the filtering tests. If the packet is permitted by the tests, it is then processed for routing. For inbound lists, permit means continue to process the packet after receiving it on an inbound interface; **deny** means discard the packet.
- Outbound access lists process packets before they leave the router. Incoming packets are routed to the outbound interface and then processed through the outbound access list. For outbound lists, permit means send it to the output buffer; deny means discard the packet.
- An access list can not be removed if that access list is being applied by an access group in use. To remove an access list, remove the access group that is referencing the access list and then remove the access list.
- Before removing an interface, which is configured with an ACL that denies certain traffic, you must remove the ACL and commit your configuration. If this is not done, then some packets are leaked through the interface as soon as the **no interface** <**interface-name**> command is configured and committed.
- An access list must exist before you can use the **ipv4** | **ipv6 access-group** command.

Helpful hints for creating IP access lists

This topic lists helpful hints to consider when you create an ACL on the Cisco 8000 Series Router.

Consider these tips when creating an ACL:

- Create the ACL before applying it to an interface.
- Organize your ACL so that more specific references in a network or subnet appear before more general ones.
- To make the purpose of individual statements more easily understood at a glance, you can write a helpful remark before or after any statement.

Wildcard mask ACL address filtering

This topic describes how ACL address filtering uses wildcard masks and implicit wildcard masks on the Cisco 8000 Series Router.

The wildcard mask is a filtering mechanism that

- indicates whether the software checks or ignores corresponding IP address bits when comparing the address bits in an access-list entry to a packet being submitted to the access list
- selects single or multiple IP addresses for permit or deny tests, and
- enables noncontiguous bit manipulation for granular control.

The system processes wildcard masks according to these rules:

- A wildcard mask bit 0 means *check* the corresponding bit value.
- A wildcard mask bit 1 means *ignore* that corresponding bit value.
- Wildcard masking for IP address bits uses the number 1 and the number 0 to specify how the software treats the corresponding IP address bits. A wildcard mask is sometimes referred to as an *inverted mask*, because a 1 and 0 mean the opposite of what they mean in a subnet (network) mask.
- You do not have to supply a wildcard mask with a source or destination address in an access list statement. If you use the **host** keyword, the software assumes a wildcard mask of 0.0.0.0.

  Unlike subnet masks, which require contiguous bits indicating network and subnet to be ones, wildcard masks allow noncontiguous bits in the mask.
- You can utilize Classless Inter-Domain Routing (CIDR) format as an alternative to explicit wildcard bits. For example, the IPv4 address 1.2.3.4 0.255.255.255 corresponds to 1.2.3.4/8 and for IPv6 address 2001:db8:abcd:0012:0000:0000:0000:0000 corresponds to 2001:db8:abcd:0012::0/64.

Configuration guidelines for ACLs

This topic lists the guidelines to follow when you configure ACLs on the Cisco 8000 Series Router.

These are the guidelines for configuring ACLs:

- If the Ternary content-addressable memory (TCAM) utilization is high and large ACLs are modified, then an error may occur. During such instances, remove the ACL from the interface and reconfigure the ACL. Later, reapply the ACL to the interface.
- ACL commit failures caused by TCAM Out of Resources (OOR) can lead to inconsistent behavior, including partial TCAM programming on other slices and misleading error messages for subsequent items in batch commits. To prevent this, apply ACL configurations one interface at a time with separate commits. After an OOR failure, verify TCAM usage and consider a line card reload to ensure resource consistency.
- Modifying an ACL when it is attached to the interface is supported.
- You can configure an ACL name with a maximum of 64 characters.
- You can configure an ACL name to comprise of only letters and numbers.
- The ACL ID scale per protocol for IPv4 and IPv6 traffic in each direction:
  - The ACL scale for the Cisco 8200 series router is 15 ACL IDs per slice pair, 45 ACL IDs per NPU, 45 ACL IDs per line card, and 45 ACL IDs per router.
  - The ACL scale for the Cisco 8812 series routers with 8800-LC-48H line cards is 15 ACL IDs per slice pair, 30 ACL IDs per NPU, 60 ACL IDs per line card, and 720 ACL IDs per router.

    - The ACL scale for the Cisco 8812 series routers with 8800-LC-36FH-M line cards is 15 ACL IDs per slice pair, 30 ACL IDs per NPU, 120 ACL IDs per line card, and 1440 ACL IDs per router.

- An ACL-dependent feature refers to a capability in network systems that relies on Access Control Lists (ACLs) for its operation. These features include both global such as Lawful Intercept (LI), BGP Flow Specification (BGPFS) and 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.

Restrictions for configuring ACLs

This topic lists the restrictions for configuring access lists on the Cisco 8000 Series Router.

These restrictions apply for ACLs:

- IPv4 and IPv6 ACLs are not supported for loopback and interflex interfaces.
- Filtering of Multiprotocol Label Switching (MPLS) packets through interface ACL is not supported.

Comments in ACLs

This topic describes how to include comments (remarks) about entries in a named IP ACLs on the Cisco 8000 Series Router.

You can include comments (remarks) about entries in any named IP ACL using the remark access list configuration command. The remarks make the ACL easier for the network administrator to understand and scan. Each remark line is limited to 255 characters.

The remark can go before or after a **permit** or **deny** statement. You should be consistent about where you put the remark so it is clear which remark describes which **permit** or **deny** statement. For example, it would be confusing to have some remarks before the associated **permit** or **deny** statements and some remarks after the associated statements. Remarks can be sequenced.

Remember to apply the access list to an interface or terminal line after the access list is created.

Purpose of IP Access Control Lists

This topic lists the purposes for which you can use IP Access Control Lists (ACLs) on the Cisco 8000 Series Router.

You can use IP ACLs to:

- Filter incoming or outgoing packets on an interface.
- Filter packets for mirroring.
- Redirect traffic as required.
- Restrict the contents of routing updates.
- Limit debug output based on an address or protocol.
- Control vty access.
- Identify or classify traffic for advanced features, such as congestion avoidance, congestion management, and priority and custom queueing.

IP access list entry sequence numbering

This topic describes the IP access list entry sequence numbering feature on the Cisco 8000 Series Router, which allows you to add sequence numbers to access-list entries and resequence them.

The IP Access List Entry Sequence Number is an ACL configuration feature that

- simplifies access list modifications
- enables precise positioning of entries within a list, and
- eliminates the need to remove and re-enter existing entries during updates.

The IP access list entry sequence numbering feature allows users to add sequence numbers to access-list entries and resequence them. When you add a new entry, you choose the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

Table 37: Feature History Table

Feature Name	Release Information	Feature Description
IP Access List Entry Sequence Numbering	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on: • 88-LC1-48Y8H-EM
IP Access List Entry Sequence Numbering	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) The IP Access List Entry Sequence Numbering feature makes managing access lists easier and less error-prone by allowing you to assign sequence numbers to each entry. You can insert, reorder, or resequence entries without removing and re-entering the entire list. By specifying a sequence number, new entries are placed exactly where needed, and the system automatically numbers entries when not specified. Sequence numbers are synchronized across the route processor and line cards, and this feature works with both standard and extended named IP access lists. *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8011-4G24Y4H-I

Sequence numbering behavior

This topic describes the sequence numbering behavior for IP access list entries on the Cisco 8000 Series Router.

These are the details of the sequence numbering behavior:

- If entries with no sequence numbers are applied, the first entry is assigned a sequence number of 10, and successive entries are incremented by 10. The maximum configurable sequence number is 2147483643 for IPv4 and IPv6 entries. For other entries, the maximum configurable sequence number is 2147483646. If the generated sequence number exceeds this maximum number, the following message displays:

```
Exceeded maximum sequence number.
```

- If you provide an entry without a sequence number, it is assigned a sequence number that is 10 greater than the last sequence number in that access list and is placed at the end of the list.
- ACL entries can be added without affecting traffic flow and hardware performance.
- If a new access list is entered from global configuration mode, then sequence numbers for that access list are generated automatically.
- Distributed support is provided so that the sequence numbers of entries in the route processor (RP) and line card (LC) are synchronized at all times.
- This feature works with named standard and extended IP access lists. Because the name of an access list can be designated as a number, numbers are acceptable.

Example: Add entries with sequence numbers

This topic provides an example of adding an entry with a sequence number to an IPv4 access list on the Cisco 8000 Series Router.

In the following example, a new entry is added to IPv4 access list acl_5.

```
ipv4 access-list acl_5
 2 permit ipv4 host 192.0.2.1 any
 5 permit ipv4 host 198.51.100.44 any
 10 permit ipv4 host 198.51.100.1 any
 20 permit ipv4 host 198.51.100.2 any
configure
ipv4 access-list acl_5
 15 permit 203.0.113.1 255.255.255.0
end
ipv4 access-list acl_5
 2 permit ipv4 host 192.0.2.1 any
 5 permit ipv4 host 198.51.100.44 any
 10 permit ipv4 host 198.51.100.1 any
 15 permit ipv4 203.0.113.1 255.255.255.0 any
 20 permit ipv4 host 198.51.100.2 any
```

Example: Add entries without sequence numbers

This topic provides an example of adding an entry without a sequence number to an IPv4 access list on the Cisco 8000 Series Router.

The following example shows how an entry with no specified sequence number is added to the end of an access list. When an entry is added without a sequence number, it is automatically given a sequence number that puts it at the end

of the access list. Because the default increment is 10, the entry will have a sequence number 10 higher than the last entry in the existing access list.

```
configure
ipv4 access-list acl_10
permit 192.0.2.1 255.255.255.0
permit 198.51.100.1 255.255.255.0
permit 203.0.113.1 255.255.255.0
end

ipv4 access-list acl_10
 10 permit ip  192.0.2.1 255.255.255.0 any
 20 permit ip  198.51.100.1 255.255.255.0 any
 30 permit ip  203.0.113.1 255.255.255.0 any

configure
ipv4 access-list acl_10
permit  203.0.113.5 255.255.255.0
end

ipv4 access-list acl_10
 10 permit ip  192.0.2.1 255.255.255.0 any
 20 permit ip  198.51.100.1 255.255.255.0 any
 30 permit ip  203.0.113.1 255.255.255.0 any
 40 permit ip  203.0.113.5 255.255.255.0 any
```

How applying ACLs work

This topic describes how ACLs are applied on the Cisco 8000 Series Router, and provides guidelines for referencing ACLs on terminal lines and network interfaces.

After you create an ACL, you must reference the ACL to make it work. ACLs can be applied on either outbound or inbound interfaces. This section describes guidelines on how to accomplish this task for both terminal lines and network interfaces.

Set identical restrictions on all the virtual terminal lines, because a user can attempt to connect to any of them.

Summary

These are the ways in which an ACL can be applied on an interface:

- Inbound ACL: Applies to packets arriving at the router, before they are routed.
- Outbound ACL: Applies to packets after they have been routed to a controlled interface, before they leave the router.

Workflow

These stages describe how Cisco IOS XR software processes packets against an applied ACL:

1. For inbound ACLs, after receiving a packet, Cisco IOS XR software checks the source address of the packet against the ACL. If the ACL permits the address, the software continues to process the packet. If the ACL rejects the address, the software discards the packet and returns an ICMP host unreachable message. The ICMP message is configurable.
2. For outbound ACLs, after receiving and routing a packet to a controlled interface, the software checks the source address of the packet against the ACL. If the ACL permits the address, the software sends the packet. If the ACL rejects the address, the software discards the packet and returns an ICMP host unreachable message.
3. When you apply an ACL that has not yet been defined to an interface, the software acts as if the ACL has not been applied to the interface and accepts all packets. Note this behavior if you use undefined ACLs as a means of security in your network.

IPv4 ACLs

This topic introduces IPv4 access control lists (ACLs) on the Cisco 8000 Series Router, which you can apply in the ingress or egress direction on interfaces to filter IPv4 traffic.

You can configure IPv4 access control lists (ACLs) in the ingress or egress direction on interfaces of the Cisco 8000 Series Router. IPv4 ingress and egress ACLs are subject to platform-specific configuration guidelines, restrictions, and per-NPU scale limits.

Configuration guidelines and restrictions for ingress IPv4 ACLs

This topic lists the configuration guidelines and restrictions for IPv4 ingress ACLs on the Cisco 8000 Series Router. These restrictions are subject to change with respect to other platforms.

These restrictions apply to IPv4 ingress ACLs:

- Ingress IPv4 ACLs are supported on all interfaces except management interfaces.
- In fixed systems, the maximum number of ACLs allowed per NPU is 45. In distributed systems, the maximum number of ACLs allowed per NPU is 30.
- Packet Length is not supported.
- From Release 7.8.1 onwards, ACL logging with input interface (using the **log-input** keyword) is supported.
- The maximum number of ingress ACLs for Cisco 8011-4G24Y4H-I is 510.

Configuration guidelines and restrictions for egress IPv4 ACLs

This topic lists the configuration guidelines and restrictions for IPv4 egress ACLs on the Cisco 8000 Series Router.

These restrictions apply to IPv4 egress ACLs:

- ACL is not supported on Management interface on egress direction.
- ACL logging is not supported on egress direction.
- The maximum number of egress ACLs for Cisco 8011-4G24Y4H-I router is 45.

Supported ACLs per NPU

This topic lists the ingress and egress IPv4 ACL scale supported per NPU across ASICs on the Cisco 8000 Series Router.

This table lists the supported ACLs per NPU.

System	ASIC	Ingress ACL per NPU	Egress ACL per NPU
Fixed	Q100	381	45
	Q200	381	45
	K100	510	510
	P100	510 (Release 24.3.1)	510 (Release 24.3.1)
		1020 (Release 24.4.1)	1020 (Release 24.4.1)

System	ASIC	Ingress ACL per NPU	Egress ACL per NPU
Modular	Q100	254	30
	Q200	254	30
	P100	510 (Release 24.3.1)	510 (Release 24.3.1)
		1020 (Release 24.4.1)	1020 (Release 24.4.1)
Centralized	Q200	381	45

Configure an ingress IPv4 ACL

This topic describes the basic configuration of an IPv4 ingress access control list (ACL) on an interface of the Cisco 8000 Series Router.

Use this procedure to configure an ingress IPv4 ACL on a HundredGigE interface.

Procedure

1. Enter global configuration mode. Configure an ingress IPv4 ACL on a HundredGigE interface.

   Example

   ```
   Router# configure terminal
   ```

2. Configure the HundredGigE interface with an IPv4 address and enable it.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/0/0
   Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
   Router(config-if)# no shut
   Router(config-if)# commit
   Router(config-if)# exit
   ```

3. Verify that the interface is up. Confirm the interface status shows **Up** for both **Status** and **Protocol**.

   Example

   ```
   Router(config)# do show ipv4 interface brief
   Thu Jul 11 08:46:51.930 UTC
   Interface                  IP-Address      Status          Protocol Vrf-Name
   HundredGigE 0/0/0/0         192.0.2.1       Up              Up       default
   ```

4. Create an IPv4 ingress access control list (ACL) named `V4-ACL-INGRESS` and add rules.

   Example

   ```
   Router(config)# ipv4 access-list V4-ACL-INGRESS
   Router(config-ipv4-acl)# 10 permit tcp 192.0.2.2 255.255.255.0 any
   Router(config-ipv4-acl)# 20 deny udp any any
   ```

```
Router(config-ipv4-acl)# 30 permit ipv4 192.0.2.64 255.255.255.0 any
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# exit
```

5. Verify the ACL creation. Confirm the ACL `V4-ACL-INGRESS` and its rules are listed.

   Example

```
Router(config)# do show access-lists ipv4

Thu Jul 11 09:01:50.445 UTC
HundredGigE 0/0/0/0 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.0.2.1/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  common access list is not set, access list is V4-ACL-INGRESS
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
```

6. Apply the ingress ACL to the HundredGigE interface.

```
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv4 access-group V4-ACL-INGRESS ingress
Router(config-if)# commit
Router(config-if)# exit
```

7. Verify that the ingress ACL has been successfully applied to the interface.

   Example

```
Router(config)# do show ipv4 interface
```

   You have successfully configured an IPv4 ingress ACL on a HundredGigE interface.

Configure an egress IPv4 ACL

This topic describes the basic configuration of an IPv4 egress access control list (ACL) on an interface of the Cisco 8000 Series Router.

Use this procedure to configure an egress IPv4 ACL on a HundredGigE interface.

Procedure

1. Enter global configuration mode. Configure an egress IPv4 ACL on a HundredGigE interface.

Example

```
Router# configure terminal
```

2. Configure the HundredGigE interface with an IPv4 address and enable it.

   Example

```
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv4 address 198.51.100.1 255.255.255.0
Router(config-if)# no shut
Router(config-if)# commit
Router(config-if)# exit
```

3. Verify that the interface is up. Confirm the interface status shows **Up** for both **Status** and **Protocol**.

   Example

```
Router(config)# do show ipv4 interface brief
Thu Jul 11 08:55:44.824 UTC
Interface                      IP-Address      Status          Protocol
Vrf-Name
HundredGigE 0/0/0/0         192.0.2.1           Up              Up
default
HundredGigE 0/0/0/1         198.51.100.1        Up              Up
default
```

4. Create an IPv4 egress access control list (ACL) named `V4-ACL-EGRESS` and add rules.

   Example

```
Router(config)# ipv4 access-list V4-ACL-EGRESS
Router(config-ipv4-acl)# 10 permit ipv4 203.0.113.1 255.255.255.0 192.0.2.1
0.255.255.255
Router(config-ipv4-acl)# 20 deny ipv4 any any
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# exit
```

5. Verify the ACL creation. Confirm the ACL `V4-ACL-EGRESS` and its rules are listed.

   Example

```
Router(config)# do show access-lists ipv4
Thu Jan 25 10:25:19.896 IST
ipv4 access-list V4-ACL-EGRESS
 10 permit ipv4 203.0.113.1 255.255.255.0 192.0.2.1 255.255.255.0
 20 deny ipv4 any any
```

6. Apply the egress ACL to the HundredGigE interface.

```
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# ipv4 access-group V4-ACL-EGRESS egress
Router(config-if)# commit
Router(config-if)# exit
```

7. Verify that the egress ACL has been successfully applied to the interface.

Example

```
Router(config)# do show ipv4 interface
```

You have successfully configured an IPv4 egress ACL on a HundredGigE interface. For more information on logging messages, see IP access list logging messages on page 256.

IPv6 ACLs

This topic introduces IPv6 access control lists (ACLs) on the Cisco 8000 Series Router, which you can create and apply to interfaces to filter IP version 6 (IPv6) traffic.

You can filter IP version 6 (IPv6) traffic by creating IPv6 access control lists (ACLs) and applying them to interfaces similar to the way that you create and apply IP version 4 (IPv4) named ACLs.

Feature History Table
Table 38: Feature History Table

Feature Name	Release Information	Feature Description
Filter Ingress IPv6 ACL Traffic based on ACL Hop Limit	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Filter Ingress IPv6 ACL Traffic based on ACL Hop Limit	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
Filter Ingress IPv6 ACL Traffic based on ACL Hop Limit	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
Filter Ingress IPv6 ACL Traffic based on ACL Hop Limit	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Filter Ingress IPv6 ACL Traffic based on ACL Hop Limit	Release 7.8.1	We ensure that IPv6 packets have a limited lifetime on your networks, thus reducing the impact of any routing loops and preventing networking failures. To that end, we have added the option to set hop limit rules in the ACLs based on the following hop limit match criteria: • **eq** - equal to • **lt** - less than • **gt** - greater than • **range** - ranges from 1 to 255 To set the hop limit, use the **ttl** keyword in the following commands: • deny (IPv6) • permit (IPv6)

Configuration guidelines and restrictions for ingress IPv6 ACLs

This topic lists the configuration guidelines and restrictions for IPv6 ingress ACLs on the Cisco 8000 Series Router.

These restrictions and guidelines apply while configuring IPv6 ingress ACLs:

- Ingress IPv6 ACLs are supported on all interfaces.
- From Release 7.3.1 onwards, the maximum number of ACLs allowed per router is 126.

  In earlier releases, for the Cisco 8100 and 8200 Series fixed chassis, the maximum number of ACLs allowed per router is 45. For the Cisco 8800 modular chassis, the maximum number of ACLs allowed per router is 30.
- Packet length (using the **pkt-length** keyword) is not supported.
- IPv6 hop limit is supported only on ingress traffic with object-group ACL (compression level 2).

- From Release 7.8.1 onwards, ACL logging with input interface (using the **log-input** keyword) is supported.

Configuration guidelines and restrictions for egress IPv6 ACLs

This topic lists the configuration guidelines and restrictions for IPv6 egress ACLs on the Cisco 8000 Series Router.

The following restrictions and guidelines apply while configuring IPv6 egress ACLs:

- In IPv6 egress ACLs, TCP flag filtering does not function for IPv6 packets with a fragmentation header. As a result, IPv6 packets with both a fragmentation header and a TCP header (ACK+SYN flags) are not appropriately filtered by the ACL rules.

Configure an ingress IPv6 ACL on a HundredGigE interface

This topic describes how to configure an IPv6 ingress access control list (ACL) on a HundredGigE interface of the Cisco 8000 Series Router.

Use this procedure to configure an ingress IPv6 ACL on a HundredGigE interface.

Procedure

1. Configure a HundredGigE interface with an IPv6 address.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/0/0
   Router(config-if)# ipv6 address 2001::1/64
   Router(config-if)# no shut
   Router(config-if)# commit
   Router(config-if)# exit
   ```

2. Verify that the interface is up.

   Example

   ```
   Router(config)# do show ipv6 interface brief
   Thu Jul 11 09:28:43.657 UTC
   HundredGigE 0/0/0/0 [Up/Up]
       fe80::bd:b9ff:fea9:5606
       2001::1
   ```

3. Create an IPv6 ingress access control list (ACL) named `V6-INGRESS-ACL` and add rules.

   Example

   ```
   Router(config)# ipv6 access-list V6-INGRESS-ACL
   Router(config-ipv6-acl)# 10 permit ipv6 any any
   Router(config-ipv6-acl)# 20 deny udp any any
   Router(config-ipv6-acl)# commit
   Router(config-ipv6-acl)# exit
   ```

4. Verify the ingress ACL creation.

Example

```
Router(config)# do show access-lists ipv6
Thu Jul 11 09:41:37.260 UTC
ipv6 access-list V6-INGRESS-ACL
 10 permit ipv6 any any
 20 deny udp any any
```

5. Apply the ingress ACL to the HundredGigE interface.

```
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv6 access-group V6-INGRESS-ACL ingress
Router(config-if)# commit
Router(config-if)# exit
```

6. Verify that the ingress ACL has been successfully applied to the interface.

Example

```
Router(config)# do show ipv6 interface
```

You have successfully configured an IPv6 ingress ACL on a HundredGigE interface.

Configure an egress IPv6 ACL on a HundredGigE interface

This topic describes how to configure an IPv6 egress access control list (ACL) on a HundredGigE interface of the Cisco 8000 Series Router.

Use this procedure to configure an egress IPv6 ACL on a HundredGigE interface.

Procedure

1. Configure a HundredGigE interface with an IPv6 address.

Example

```
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# ipv6 address 2001::1/64
Router(config-if)# no shut
Router(config-if)# commit
Router(config-if)# exit
```

2. Verify that the interface is up.

Example

```
Router(config)# do show ipv6 interface brief
Thu Jul 11 09:47:50.812 UTC
HundredGigE 0/0/0/0 [Up/Up]
    fe80::bd:b9ff:fea9:5606
    1001::1
HundredGigE 0/0/0/1 [Up/Up]
    fe80::23:e9ff:fea8:a44e
    2001::1
```

3. Create an IPv6 egress access control list (ACL) named `V6-EGRESS-ACL` and add rules.

   Example

```
Router(config)# ipv6 access-list V6-EGRESS-ACL
Router(config-ipv6-acl)# 10 permit ipv6 any any
Router(config-ipv6-acl)# 20 deny udp any any
Router(config-ipv6-acl)# commit
Router(config-ipv6-acl)# exit
```

4. Verify the egress ACL creation.

   Example

```
Router(config)# do show access-lists ipv6
Thu Jul 11 09:51:16.687 UTC
ipv6 access-list V6-EGRESS-ACL
 10 permit ipv6 any any
 20 deny udp any any
```

5. Apply the egress ACL to the HundredGigE interface.

```
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# ipv6 access-group V6-EGRESS-ACL egress
Router(config-if)# commit
Router(config-if)# exit
```

6. Verify that the egress ACL has been successfully applied to the interface.

   Example

```
Router(config)# do show ipv6 interface
```

   You have successfully configured an IPv6 egress ACL on a HundredGigE interface.

Configure ingress and egress IPv6 ACLs on bundle interfaces

This topic describes how to configure ingress and egress IPv6 access control lists (ACLs) on a bundle interface of the Cisco 8000 Series Router.

Use this procedure to configure ingress and egress IPv6 ACLs on a bundle interface.

Procedure

1. Configure a bundle interface with an IPv6 address.

   Example

```
Router(config)# interface Bundle-Ether 1
Router(config-if)# ipv6 address 2001::1/64
Router(config-if)# no shut
Router(config-if)# commit
Router(config-if)# exit
```

2. Configure an IPv6 egress ACL for the bundle interface.

   Example

```
Router(config)# ipv6 access-list V6-EGRESS-ACL-bundle interface
Router(config-ipv6-acl)# 10 permit tcp any any range 3000 4000
Router(config-ipv6-acl)# 20 permit ipv6 any any
Router(config-ipv6-acl)# commit
Router(config-ipv6-acl)# exit
```

3. Configure an IPv6 ingress ACL to deny ingress traffic on the bundle interface.

   Example

```
Router(config)# ipv6 access-list V6-DENY-INGRESS-ACL
Router(config-ipv6-acl)# 10 deny ipv6 any any
Router(config-ipv6-acl)# commit
Router(config-ipv6-acl)# exit
```

4. Verify the egress and ingress ACL creation.

   Example

```
Router(config)# do show access-lists ipv6
Thu Jul 11 10:04:35.798 UTC
ipv6 access-list V6-DENY-INGRESS-ACL
 10 deny ipv6 any any
ipv6 access-list V6-EGRESS-ACL-BI
 10 permit tcp any any range 3000 4000
 20 permit ipv6 any any
```

5. Apply the egress and ingress ACLs to the bundle interface.

```
Router(config)# interface Bundle-Ether 1
Router(config-if)# ipv6 access-group V6-EGRESS-ACL-BI egress
Router(config-if)# ipv6 access-group V6-DENY-INGRESS-ACL ingress
Router(config-if)# commit
Router(config-if)# exit
```

6. Verify that the ACLs have been successfully applied to the interface.

   Example

```
Router(config)# do show ipv6 interface
```

   You have successfully configured ingress and egress IPv6 ACLs on a bundle interface.

Ingress IPv6 ACL with hop limit

This topic describes how you can use the hop limit field in IPv6 packets to filter ingress traffic through IPv6 ACLs on the Cisco 8000 Series Router.

Similar to the Time to Live (TTL) field in the IPv4 packet header, the TTL field is referred to as hop limit field in IPv6 packets. To filter an IPv6 packet on your source interface, you can define the following criteria as hop limit value that you set in your ACLs:

- Equal (eq) – permit or deny a packet if the hop limit matches the value as defined in the ACL.
- Less than (lt) – permit or deny a packet if the hop limit matches the value as defined in the ACL.
- Greater than (gt) – permit or deny a packet if the hop limit is greater than as defined in the ACL.
- Range – permit or deny a packet if the hop limit range matches as defined in the ACL.

> **Note**
>
> Hop limit is referred to as **ttl**.

Feature History Table
Table 39: Feature History Table

Feature Name	Release Information	Feature Description
Filter Ingress IPv6 ACL Traffic based on ACL Hop Limit	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
Filter Ingress IPv6 ACL Traffic based on ACL Hop Limit	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
Filter Ingress IPv6 ACL Traffic based on ACL Hop Limit	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
Filter Ingress IPv6 ACL Traffic based on ACL Hop Limit	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Filter Ingress IPv6 ACL Traffic based on ACL Hop Limit	Release 7.8.1	We ensure that IPv6 packets have a limited lifetime on your networks, thus reducing the impact of any routing loops and preventing networking failures. To that end, we have added the option to set hop limit rules in the ACLs based on the following hop limit match criteria: • **eq** - equal to • **lt** - less than • **gt** - greater than • **range** - ranges from 1 to 255 To set the hop limit, use the **ttl** keyword in the following commands: • deny (IPv6) • permit (IPv6)

Configure an ingress IPv6 ACL with hop limit on a bundle interface

This topic describes how to configure an ingress IPv6 access control list (ACL) with hop limit match criteria on a bundle interface of the Cisco 8000 Series Router.

Use this procedure to configure an ingress IPv6 ACL with hop limit on a bundle interface. In the following example, the hop limit criteria, equal (eq) and range (range) is used to filter the traffic. While deny action is used for the 'eq' criteria, permit action is used for 'range' criteria. This means that packets that matches the 'eq' criteria will be denied and the packets that matches the 'range' criteria will be permitted.

Procedure

1. Configure a bundle interface with an IPv6 address.

Example

```
Router(config)# interface Bundle-Ether 1
Router(config-if)# ipv6 address 2001::1/64
Router(config-if)# no shut
Router(config-if)# commit
Router(config-if)# exit
```

2. Create an object group with the list of hosts.

Example

```
Router(config)# object-group network ipv6 network_object_group_bundle_host
Router(config-object-group-ipv6)# host 140:1:2::1
Router(config-object-group-ipv6)# host 150:1:2::1
Router(config-object-group-ipv6)# host 160:1:2::1
Router(config-object-group-ipv6)# commit
Router(config-object-group-ipv6)# exit
```

3. Apply the ACL definition to filter packets based on hop-limit match on IPv6.

Example

```
Router(config)# ipv6 access-list ipv6_ob_network_tcp_bundle_ingress_main
Router(config-if)# 10 deny tcp net-group network_object_group_bundle_host
net-group network_object_group_bundle_host ttl eq 205
Router(config-if)# 20 permit udp net-group network_object_group_bundle_host
net-group network_object_group_bundle_host ttl range 250 255 log-input
!
```

4. Associate the ACL configuration onto the ingress interface to match based on IPv6 hop limit.

```
Router(config)# interface Bundle-Ether500
Router(config-if)# ipv4 address 30.1.0.1 255.255.255.0
Router(config-if)# ipv6 address 30:1::1/96
Router(config-if)# ipv6 access-group ipv6_ob_network_tcp_bundle_ingress_main
 ingress compress level 2
!
```

5. Verify the running configuration of the IPv6 ingress ACL with hop limit.

Example

```
Router:R1# sh access-lists ipv6 ipv6_ob_network_tcp_bundle_ingress_main
hardware ingress detail location 0/0/CPU0
Fri Nov  4 06:39:15.155 UTC

ipv6_ob_network_tcp_bundle_ingress_main Details:
Sequence Number: 10
NPU ID: 0
Number of DPA Entries: 1
ACL ID: 1
ACE Action: DENY
ACE Logging: DISABLED
ABF Action: 0(ABF_NONE)
Hit Packet Count: 0
Source Address: 0:1:0:1::
 Source Address Mask: 0:0:0:0::
```

```
Destination Address: 0:0:0:0::
 Destination Address Mask: 0:0:0:0::
DPA Entry: 1
 Entry Index: 0
 DPA Handle: 0x8E8A10A8
 TTL Match: 0xCD (Mask 0xFF)
Sequence Number: 20
NPU ID: 0
Number of DPA Entries: 2
ACL ID: 1
ACE Action: PERMIT
ACE Logging: ENABLED
ABF Action: 0(ABF_NONE)
Hit Packet Count: 71
Source Address: 0:1:0:1::
 Source Address Mask: 0:0:0:0::
Destination Address: 0:0:0:0::
 Destination Address Mask: 0:0:0:0::
DPA Entry: 1
 Entry Index: 0
 DPA Handle: 0x8E8A1398
 TTL Match: 0xFA (Mask 0xFE)
DPA Entry: 2
 Entry Index: 1
 DPA Handle: 0x8E8A1688
 TTL Match: 0xFC (Mask 0xFC)
<Output truncated>
```

6. Verify the ACL hit count for the ACE and the ACL logging.

Example

```
Router(config)# show running-config | i hw
Fri Nov 4 06:26:27.382 UTC
Building configuration...
hw-module profile stats acl-permit

Router(config)# sh access-lists ipv6 ipv6_ob_network_tcp_bundle_ingress_main
 h i location 0/0/CPU0
Fri Nov  4 06:38:43.664 UTC
ipv6 access-list ipv6_ob_network_tcp_bundle_ingress_main
 10 deny tcp net-group network_object_group_bundle_host net-group
network_object_group_bundle_host ttl eq 205
 20 permit udp net-group network_object_group_bundle_host net-group
network_object_group_bundle_host ttl range 250 255 log-input (71 matches)

Router(config)# show logging | i permit
Mon Nov  7 06:42:30.146 UTC
Router:Nov  7 06:41:59.614 UTC: ipv6_acl_daemon[396]:
%ACL-IPV6_ACL-6-IPACCESSLOGNP : access-list
ipv6_ob_network_tcp_bundle_ingress_main (20) permit 58 150:1:2::1
(Bundle-Ether500)-> 140:1:2::1, 1 packet
```

You have successfully configured an IPv6 ingress ACL on a bundle interface with hop limit. For more information on logging messages, see IP access list logging messages on page 256.

Configure extended ACLs

This topic describes how to configure extended ACLs on the Cisco 8000 Series Router to verify more than just the source address of packets by matching on attributes such as destination address, IP protocols, UDP or TCP port numbers, and DSCP.

Use this procedure to configure an extended ACL. Extended ACLs verify more than just the source address of the packets. Attributes such as destination address, specific IP protocols, UDP or TCP port numbers, DSCP, and so on are validated. Traffic is controlled by a comparison of the attributes stated in the ACL with those in the incoming or outgoing packets.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create an extended IPv4 ACL named `acl_1` and add a remark.

   Example

   ```
   Router(config)# ipv4 access-list acl_1
   Router(config-ipv4-acl)# 10 remark Do not allow user1 to telnet out
   ```

3. Specify the conditions to permit or deny network traffic and commit the configuration.

   Example

   ```
   Router(config-ipv4-acl)# 10 permit 172.16.0.0 0.0.255.255
   Router(config-ipv4-acl)# 20 deny 192.168.34.0 0.0.0.255
   Router(config-ipv4-acl)# commit
   ```

4. Verify the running configuration. Confirm the ACL `acl_1` and its rules are listed.

   Example

   ```
   Router# show running-config
   Mon Jul 29 05:56:14.315 UTC
   Building configuration...
   !! IOS XR Configuration

   !
   ipv4 access-list acl_1
    10 permit ipv4 172.16.0.0 0.0.255.255 any
    20 deny ipv4 192.168.34.0 0.0.0.255 any
   !
   ```

5. Verify that the extended ACL is applied and matching traffic.

   Example

   ```
   Router# show access-lists ipv4 acl_1 hardware ingress location 0/0/CPU0
   Tue Jul 2 08:03:29.495 UTC
   ```

```
ipv4 access-list acl_1
66 deny igmp 30.0.20.0 0.0.0.255 30.0.10.0 0.0.0.255 v3-report (11604 matches)
67 deny igmp host 30.0.20.1 host 30.0.10.1 v2-report
```

You have successfully configured an extended ACL.

Modify ACLs

This topic describes how to modify existing access control lists (ACLs) on the Cisco 8000 Series Router by adding, deleting, and updating access control entries (ACEs).

This section describes a sample configuration for modification of ACLs.

Note

While ACLs allow configuration without sequence numbers, use sequence numbers when adding or deleting ACL entries. In classic ACLs, sequence numbers help minimize backend changes and make it easier for users to track modifications. By specifying sequence numbers, users can add or remove specific entries without affecting other ACEs in hardware. The ACL's appearance remains unchanged except for the intended updates.

Procedure

1. Create an access list and add entries (ACEs) to it.

   Example

```
/* Create an access list */
Router(config)# ipv4 access-list acl_1

/* Add entries (ACEs) to the ACL */
Router(config-ipv4-acl)# 10 permit ip host 10.3.3.3 host 172.16.5.34
Router(config-ipv4-acl)# 20 permit icmp any any
Router(config-ipv4-acl)# 30 permit tcp any host 10.3.3.3
Router(config-ipv4-acl)# end

/* Verify the entries of the ACL */
Router# show access-lists ipv4 acl_1
ipv4 access-list acl_1
 10 permit ip host 10.3.3.3 host 172.16.5.34
 20 permit icmp any any
 30 permit tcp any host 10.3.3.3
```

2. Add a new entry with sequence number "15" to the ACL, and delete the entry with sequence number "30".

   Example

```
Router(config)# ipv4 access-list acl_1
Router(config-ipv4-acl)# 15 permit 10.5.5.5 0.0.0.255
Router(config-ipv4-acl)# no 30
```

```
Router(config-ipv4-acl)# permit 10.4.4.4 0.0.0.255
Router(config-ipv4-acl)# commit

/* Verify the entries of the ACL */
Router# show access-lists ipv4 acl_1
ipv4 access-list acl_1
 10 permit ipv4 host 10.3.3.3 host 172.16.5.34
 15 permit 10.5.5.5 0.0.0.255      /* newly added ACE (with the sequence
number) */
 20 permit icmp any any
 30 permit ipv4 10.4.4.0 0.0.0.255 any    /* newly added ACE (without the
sequence number) */
```

The entry with the sequence number 30, that is, "30 permit tcp any host 10.3.3.3" is deleted from the ACL.

3. Add another entry without a sequence number to the ACL.

Example

```
Router(config)# ipv4 access-list acl_1
Router(config-ipv4-acl)# permit 10.4.4.4 0.0.0.255
Router(config-ipv4-acl)# commit

/* Verify the entries of the ACL */
Router# show access-lists ipv4 acl_1
ipv4 access-list acl_1
 10 permit ipv4 host 10.3.3.3 host 172.16.5.34
 15 permit 10.5.5.5 0.0.0.255
 20 permit icmp any any
 30 permit ipv4 10.4.4.0 0.0.0.255   /* newly added ACE (without the sequence
 number) */
```

When an entry is added without a sequence number, it is automatically given a sequence number that puts it at the end of the access list. Because the default increment is 10, the entry has a sequence number 10 higher than the last entry in the existing access list.

4. Modify an existing entry in the ACL using the sequence number.

Example

```
Router(config)# ipv4 access-list acl_1
Router(config-ipv4-acl)# 10 permit 10.2.1.1 0.0.255.255
Router(config-ipv4-acl)# commit

/* Verify the entries of the ACL */
Router# show access-lists ipv4 acl_1
ipv4 access-list acl_1
 10 permit 10.2.1.1 0.0.255.255      /* Modified ACL entry */
 15 permit 10.5.5.5 0.0.0.255
 20 permit icmp any any
 30 permit ipv4 10.4.4.0 0.0.0.255
```

You have successfully modified ACLs in operation.

TCP flags in ACLs

This topic describes how TCP flags are used in access control lists (ACLs) to permit or deny packets on the Cisco 8000 Series Router.

The Transmission Control Protocol (TCP) flags are indicators used in ACLs that

- influence the flow of data across a TCP connection
- provide information about the connection state, and
- control packet transfer behavior.

The Transmission Control Protocol (TCP) is one of the most widely used protocols for data transmission in networks. The TCP header contains several one-bit boolean fields known as flags used to influence the flow of data across a TCP connection. TCP packets use TCP flags during a packet transfer to indicate connection state or provide additional information about the packet transfer. This list describes the capabilities of ACLs when filtering packets based on TCP flags:

- ACLs allow the creation of Access Control Entries (ACEs) that filter packets based on whether a TCP flag is set or not.
- ACLs enable filtering of packets based on the presence or absence of any single TCP flag or a combination of multiple TCP flags.
- ACLs provide increased flexibility in packet filtering and enhance security by permitting, for example, packets with a SYN flag to ensure verified sources.

The TCP flags include SYN, ACK, FIN, RST, URG, PSH, and EST, each serving to indicate specific connection states or provide additional packet transfer information.

Feature History Table
Table 40: Feature History Table

Feature Name	Release Information	Description
TCP Flags in Egress IPv6 ACLs	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on: • 88-LC1-48Y8H-EM
TCP Flags in Egress IPv6 ACLs	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

Feature Name	Release Information	Description
TCP Flags in Egress IPv6 ACLs	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
TCP Flags in Egress IPv6 ACLs	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
TCP Flags in Egress IPv6 ACLs	Release 7.3.15	You can configure an egress IPv6 ACL that permits or denies packets on the basis of TCP flags set in them. Through this feature, you can restrict and manage traffic streams thereby enhancing network security. The following commands are updated: • deny (IPv6) • permit (IPv6)

TCP flags

This topic lists the TCP flags that can be present in a packet and used in access control list (ACL) entries on the Cisco 8000 Series Router.

This section lists the TCP flags that can be present in a packet.

The following TCP flags can be present in a packet:

- SYN: Both the sender and receiver devices use the synchronization (SYN) flag in only the first packet that is sent.
- ACK: The receiver devices use the acknowledgment (ACK) flag in the packet that is sent to acknowledge the successful receipt of a packet.
- FIN: The sender device uses the finished (FIN) flag in the last packet to indicate that there is no more data to be sent.
- RST: The receiver device uses the reset (RST) flag in the packet sent to the sender device when the receiver device receives a packet that is not expected.

- URG: The sender device uses the urgent (URG) flag in the packets to notify the receiver device to process the urgent packets before processing all other packets.
- PSH: The receiver device uses the push (PSH) flag that is similar to the URG flag and tells the receiver to process these packets as soon as they are received without waiting for any other packets to be received.
- EST: When a remote host receives TCP packets with a SYN flag set and if it does not support such a service, the remote host replies with an EST flag set in the packet. EST flag signifies both ACK and RST flags set in the packet.

Configuration guidelines and restrictions for ACLs based on TCP flags

This topic lists the configuration guidelines and restrictions for access control lists (ACLs) that filter packets based on TCP flags on the Cisco 8000 Series Router.

The following restrictions apply when you configure ACLs based on TCP flags:

- Before Release 7.3.15, you cannot filter egress IPv6 packets based on TCP flags.

The following guidelines apply when you configure ACLs based on TCP flags:

- You must configure the PBR policy to explicitly match the TCP protocol along with the required TCP flags in the class-map to ensure precise traffic classification and prevent unintended redirection.

Configure ACLs based on TCP flags

This topic describes how to configure an access control list (ACL) that permits or denies packets based on TCP flags on the Cisco 8000 Series Router.

Use this procedure to configure an ACL that filters packets based on TCP flags. Use the **match-any** keyword to permit or deny packets based on whether any of the configured TCP flags is set. Use the **match-all** keyword to permit or deny packets based on whether all the configured TCP flags are set.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create an IPv4 ACL named `ACL-TCP`.

   Example

   ```
   Router(config)# ipv4 access-list ACL-TCP
   ```

3. Configure an ACE using the **match-any** keyword to permit packets based on the required TCP flags, and commit the configuration.

   The following example shows how to permit packets that have either the PSH or URG TCP flags set:

   Example

   ```
   Router(config-ipv4-acl)# 10 permit tcp any any match-any + urg + psh
   Router(config-ipv4-acl)# commit
   ```

The following example shows how to permit packets that have either the SYN or ACK TCP flag set:

Example

```
Router(config-ipv4-acl)# 10 permit tcp any any match-any + syn + ack
Router(config-ipv4-acl)# commit
```

The following example shows how to permit packets that have the SYN flag set or the ACK flag not set:

Example

```
Router(config-ipv4-acl)# 10 permit tcp any any match-any + syn - ack
Router(config-ipv4-acl)# commit
```

4. Configure an ACE using the **match-all** keyword to permit packets based on the required TCP flags, and commit the configuration.

   The following example shows how to permit packets that have both the URG flag and FIN flag set:

   Example

   ```
   Router(config-ipv4-acl)# 10 permit tcp any any match-all + urg + fin
   Router(config-ipv4-acl)# commit
   ```

   The following example shows how to permit packets that have both the SYN flag set and the ACK flag not set:

   Example

   ```
   Router(config-ipv4-acl)# 10 permit tcp any any match-all + syn - ack
   Router(config-ipv4-acl)# commit
   ```

   You have successfully configured an ACL based on TCP flags.

8 Hybrid and object-group ACLs

Topics:

This chapter describes hybrid ACLs and object-group ACLs on the Cisco 8000 Series Router, including network and port object-group configuration, compression verification, and 24-bit bincode support for egress object-group ACLs.

This chapter contains the following sections:

Hybrid ACLs

This topic describes hybrid ACLs, also known as object-group ACLs on the Cisco 8000 Series Router, which simplify access control by grouping users, devices, or protocols into object groups and reducing the number of Access Control Entries (ACEs).

A hybrid ACL is an access control mechanism that

- simplifies access control policies by grouping users, devices, or protocols into object groups
- reduces the number of Access Control Entries (ACEs), making ACLs easier to manage and more readable, and
- optimizes TCAM storage by using object-group ACLs instead of conventional ACLs, allowing compression levels for object-group ACLs and supporting up to 4000 ACEs per line card in the ingress direction.

The bit compression for OG-ACLs is a method that

- enhances capability by expanding the compression result sizes from 24 bits to 26 bits for both IPv4 and IPv6 ingress OG-ACLs, and
- supports longer lists of source or destination prefixes with variable lengths to address requirements such as wider compression results.

Table 41: Feature History Table

Feature Name	Release Information	Feature Description
Egress hybrid ACL support	Release 26.3.1	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.
Viewing TCAM usage for source prefixes	Release 26.3.1	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.
Viewing TCAM usage for source prefixes	Release 26.3.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) (select variants only*) *This feature is supported on Cisco 8711-28H8F-M.

Feature Name	Release Information	Feature Description
24 bit bincode support for egress object-group ACLs	Release 26.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100]), 8010 [ASIC: A100]; Modular Systems (8800 [LC ASIC: P100]) You can now improve network policy granularity and control by supporting 24-bit bincode sizes for egress object-group ACLs. This enhancement improves the efficiency of handling extended network object-groups and supports the inclusion of larger, more detailed prefix lists. This feature introduces the **hw-module profile tcam format og-compr-id-extension egress** command.
Slice-Aware Prefix Programming for Egress Object-Group ACLs	Release 26.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q200, P100]) This feature enhances hardware efficiency by programming Egress Object-Group ACL (OG-ACL) prefixes only on the active slices where an ACL is applied, rather than replicating them across all slices. Previously, prefixes were duplicated on every slice, leading to higher TCAM and HCAM usage and reduced performance. The software now automatically identifies and programs prefixes only for relevant slices, dynamically replicating them when new interfaces or bundle members are added. This optimization increases scalability for other hardware-based applications, supports both IPv4 and IPv6 OG-ACLs, and requires no additional configuration.
Egress hybrid ACL support	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 routers.

Feature Name	Release Information	Feature Description
Viewing TCAM usage for source prefixes	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 routers.
Egress hybrid ACL support	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: K100], 8010 [ASIC: A100])(select variants only*), 8700 [ASIC: K100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D P100-based ASICs now support a larger bincode size of 18 bits instead of 14 bits. This increase allows the forwarding pipeline to encode more feature states simultaneously, enabling support for complex egress processing scenarios such as hybrid egress ACL deployments.
Enhanced 26 bit compression for object-group ACLs	Release 25.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100]); Modular Systems (8800 [LC ASIC: P100]) You can now improve network policy granularity and control by supporting 26-bit bincode sizes for both source and destination prefixes in ingress object-group ACLs. This enhancement improves the efficiency of handling extended network object-groups and supports the inclusion of larger, more detailed prefix lists. This feature introduces the **hw-module profile tcam format og-compr-id-extension** command.

Feature Name	Release Information	Feature Description
Egress Hybrid ACL Support	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) Compression levels for object-group ACLs, also known as hybrid ACLs, can now be applied to egress traffic. This ACL compression optimizes the usage of TCAM space, allowing the router to support additional ACLs or features and ensuring efficient utilization of the limited TCAM resources available. This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Viewing TCAM usage for source prefixes	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]) This features ensures that source prefixes of the egress hybrid ACLs are now stored in a separate TCAM interface, enabling accurate resource monitoring and tracking. Use the **show controllers npu resource egressacltcam location** command to view the TCAM usage. In earlier releases, while the destination prefixes of the hybrid ACLs were stored in the ACL TCAM interface, the source prefixes were unreported, making it challenging to monitor their accurate TCAM resource usage.

Feature Name	Release Information	Feature Description
Enhanced 24 bit compression for object-group ACLs	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) This release introduces support for more efficient handling of extended network object-groups by expanding source and destination compression results. This is achieved by enabling support for 24 bit bincode sizes for both source and destination prefixes in the ingress object-group ACLs, which accommodates larger and more detailed prefix lists for supporting more complex network object-groups and configurations. Prior to release 25.1.1, this capability was limited to 20 bit compression for both source and destination prefixes, restricting the length and variability of network prefixes that could be managed within ACLs.
Egress Hybrid ACL Support	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Egress Hybrid ACL Support	Release 7.10.1	From this release onwards, you can apply compression levels for object-group ACLs, also known as hybrid ACLs, at the egress traffic. Because ACL compression optimizes TCAM space usage, it enables the router to accommodate additional ACLs or features. This feature is supported only on Q200 ASIC based systems.

Feature Name	Release Information	Feature Description
Hybrid ACLs	Release 7.3.1	You can apply compression levels for object-group ACLs and attach up to 4000 ACEs per line card in the ingress direction. This leads to optimal TCAM space usage and resources utilization.

Benefits of hybrid ACLs

This topic describes the benefits of hybrid ACLs.

Hybrid ACLs provide these benefits:

- streamline access control by grouping users, devices, or protocols,
- enhance scalability and efficiency by reducing the number of ACEs per ACL, and
- support larger bincode sizes to accommodate longer and more detailed prefix lists.

Streamline access control

Hybrid ACLs or object-group ACLs classifies users, devices, or protocols into groups so that you can have a group-level access control policy. Instead of specifying individual IP addresses, protocols, and port numbers in multiple ACEs, you can specify just the object group in a single ACL.

Enhance scalability and efficiency

This feature is very beneficial in large scale networks which currently contain hundreds of ACLs. By using the object-group ACL feature, the number of ACEs per ACL are significantly reduced. Object-group ACLs are also more readable, and easier to manage than conventional ACLs. Using object-group ACLs instead of conventional ACLs optimizes the storage needed in TCAM.

Bincode size enhancement

Starting with Cisco IOS XR Release 25.4.1, you can use P100-based ASICs that support an increased bincode size of 18 bits from 14 bits. This enhancement allows you to process network policies and configurations that require longer or more detailed prefix lists. This enhancement enables greater scalability and efficiency for managing network object-groups and ACLs on P100-based ASICs.

Starting with Cisco IOS XR Release 25.2.1, the bincode size for both source and destination prefixes has been increased from 24 bits to 26 bits. This enhancement supports network object-groups with longer prefix lists, improving scalability for applications requiring longer source and destination prefix lists.

Types of hybrid ACLs

This topic lists the types of hybrid or object-group ACLs that you can create on the Cisco 8000 Series Router.

You can create two types of object-group ACLs on Cisco IOS XR:

- Network object-group ACLs: Consist of groups of host IP Addresses and network IP addresses.
- Port object-group ACLs: Consist of groups of ports and supporting Layer 3 or Layer 4 protocols.

ACL compression

This topic describes ACL compression for object-group ACLs, which reduces TCAM usage by compressing the source and destination IP prefix fields of an ACE.

ACL compression is an object-group ACL mechanism that

- accommodates a large number of ACEs by compressing selected ACE fields,
- compresses the source IP prefix and destination IP prefix fields of an ACE, and
- provides four compression levels for the access-group configuration on an ingress interface.

From Release 25.1.1 onwards, you can apply compression levels for object-group ACLs, also known as hybrid ACLs, at the egress traffic on P100-based ASICs.

Compression levels

There are four compression levels in the access-group configuration for an ACL on an ingress interface:

- **Compress level 0**: No compression is done on the ACE fields.

  In this mode, the object-group ACL behaves like a traditional ACL.

- **Compress level 1**: Only the source IP field in an ACE is compressed.

  A User-Defined Key (UDK) for compression level 1 can be configured using the following commands:

```
hw-module profile tcam format access-list ipv4 src-object-group src-port
dst-port proto tcp-flags frag-bit dst-addr
hw-module profile tcam format access-list ipv6 src-object-group src-port
dst-port next-hdr frag-bit tcp-flags dst-addr
```

- **Compress level 2**: Two fields (source IP and destination IP) in an ACE are compressed.

  In this mode, for ingress traffic, central EM (Exact Match) is used for prefix lookup, and internal TCAM is used for ACE lookup.

  For egress traffic, central TCAM is used for both prefix lookup and ACE lookup.

- **Compress level 4**: Only the destination IP field in an ACE is compressed.

  A UDK for compression level 4 can be configured using the following commands:

```
hw-module profile tcam format access-list ipv4 src-addr src-port dst-port
proto tcp-flags frag-bit dst-object-group
hw-module profile tcam format access-list ipv6 src-addr src-port dst-port
next-hdr frag-bit tcp-flags dst-object-group
```

Restrictions for hybrid ACLs

This topic lists the restrictions for egress hybrid ACLs on the Cisco 8000 Series Router.

Hybrid ACL restrictions

These restrictions apply when you configure hybrid ACLs:

- hybrid ACLs can only be configured to an interface. They cannot be used or referenced by applications like SSH, SNMP, NTP.
- To delete an hybrid, you must first delete it from all ACLs.
- You cannot configure hybrid ACLs along with QoS policies.
- hybrid ACLs are not supported in any policy based configuration.
- Any inline ACE update to an object group ACL clears complete stats of the ACL.

Egress hybrid ACL restrictions

These restrictions apply when you configure egress hybrid ACLs:

- The 8011-4G24Y4H-I router does not support egress hybrid ACLs.
- Egress hybrid ACLs do not support the 24-bit compression feature for object-group ACLs; this feature applies only to ingress ACLs.
- The slice-aware prefix programming for egress object-group ACLs optimization is specific to Egress Object-Group ACLs. Ingress OG-ACLs continue to use LPM (Longest Prefix Match) or CEM (Content Exact Match) programming.

Configure a hybrid ACL

This topic describes the prerequisites and restrictions for configuring an hybrid or hyrid ACL.

You can configure two types of hybrid ACLs:

- Network hybrid ACLs. See the following tasks:
  - Configure a network hybrid ACL for compress level 1 for IPv4 address on page 186
  - Configure a network hybrid ACL for compress level 1 for IPv6 address on page 188
  - Configure a network hybrid ACL for compress level 2 for IPv4 address on page 189
  - Configure a network hybrid ACL for compress level 2 for IPv6 address on page 190
  - Verify source prefix for an egress network hybrid ACL for compress level 2 on page 191
  - Verify TCAM usage for IPv4 and IPv6 ACLs on page 192
  - Configure a network hybrid ACL for compress level 4 on page 198
  - Configure user-defined TCAM keys for IPv4 and IPv6 for compress levels 1 and 4 on page 200
  - Configure an egress IPv4 or IPv6 hybrid ACL on a HundredGigE Interface on page 201
- Port hybrid ACLs. See Configure a port object-group ACL on page 203.

Before you begin

You must be aware of this information that apply to hybrid ACLs:

- You can configure ACLs that contain both conventional and hybrid ACEs.
- You can modify the objects in an object group dynamically without redefining the object group or the ACE that references the object group.
- You can configure an hybrid ACL multiple times with a source group, or a destination group, or both source and destination groups.

Configure a network hybrid ACL for compress level 1 for IPv4 address

This topic describes how to configure a network hybrid ACL for Compress Level 1 for an IPv4 address.

Use this procedure to configure a network hybrid ACL that applies Compress Level 1 to an IPv4 address on a HundredGigE interface.

> **Note**
>
> Compress Level 1 supports prefix masks /n, not arbitrary address masks a.b.c.d.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create a network object group named `netobj1` and add host, network, and range entries.

   Example

   ```
   Router(config)# object-group network ipv4 netobj1
   Router(config-object-group-ipv4)# description my-network-object
   Router(config-object-group-ipv4)# host 10.1.1.1
   Router(config-object-group-ipv4)# 10.2.1.0 255.255.255.0
   Router(config-object-group-ipv4)# range 10.3.1.10 10.3.1.50
   ```

3. Create an IPv4 access list that references the object group.

   Example

   ```
   Router(config)# ipv4 access-list network-object-acl permit ipv4 net-group
   netobj1 any
   ```

4. Apply the access list to a HundredGigE interface with **compress level 1** and commit the configuration.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/10/3
   Router(config-if)# ipv4 address 1.1.1.1/24
   Router(config-if)# no shut
   Router(config-if)# ipv4 access-group network-object-acl ingress compress level
    1
   Router(config-if)# commit
   Router(config-if)# exit
   ```

5. Verify the configuration by using the **show run** command.

   Example

   ```
   Router(config)# show run
   Tue Mar 28 10:37:55.737 IST

   Building configuration...
   !! IOS XR Configuration 0.0.0
   ...
   ```

```
!
object-group network ipv4 netobj1
 10.2.1.0/24
 host 10.1.1.1
 range  10.3.1.10 10.3.1.50
 description my-network-object
!
!
ipv4 access-list network-object-acl
 10 permit ipv4 net-group netobj1 any
!
interface HundredGigE 0/0/10/3
 ipv4 address 1.1.1.1 255.255.255.0
 ipv4 access-group network-object-acl ingress compress level 1
!
```

You have successfully configured a network hybrid ACL for Compress Level 1 for an IPv4 address.

Configure a network hybrid ACL for compress level 1 for IPv6 address

This topic describes how to configure a network hybrid ACL for Compress Level 1 for an IPv6 address.

Use this procedure to configure a network hybrid ACL that applies Compress Level 1 to an IPv6 address on a HundredGigE interface.

> **Note**
>
> Compress Level 1 supports prefix masks /n, not arbitrary address masks a.b.c.d.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create a network object group named `netobj1` and add host, network, and range entries.

   Example

   ```
   Router(config)# object-group network ipv6 netobj1
   Router(config-object-group-ipv6)# description my-network-object
   Router(config-object-group-ipv6)# host 2001:DB8:1::1
   Router(config-object-group-ipv6)# 2001:DB8::1 2001:DB8:0:ABCD::1
   Router(config-object-group-ipv6)# range 2001:DB8::2 2001:DB8::5
   ```

3. Create an IPv6 access list that references the object group.

Example

```
Router(config)# ipv6 access-list network-object-acl permit ipv6 net-group
netobj1 any
```

4. Apply the access list to a HundredGigE interface with **compress level 1** and commit the configuration.

Example

```
Router(config)# interface HundredGigE 0/0/10/3
Router(config-if)# ipv6 address 2001:DB8::1/32
Router(config-if)# no shut
Router(config-if)# ipv6 access-group network-object-acl ingress compress level
 1
Router(config-if)# commit
Router(config-if)# exit
```

You have successfully configured a network hybrid ACL for Compress Level 1 for an IPv6 address.

Configure a network hybrid ACL for compress level 2 for IPv4 address

This topic describes how to configure a network hybrid ACL for Compress Level 2 for an IPv4 address on the Cisco 8000 Series Router.

Use this procedure to configure a network hybrid ACL that applies Compress Level 2 to an IPv4 address on a HundredGigE interface.

> **Note**
>
> Compress Level 2 supports prefix masks /n, not arbitrary address masks a.b.c.d.

Procedure

1. Enter global configuration mode.

Example

```
Router# configure
```

2. Create a network object group named `netobj1` and add host, network, and range entries.

Example

```
Router(config)# object-group network ipv4 netobj1
Router(config-object-group-ipv4)# description my-network-object
Router(config-object-group-ipv4)# host 10.1.1.1
Router(config-object-group-ipv4)# 10.2.1.0 255.255.255.0
Router(config-object-group-ipv4)# range 10.3.1.10 10.3.1.50
```

3. Create an IPv4 access list that references the object group.

Example

```
Router(config)# ipv4 access-list network-object-acl permit ipv4 net-group
netobj1 any
```

4. Apply the access list to a HundredGigE interface with **compress level 2** and commit the configuration.

Example

```
Router(config)# interface HundredGigE 0/0/10/3
Router(config-if)# ipv4 address 1.1.1.1/24
Router(config-if)# no shut
Router(config-if)# ipv4 access-group network-object-acl ingress compress level
 2
Router(config-if)# commit
Router(config-if)# exit
```

5. Verify the configuration by using the **show run** command.

Example

```
Router(config)# show run
Tue Mar 28 10:37:55.737 IST

Building configuration...
!! IOS XR Configuration 0.0.0
...

!
object-group network ipv4 netobj1
 10.2.1.0/24
 host 10.1.1.1
 range  10.3.1.10 10.3.1.50
 description my-network-object
!
!
ipv4 access-list network-object-acl
 10 permit ipv4 net-group netobj1 any
!
interface HundredGigE 0/0/10/3
 ipv4 address 1.1.1.1 255.255.255.0
 ipv4 access-group network-object-acl ingress compress level 2
!
```

You have successfully configured a network hybrid ACL for Compress Level 2 for an IPv4 address.

Configure a network hybrid ACL for compress level 2 for IPv6 address

This topic describes how to configure a network hybrid ACL for Compress Level 2 for an IPv6 address.

Use this procedure to configure a network hybrid ACL that applies Compress Level 2 to an IPv6 address on a HundredGigE interface.

> **Note**
>
> Compress Level 2 supports prefix masks /n, not arbitrary address masks a.b.c.d.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create a network object group named `netobj1` and add host, network, and range entries.

   Example

   ```
   Router(config)# object-group network ipv6 netobj1
   Router(config-object-group-ipv6)# description my-network-object
   Router(config-object-group-ipv6)# host 2001:DB8:1::1
   Router(config-object-group-ipv6)# 2001:DB8::1 2001:DB8:0:ABCD::1
   Router(config-object-group-ipv6)# range 2001:DB8::2 2001:DB8::5
   ```

3. Create an IPv6 access list that references the object group.

   Example

   ```
   Router(config)# ipv6 access-list network-object-acl permit ipv6 net-group
   netobj1 any
   ```

4. Apply the access list to a HundredGigE interface with **compress level 2** and commit the configuration.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/10/3
   Router(config-if)# ipv6 address 2001:DB8::1/32
   Router(config-if)# no shut
   Router(config-if)# ipv6 access-group network-object-acl ingress compress level
    2
   Router(config-if)# commit
   Router(config-if)# exit
   ```

   You have successfully configured a network hybrid ACL for Compress Level 2 for an IPv6 address.

Verify source prefix for an egress network hybrid ACL for compress level 2

This topic describes how to verify the source prefix for an egress network hybrid ACL that uses Compress Level 2.

Use this procedure after you configure an IPv4 or IPv6 ACL with Compress Level 2 to verify that the source prefix is captured in the TCAM.

Procedure

Verify the source prefix for the egress network hybrid ACL by using the **show access-lists** command with the **hardware egress verify location** options.

In the following example, the **SRC IP** prefix type records **2** entries for the `network-object-acl` ACL.

Example

```
Router# show access-lists network-object-acl hardware egress verify location
0/0/cpu0
Verifying TCAM entries for network-object-acl
Please wait...
INTF    NPU lookup  ACL # intf Total  compression Total   result failed(Entry)
 TCAM entries
                type    ID  shared ACES   prefix-type Entries        ACE SEQ
#     verified
 ---------- --- ------- --- ------ ------ ----------- ------- ------
------------ ------------

HundredGigE0_0_0_0 (ifhandle: 0x130)

              0 IPV4      1      1      2 COMPRESSED        2 passed
                 2
                                          SRC IP            2 passed
                 2
                                          DEST IP           2 passed
                 2
                                          SRC PORT          0 passed
                 0
```

You have successfully verified the source prefix for an egress network hybrid ACL with Compress Level 2.

Verify TCAM usage for IPv4 and IPv6 ACLs

This topic describes how to verify Ternary Content Addressable Memory (TCAM) usage for IPv4 and IPv6 ACLs.

Use this procedure to verify TCAM resource utilization for IPv4 and IPv6 ACLs, and to confirm the location of the source and destination prefixes.

Table 42: TCAM utilization for IPv4 and IPv6 ACLs

ACL	Source Prefix Location	Destination Prefix Location
IPv4	narrow_1 band	narrow_0 band. The ACEs are stored in the narrow_0 band.
IPv6	narrow_1 band	narrow_0 band. ACEs are stored in the wide band.

Procedure

1. Verify TCAM resource utilization for IPv4 and IPv6 ACLs by using the **show controllers npu resource egressacltcam location** command.

Example

```
/* This is a sample for IPv4 ACLs that captures narrow_1 and narrow_0 band
for source and destination prefixes. */

Router#show controllers npu resources egressacltcam location$
Wed Jan  8 07:53:02.101 UTC
HW Resource Information
    Name                            : egress_acl_tcam
    Asic Type                       : Q200
    Insight exported timestamp      : 2025.Jan.08 07:52:58 UTC

NPU-0
OOR Summary
        Red Threshold               : 95 %
        Yellow Threshold            : 80 %



Current Hardware Usage
    Name: egress_acl_tcam


       Name: narrow_0


           Name: slice_0
Estimated Max Entries       : 9728
               Total In-Use                 : 2        (0 %)
               High Water Mark              : 2
               High Water Mark Time         : 2025.Jan.08 07:52:52 UTC


           Name: slice_1
               Estimated Max Entries        : 9728
               Total In-Use                 : 2        (0 %)
               High Water Mark              : 2
               High Water Mark Time         : 2025.Jan.08 07:52:52 UTC


           Name: slice_2
               Estimated Max Entries        : 9728
               Total In-Use                 : 4        (0 %)
               High Water Mark              : 4
               High Water Mark Time         : 2025.Jan.08 07:52:52 UTC


       Name: narrow_1


           Name: slice_0
               Estimated Max Entries        : 9728
               Total In-Use                 : 2        (0 %)
               High Water Mark              : 2
               High Water Mark Time         : 2025.Jan.08 07:52:52 UTC


           Name: slice_1
               Estimated Max Entries        : 9728
               Total In-Use                 : 2        (0 %)
               High Water Mark              : 2
               High Water Mark Time         : 2025.Jan.08 07:52:52 UTC
```

```
           Name: slice_2
               Estimated Max Entries       : 9728
               Total In-Use                : 2         (0 %)
               High Water Mark             : 2
               High Water Mark Time        : 2025.Jan.08 07:52:52 UTC


       Name: wide


           Name: slice_0
               Estimated Max Entries       : 5118
               Total In-Use                : 0         (0 %)


           Name: slice_1
               Estimated Max Entries       : 5118
               Total In-Use                : 0         (0 %)


           Name: slice_2
               Estimated Max Entries       : 5116
               Total In-Use                : 0         (0 %)



NPU-1
OOR Summary
        Red Threshold              : 95 %
        Yellow Threshold           : 80 %



Current Hardware Usage
    Name: egress_acl_tcam


       Name: narrow_0


           Name: slice_0
               Estimated Max Entries       : 10240
               Total In-Use                : 0         (0 %)


           Name: slice_1
               Estimated Max Entries       : 10240
               Total In-Use                : 0         (0 %)


           Name: slice_2
               Estimated Max Entries       : 10240
               Total In-Use                : 0         (0 %)


       Name: narrow_1


           Name: slice_0
               Estimated Max Entries       : 10240
               Total In-Use                : 0         (0 %)
```

```
           Name: slice_1
               Estimated Max Entries       : 10240
               Total In-Use                : 0        (0 %)


           Name: slice_2
               Estimated Max Entries       : 10240
               Total In-Use                : 0        (0 %)


       Name: wide


           Name: slice_0
               Estimated Max Entries       : 5120
               Total In-Use                : 0        (0 %)


           Name: slice_1
               Estimated Max Entries       : 5120
               Total In-Use                : 0        (0 %)


           Name: slice_2
               Estimated Max Entries       : 5120
               Total In-Use                : 0        (0 %)
```

Example
This is a sample for IPv6 ACLs that captures narrow_1 and narrow_0 band for source and destination prefixes.

```
Router#show controllers npu resources egressacltcam locatio$
Wed Jan  8 07:59:15.111 UTC
HW Resource Information
    Name                            : egress_acl_tcam
    Asic Type                       : Q200
    Insight exported timestamp      : 2025.Jan.08 07:58:58 UTC

NPU-0
OOR Summary
        Red Threshold               : 95 %
        Yellow Threshold            : 80 %



Current Hardware Usage
    Name: egress_acl_tcam


       Name: narrow_0


           Name: slice_0
               Estimated Max Entries       : 9728
               Total In-Use                : 2        (0 %)
               High Water Mark             : 2
               High Water Mark Time        : 2025.Jan.08 07:52:52 UTC
```

```
            Name: slice_1
                Estimated Max Entries       : 9728
                Total In-Use                : 2         (0 %)
                High Water Mark             : 2
                High Water Mark Time        : 2025.Jan.08 07:52:52 UTC


            Name: slice_2
                Estimated Max Entries       : 9724
                Total In-Use                : 2         (0 %)
                High Water Mark             : 4
                High Water Mark Time        : 2025.Jan.08 07:52:52 UTC


        Name: narrow_1


            Name: slice_0
                Estimated Max Entries       : 9728
                Total In-Use                : 2         (0 %)
                High Water Mark             : 2
                High Water Mark Time        : 2025.Jan.08 07:52:52 UTC


            Name: slice_1
                Estimated Max Entries       : 9728
                Total In-Use                : 2         (0 %)
                High Water Mark             : 2
                High Water Mark Time        : 2025.Jan.08 07:52:52 UTC


            Name: slice_2
                Estimated Max Entries       : 9724
                Total In-Use                : 2         (0 %)
                High Water Mark             : 2
                High Water Mark Time        : 2025.Jan.08 07:52:52 UTC


        Name: wide


            Name: slice_0
                Estimated Max Entries       : 5118
                Total In-Use                : 0         (0 %)


            Name: slice_1
                Estimated Max Entries       : 5118
                Total In-Use                : 0         (0 %)


            Name: slice_2
                Estimated Max Entries       : 5118
                Total In-Use                : 4         (0 %)
                High Water Mark             : 4
                High Water Mark Time        : 2025.Jan.08 07:58:52 UTC



NPU-1
OOR Summary
        Red Threshold               : 95 %
```

```
        Yellow Threshold            : 80 %


Current Hardware Usage
    Name: egress_acl_tcam


       Name: narrow_0


           Name: slice_0
               Estimated Max Entries       : 10240
               Total In-Use                : 0         (0 %)


           Name: slice_1
               Estimated Max Entries       : 10240
               Total In-Use                : 0         (0 %)


           Name: slice_2
               Estimated Max Entries       : 10240
               Total In-Use                : 0         (0 %)


       Name: narrow_1


           Name: slice_0
               Estimated Max Entries       : 10240
               Total In-Use                : 0         (0 %)


           Name: slice_1
               Estimated Max Entries       : 10240
               Total In-Use                : 0         (0 %)


           Name: slice_2
               Estimated Max Entries       : 10240
               Total In-Use                : 0         (0 %)


       Name: wide


           Name: slice_0
               Estimated Max Entries       : 5120
               Total In-Use                : 0         (0 %)


           Name: slice_1
               Estimated Max Entries       : 5120
               Total In-Use                : 0         (0 %)


           Name: slice_2
               Estimated Max Entries       : 5120
               Total In-Use                : 0         (0 %)
```

2. For IPv4 ACLs, verify that the source prefixes are captured in **narrow_1**, and that the **narrow_0** band captures the destination prefix and ACEs.
3. For IPv6 ACLs, verify that the source prefix is captured in the **narrow_1** band, the destination prefix is captured in the **narrow_0** band, and the ACEs are captured in the **wide** band.

   You have successfully verified TCAM usage for IPv4 and IPv6 ACLs.

Configure a network hybrid ACL for compress level 4

This topic describes how to configure a network object-group ACL for Compress Level 4 for IPv4 and IPv6 addresses.

Use this procedure to configure a network object-group ACL that applies Compress Level 4 to IPv4 and IPv6 addresses on a HundredGigE interface.

> **Note**
>
> Compress Level 4 supports prefix masks /n, not arbitrary address masks a.b.c.d.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create an IPv4 network object group named `netobj1` and add host, network, and range entries.

   Example

   ```
   Router(config)# object-group network ipv4 netobj1
   Router(config-object-group-ipv4)# description my-network-object
   Router(config-object-group-ipv4)# host 10.1.1.1
   Router(config-object-group-ipv4)# 10.2.1.0 255.255.255.0
   Router(config-object-group-ipv4)# range 10.3.1.10 10.3.1.50
   ```

3. Create an IPv4 access list that references the object group.

   Example

   ```
   Router(config)# ipv4 access-list network-object-acl permit ipv4 net-group
   netobj1 any
   ```

4. Apply the IPv4 access list to a HundredGigE interface with **compress level 4** and commit the configuration.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/10/3
   Router(config-if)# ipv4 address 1.1.1.1/24
   ```

```
Router(config-if)# no shut
Router(config-if)# ipv4 access-group network-object-acl ingress compress level
 4
Router(config-if)# commit
Router(config-if)# exit
```

5. Create an IPv6 network object group named `netobj1` and add host, network, and range entries.

   Example

```
Router(config)# object-group network ipv6 netobj1
Router(config-object-group-ipv6)# description my-network-object
Router(config-object-group-ipv6)# host 2001:DB8:1::1
Router(config-object-group-ipv6)# 2001:DB8::1 2001:DB8:0:ABCD::1
Router(config-object-group-ipv6)# range 2001:DB8::2 2001:DB8::5
```

6. Create an IPv6 access list that references the object group.

   Example

```
Router(config)# ipv6 access-list network-object-acl permit ipv6 net-group
netobj1 any
```

7. Apply the IPv6 access list to a HundredGigE interface with **compress level 4** and commit the configuration.

   Example

```
Router(config)# interface HundredGigE 0/0/10/3
Router(config-if)# ipv6 address 2001:DB8::1/32
Router(config-if)# no shut
Router(config-if)# ipv6 access-group network-object-acl ingress compress level
 4
Router(config-if)# commit
Router(config-if)# exit
```

8. Verify the configuration by using the **show run** command.

   Example

```
Router(config)# show run
Tue Mar 28 10:37:55.737 IST

Building configuration...
!! IOS XR Configuration 0.0.0
...

!
object-group network ipv4 netobj1
 10.2.1.0/24
 host 10.1.1.1
 range  10.3.1.10 10.3.1.50
 description my-network-object
!
!
ipv4 access-list network-object-acl
 10 permit ipv4 net-group netobj1 any
!
```

```
interface HundredGigE 0/0/10/3
 ipv4 address 1.1.1.1 255.255.255.0
 ipv4 access-group network-object-acl ingress compress level 4
!
```

You have successfully configured a network object-group ACL for Compress Level 4.

Configure user-defined TCAM keys for IPv4 and IPv6 for compress levels 1 and 4

This topic describes how to configure User-Defined TCAM Keys (UDK) for IPv4 and IPv6 ACLs for Compress Levels 1 and 4.

Use this procedure to define UDKs for IPv4 and IPv6 for Compress Level 1 and Compress Level 4.

Before you begin

- Modification for UDK is not supported using the **hw-module profile tcam format** command.
- First, remove the existing UDK using the **no hw-module profile tcam format** command, then add a new UDK definition.
- Make sure that you reload the line card for the UDK configuration to take effect.
- If you configure UDK, you cannot use the default keys. But you can explicitly define the default fields in UDK.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Define the UDK for IPv4 and IPv6 for Compress Level 1.

   Example

   ```
   Router(config)# hw-module profile tcam format access-list ipv4 src-object-group
    src-port dst-port proto tcp-flags frag-bit dst-addr
   Router(config)# hw-module profile tcam format access-list ipv6 src-object-group
    src-port dst-port next-hdr frag-bit tcp-flags dst-addr
   ```

3. Define the UDK for IPv4 and IPv6 for Compress Level 4.

   Example

   ```
   Router(config)# hw-module profile tcam format access-list ipv4 src-addr
   src-port dst-port proto tcp-flags frag-bit dst-object-group
   Router(config)# hw-module profile tcam format access-list ipv6 src-addr
   src-port dst-port next-hdr frag-bit tcp-flags dst-object-group
   ```

4. Manually reload the node on the line card for the UDK configuration to take effect.
5. Configure the ACL using the fields defined in the UDK, and attach the ACL to an interface in the ingress direction.

You have successfully configured User-Defined TCAM Keys for IPv4 and IPv6 for Compress Levels 1 and 4.

Configure an egress IPv4 or IPv6 hybrid ACL on a HundredGigE Interface

This topic describes how to configure an egress IPv4 or IPv6 hybrid ACL on a HundredGigE interface.

From Release 7.10.1 onwards, you can configure an egress IPv4 or IPv6 hybrid ACL on an interface in Q200 ASIC based systems. This allows you to separate address prefixes and ports into two object groups or access control entries (ACEs), which improves network traffic security and allows better use of space and resources to accommodate more ACLs.

Use this procedure to configure an egress IPv4 hybrid ACL on a HundredGigE interface. You can configure an egress IPv6 hybrid ACL by using the same steps with IPv6 syntax.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create a network object group named `netobj1` and commit the configuration.

   Example

   ```
   Router(config)# object-group network ipv4 netobj1
   Router(config-object-group-ipv4)# 12.1.0.0/16
   Router(config-object-group-ipv4)# 14.1.0.0/16
   Router(config-object-group-ipv4)# 105.2.0.0/16
   Router(config-object-group-ipv4)# commit
   ```

3. Create an IPv4 access list named `test-v4` that references the object group, and commit the configuration.

   Example

   ```
   Router(config)# ipv4 access-list test-v4
   Router(config-ipv4-acl)# 10 deny ipv4 any net-group netobj1
   Router(config-ipv4-acl)# 20 permit ipv4 any any
   Router(config-ipv4-acl)# commit
   ```

4. Apply the access list to a HundredGigE interface with the appropriate compression level and commit the configuration.

   The following example applies the access list with **compress level 1**.

   Example

   ```
   Router(config)# interface fourHundredGigE 0/0/0/1
   Router(config-if)# ipv4 access-group test-v4 egress compress level 1
   Router(config-if)# commit
   ```

   The following example applies the access list with **compress level 2**.

Example

```
Router(config)# interface fourHundredGigE 0/0/0/1
Router(config-if)# ipv4 access-group test-v4 egress compress level 2
Router(config-if)# commit
```

The following example applies the access list with **compress level 4**.

Example

```
Router(config)# interface fourHundredGigE 0/0/0/1
Router(config-if)# ipv4 access-group test-v4 egress compress level 4
Router(config-if)# commit
```

5. Verify the running configuration for the network object group, access list, and interface by using the **show running-config** command.

Example

```
Router# show running-config object-group network ipv4 netobj1
Tue Jul 25 22:44:20.516 UTC
object-group network ipv4 netobj1
12.1.0.0/16
14.1.0.0/16
105.2.0.0/16
!

Router# show running-config ipv4 access-list test-v4
Tue Jul 25 22:44:39.010 UTC
ipv4 access-list test-v4
10 deny ipv4 any net-group netobj1
20 permit ipv4 any any
!

Router# show running-config interface fourHundredGigE 0/0/0/1
Tue Jul 25 22:45:13.769 UTC
interface FourHundredGigE0/0/0/1
ipv4 address 12.1.0.1 255.255.255.0
ipv6 address 12:1::1/64
ipv4 access-group test-v4 egress compress level 1
!
```

6. Verify the packet count for the filtered traffic by using the **show access-lists ipv4 hardware egress location** command.

Example

```
Router# show access-lists ipv4 test-v4 hardware egress location 0/RP0/CPU0
Tue Jul 25 22:53:38.980 UTC
ipv4 access-list test-v4
10 deny ipv4 any net-group netobj1 (47 matches)
20 permit ipv4 any any (100008 matches)
```

You have successfully configured an egress IPv4 or IPv6 hybrid ACL on a HundredGigE interface. The filtered packet count in the example is 47 and 100008, which implies that traffic is filtered according to the rules in the ACL.

Configure a port object-group ACL

This topic describes how to configure a port object-group ACL on the Cisco 8000 Series Router. A port object-group can contain a single or multiple port objects.

A port object-group can contain a single or multiple port objects.

Before you begin

If you configure port object-group ACLs with compression level 2, it does not result in a lower number of TCAM entries because the compression algorithm only uses the source and destination prefixes and not the port or protocol numbers.

Procedure

1. From the global configuration mode, create a port object group, and commit your configuration.

   Example

   ```
   Router(config)# object-group port portobj1
   Router(config-object-group-ipv4)# description my-port-object
   Router(config-object-group-ipv4)# eq bgp
   Router(config-object-group-ipv4)# range 100 200
   Router(config-object-group-ipv4)# commit
   Router(config-object-group-ipv4)# exit
   ```

2. Create an access list referencing the object group.

   Example

   ```
   Router(config)# ipv4 access-list port-object-acl permit ipv4 net-group portobj1
   ```

3. Apply the access list containing the object group to the desired interface and commit your configuration.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/10/3
   Router(config-if)# ipv4 address 2.2.2.2/24
   Router(config-if)# ipv4 access-group port-obj-acl ingress compress level 2
   Router(config-if)# no shut
   Router(config-if)# commit
   Tue Mar 28 10:23:34.106 IST

   Router(config-if)# interface HundredGigE 0/0/10/3, changed state to Down
   Router(config-if)# interface HundredGigE 0/0/10/3, changed state to Up

   Router(config-if)# exit
   ```

Confirm your configuration.

```
Router(config)# show run
Tue Mar 28 10:37:55.737 IST

Building configuration...
!! IOS XR Configuration 0.0.0
...
```

```
object-group port portobj1
 eq bgp
 range 100 200
!
ipv4 access-list port-object-acl
 10 permit tcp net-group portobj1
!
interface HundredGigE 0/0/10/3
 ipv4 access-group port-obj-acl ingress compress level 2
!
end
!
```

You have successfully configured a port object-group ACL.

Verify hybrid ACL compression

This topic describes how to verify the configured object-group ACLs in operation and the compression of the ACEs in the ACL on the Cisco 8000 Series Router.

You can use the commands described in this section to verify the configured object-group ACLs in operation and the compression of the ACEs in the ACL.

> **Note**
>
> The outputs provided in this section are a standalone sample and are not related to the configurations provided in the preceding sections.

Procedure

1. Verify the entries of the ACL in operation.

   Example

   ```
   Router# show access-lists ipv4 network-object-acl hardware ingress location
   0/0/CPU0
   ipv4 access-list network-object-acl
   40 permit ospf net-group n_192.168.0.0_16 any (20898463272 matches)
   70 permit tcp any net-group TEST_ALL_V4 established
   100 permit udp net-group INTERNAL port-group KERBEROS_UDP net-group TEST_ALL_V4
   130 permit udp net-group INTERNAL port-group DNS_UDP net-group TEST_ALL_V4
   160 permit udp net-group INTERNAL port-group NTP net-group TEST_ALL_V4
   190 permit udp net-group INTERNAL port-group LDAP_UDP net-group TEST_ALL_V4
   ...
   1500 permit udp net-group VLAN60_SECURITY net-group h_192.168.77.242 port-group
    UDP_50000-50100
   1530 deny ipv4 net-group VLAN60_SECURITY any log (20891956640 matches)
   ...
   ```

2. Verify the ACE compression in the ACL.

Example

```
Router# show access-lists ipv4 network-object-acl hardware ingress verify
location 0/0/CPU0
Verifying TCAM entries for network-object-acl
Please wait...



    INTF    NPU lookup  ACL # intf Total  compression Total   result
failed(Entry) TCAM entries
                type    ID  shared ACES   prefix-type Entries        ACE SEQ
 #     verified
---------- --- ------- --- ------ ------ ----------- ------- ------
------------ ------------

HundredGigE 0_0_10_3 (ifhandle: 0x1c8)

              1 IPV4      2      1    247 COMPRESSED      810 passed
                810
                                         SRC IP         2746 passed
               2746
                                         DEST IP        3413 passed
               3413
                                         SRC PORT        340 passed
                340
```

You have successfully verified the compression of ACEs within an ACL.

Note

The command `show access-lists` *`access-list-name`* `hardware ingress detail location` *`location`* displays compressed output for source and destination IP addresses when the `detail` keyword is used while attaching ACLs to interfaces.

24 bit bincode support for egress object group ACLs

This topic describes 24 bit bincode support for egress object group ACLs on the Cisco 8000 Series Router, which increases the result key size to support complex access lists that exceed the 14 bit limit.

24 bit bincode support is an Object Group (OG) or hybrid ACL functionality that

- increases the result key size for egress Object Group Access Control Lists (OG-ACLs),
- optimizes hardware programming for complex access lists that previously exceeded the 14 bit limit,
- provides configurable modes for ingress, egress, or both directions, and
- enables the OG compression ID extension in the Ternary Content Addressable Memory (TCAM).

Benefits of 24 bit bincode support for egress OG ACL

These are the benefits of 24 bit bincode support for egress OG ACL:

- The feature increases the result key size to 24 bits.
- It prevents programming failures caused by hardware key size limits.
- It allows the router to store large address and port ranges efficiently.

Configuration guidelines for 24 bit egress object group ACL support

This topic lists the configuration guidelines for 24 bit egress object group ACL support on the Cisco 8000 Series Router.

These guidelines apply for 24 bit egress object group ACL support:

- This feature is enabled on these systems:
  - Cisco Silicon One P100 ASIC-based systems
  - Cisco Silicon One K100 ASIC-based systems
  - Cisco Silicon One A100 ASIC-based systems
  - Cisco Silicon One P200 ASIC-based systems
- Ensure the router runs IOS XR Release 26.2.1 or later.
- Reload the router and the line cards to enable this feature after you've configured it.

Configure 24 bit support for egress object group ACL

This topic describes how to enable the bincode extension for the egress direction on the Cisco 8000 Series Router.

This section includes configuration steps for 24 bit support for egress object group ACL.

Procedure

1. Enter the global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter the command to enable the bincode extension for the egress direction.

   Example

   ```
   Router# hw-module profile tcam format og-compr-id-extension egress
   ```

3. Save the changes.

   Example

   ```
   Router# commit
   ```

4. Reload the router or line cards to activate the new hardware profile.

Example

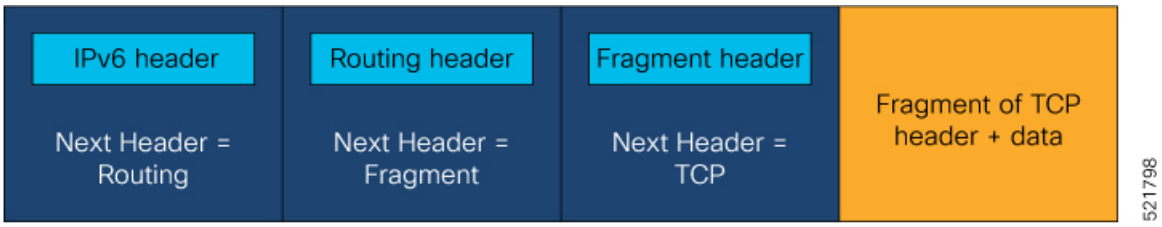
```
Router# reload
```

IPv6 extension headers in ingress IPv6 hybrid ACLs

This topic describes how ingress IPv6 hybrid ACLs with compression level 2 can filter packets on the basis of IPv6 extension headers on the Cisco 8000 Series Router.

As illustrated in the figure, an IPv6 packet may carry zero, one, or more extension headers, each identified by the Next Header field of the preceding header. To configure ingress hybrid ACLs with ACEs that permit or deny packets on the basis of IPv6 extension headers, the IPv6 extension headers must be set immediately after the base IPv6 header as shown in the figure.

Figure 13: IPv6 Extension Header in an IPv6 Packet



Therefore, you can filter ingress IPv6 packets through hybrid ACLs with compression level 2 on the basis of IPv6 extension headers set in them.

Note

You can filter ingress IPv6 packets having hop-by-hop extension headers through regular ACLs or through hybrid ACLs with compression level 2.

Feature History Table
Table 43: Feature History Table

Feature Name	Release Information	Description
IPv6 Extension Headers in Hybrid ACLs	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on Cisco 88-LC1-48Y8H-EM line cards.
IPv6 Extension Headers in Hybrid ACLs	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

Feature Name	Release Information	Description
IPv6 Extension Headers in Hybrid ACLs	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
IPv6 Extension Headers in Hybrid ACLs	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
IPv6 Extension Headers in Hybrid ACLs	Release 7.3.15	You can configure ingress IPv6 hybrid ACLs with compression level 2 to permit or deny packets on the basis of IPv6 extension headers set in them. IPv6 extension headers include routing headers, authentication headers, and destination option headers. These extension headers contain information that is used by network devices (routers and switches) to route or process an ingress IPv6 packet. The ipv6 access-group command is updated.

Configuration guidelines and restrictions for IPv6 extension headers in hybrid ACLs

This topic lists the configuration guidelines and restrictions for using IPv6 extension headers in ingress IPv6 hybrid ACLs on the Cisco 8000 Series Router.

The following restrictions apply when you configure IPv6 extension headers in ingress IPv6 hybrid ACLs:

- You cannot configure a protocol and an IPv6 extension header in the same ACE. However, you can create separate ACEs with protocols and extension header in a single ACL.
- You cannot filter egress IPv6 packets with hybrid ACLs that have compression level 2 and have IPv6 extension headers set as an ACE.
- You cannot configure custom headers or mobility headers in an ACE filter ingress IPv6 packets through hybrid ACLs with compression level 2.

Configure IPv6 extension headers in ingress IPv6 hybrid ACLs

This topic describes how to create an ingress IPv6 hybrid ACL with compression level 2 based on IPv6 extension headers on the Cisco 8000 Series Router.

Use this procedure to create an ingress IPv6 hybrid ACL with compression level 2 based on IPv6 extension headers.

Procedure

1. Enter the global configuration mode and create an IPv6 hybrid ACL with an ACE that has routing extension header.

   Example

   ```
   Router# configure
   Router(config)# ipv6 access-list ACL-EXT-HEADER
   Router(config-ipv6-acl)# 10 deny ipv6 any any routing
   Router(config-ipv6-acl)# 20 deny ospf any any
   Router(config-ipv6-acl)# commit
   Router(config-ipv6-acl)# exit
   ```

2. Enter the interface configuration mode and configure an ingress interface with the hybrid ACL with compression level 2.

   Example

   ```
   Router(config)# interface hundredGigE 0/4/0/36
   Router(config-if)# ipv6 access-group ACL-EXT-HEADER ingress compress level 2
   Router(config-if)# commit
   ```

   You have successfully configured IPv6 extension headers in an ingress IPv6 hybrid ACL.

9 Network protocol and interface integration

Topics:

This chapter describes how to integrate ACLs with network protocols and interfaces on the Cisco 8000 Series Router, including use in class maps, inline action modifications, bridge virtual interfaces, Layer 2 interfaces, ACL-based forwarding, and internal VRF-based forwarding.

This chapter contains the following sections:

IPv4 and IPv6 ACL in class map

This topic describes the enhanced Quality of Service (QoS) support for using IPv4 and IPv6 access control lists (ACLs) in class maps on the Cisco 8000 Series Router.

Quality of Service (QoS) features are enhanced to support these:

- Support on L3 interface, sub-interface, bundle interface, and bundle sub-interface
- Support for only ingress direction
- IPv6-supported match fields:
  - Destination Port
  - Fragment bit
  - ICMP type and code
  - IGMP type and code
  - IPv6 Destination Address
  - IPv6 Source Address
  - IPv6 Protocol
  - Precedence/DSCP
  - Source Port

Configure IPv6 ACL QoS

This topic describes how to configure IPv6 ACL QoS along with an IPv4 ACL and other match fields.

Use this procedure to configure IPv6 ACL QoS by matching packets against an IPv6 ACL, an IPv4 ACL, and a CoS value in a class-map, and then applying a QoS action through a policy-map.

Procedure

1. Create an IPv6 access list `aclv6` and add ACEs to permit the required IPv6 traffic.

   Example

   ```
   ipv6 access-list aclv6
   10 permit ipv6 1111:6666::2/64 1111:7777::2/64 authen
   30 permit tcp host 1111:4444::2 eq 100 host 1111:5555::2
   !
   ```

2. Create an IPv4 access list `aclv4` and add an ACE to permit the required IPv4 traffic.

   Example

   ```
   ipv4 access-list aclv4
   10 permit ipv4 host 10.6.10.2 host 10.7.10.2
   !
   ```

3. Create a class-map `c.aclv6` using the **match-any** keyword to match packets against the IPv6 ACL, the IPv4 ACL, and a CoS value.

   Example

   ```
   class-map match-any c.aclv6
   match access-group ipv6 aclv6
   match access-group ipv4 aclv4
   match cos 1
   end-class-map
   !
   ```

4. Create a policy-map `p.aclv6` that sets the IP precedence for traffic matching the class-map, and include the default class.

   Example

   ```
   policy-map p.aclv6
   class c.aclv6
     set precedence 3
   !
   class class-default
   !
   end-policy-map
   !
   ```

   You have successfully configured IPv6 ACL QoS with an IPv4 ACL and CoS match fields.

Inline ACL action modifications

This topic explains how supported Cisco 8000 Series Routers update ACL actions without repositioning hardware entries and describes update behavior, forwarding continuity, scope, restrictions, and verification.

An inline access control list (ACL) action modification is a hardware update method that:

- Changes supported action fields on an existing access control entry without creating or deleting that entry.
- Preserves the programmed match fields and avoids hardware entry repositioning.
- Keeps the entry present in hardware while the action changes.

Feature History Table
Table 44: Feature History Table

Feature name	Release information	Feature description
Inline ACL action modifications	Release 26.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]; 8700 [ASIC: K100, P100], 8200 [ASIC: P100]); Modular Systems (8800) (select variants only*) Adds inline hardware updates for supported action-only changes to existing ACL entries. Match-field changes continue to use the existing hardware update method. The update method is selected automatically and does not add a configuration command. *This feature is supported on: • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8712-MOD-M • 8711-48Z-M • 8212-48FH-M • 8711-32FH-M • 8223-64EF-M(O) Line cards: • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Key characteristics of inline ACL action modifications

- The router selects the update method automatically. An action-only change uses the inline method.
- A match-field change or a change that includes match fields uses the existing hardware update method.
- The inline method applies to supported action changes such as permit or deny, policer settings, and Differentiated Services Code Point (DSCP) remarking.
- The action change does not require a different ACL configuration.

ACL update behavior

This table compares action-only changes with changes that include match fields.

Table 45: ACL update methods

Change type	Hardware update	Forwarding behavior
Action-only change	Updates the supported action fields on the existing hardware entry without repositioning the entry.	Keeps the access control entry present while the action changes.
Match-field change or combined match-field and action change	Uses the existing hardware update method.	Preserves the existing behavior for match-field updates.

Verify inline ACL action modifications

This topic verifies that an updated ACL entry shows the expected action in ingress hardware and that the hardware programming check completes for the specified node.

Use these checks after you commit an action-only change to an existing ACL entry. The commands use IPv4 syntax. For an IPv6 ACL, replace the **ipv4** keyword with **ipv6**.

Before you begin, identify the ACL name and the node where the ACL is programmed.

Procedure

1. Display detailed ingress hardware information for the ACL.

   Example

   ```
   Router# show access-lists ipv4 <acl-name> hardware ingress detail location
   <node-id>
   ```

   Confirm that the modified access control entry shows the expected action.

2. Verify the ingress hardware programming for the ACL.

   Example

   ```
   Router# show access-lists ipv4 <acl-name> hardware ingress verify location
   <node-id>
   ```

   Confirm that the verification completes without a hardware programming error.

   The hardware entry displays the updated action, and the hardware verification completes without an error.

ACLs on bridge virtual interfaces

This topic describes how access control lists (ACLs) on a Bridge Virtual Interface (BVI) filter traffic for a group of Layer 2 interfaces.

The Bridge Virtual Interface (BVI) is a virtual interface that

- acts as a bridge between the routing and bridging domains on a router

- is a logical interface that operates as a regular routed interface with an IP address, and
- enables traffic filtering through Access Control Lists (ACLs) for the network using the interface.

Feature History Table
Table 46: Feature History Table

Feature Name	Release Information	Description
ACLs on BVI	Release 26.3.1	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.
Extend support for ACLs on BVI to A100-based ASICs	Release 26.2.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100] You can now apply ACLs on Bridged Virtual Interfaces (BVIs) on A100-based ASICs. This feature allows the router to block malicious traffic that targets the router. You can apply ACLs in both ingress and egress directions on a BVI. This feature support is now extended to: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8011-4G24Y4H-I
Extend support for ACLs on BVI to K100-based ASICs	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]); Modular Systems (8800 [LC ASIC: K100]) You can now apply ACLs on Bridged Virtual Interfaces (BVIs) on K100-based ASICs. This feature allows the router to block malicious traffic that targets the router. You can apply ACLs in both ingress and egress directions on a BVI. This feature support is now extended to: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D • 8011-4G24Y4H-I

Feature Name	Release Information	Description
Extend support for ACLs on BVI to P100-based ASICs	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100]); Fixed Systems (8700 [ASIC: P100]); Modular Systems (8800 [LC ASIC: P100]) You can now apply ACLs on Bridged Virtual Interfaces (BVIs) on P100-based ASICs. This feature allows the router to block malicious traffic that targets the router. You can apply ACLs in both ingress and egress directions on a BVI. This feature support is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E • 88-LC1-36EH
ACLs on BVI	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100])(select variants only*) This feature is now supported on: • 8011-4G24Y4H-I • 8712-MOD-M
ACLs on BVI	Release 7.3.1	This feature allows traffic filtering by configuring ACLs on Bridge Virtual Interfaces (BVIs). A single configuration can be applied for multiple interfaces that are part of the BVI. You can therefore, filter traffic for a group of interfaces with a particular purpose.

Note

The Cisco 8010 Series Routers do not support this feature. For a list of supported features on the Cisco 8010 Series Routers, see Compatibility Matrix for Cisco 8010 Series Routers.

Increased TCAM consumption with configuring ACLs on BVIs

The consumption of TCAM resources is impacted in the following manner when ACLs are configured on BVIs:

- When an ACL is attached to a BVI interface, TCAM entries are programmed on all line cards regardless of physical interface membership. This process leads to greater consumption of TCAM resources even on line cards that do not have BVI member interfaces.
- For ingress and egress ACLs, the TCAM entries for the same ACL are shared across interfaces on the same NPU.

Restrictions for configuring ACLs on BVIs

This topic lists the restrictions to consider before you configure access control lists (ACLs) on Bridge Virtual Interfaces (BVIs).

These restrictions apply to ACLs on BVI:

- When you apply a security ACL on a BVI interface, TCAMs are programmed on all NPUs of all line cards. On a distributed system, ingress ACL stats on a BVI interface are seen on the line card where the L2 interface resides. Egress ACL stats are seen on the line card of the L3 interface.
- ACLs on BVI are not supported on Cisco 8011-4G24Y4H-I router.

Configure ACLs on bridge virtual interfaces

This topic describes how to configure IPv4 ingress and egress access control lists (ACLs) on a Bridge Virtual Interface (BVI) to filter traffic for a group of Layer 2 interfaces on the Cisco 8000 Series Router.

Use this procedure to configure IPv4 ingress and egress ACLs on a Bridge Virtual Interface (BVI) and attach them to a bridge domain.

Procedure

1. Enter the global configuration mode and configure an IPv4 ingress ACL.

   Example

   ```
   Router(config)# ipv4 access-list v4-acl-ingress
   Router(config-ipv4-acl)# 10 permit tcp any 10.1.1.0/24 dscp cs6
   Router(config-ipv4-acl)# 20 deny udp any any eq ssh
   Router(config-ipv4-acl)# 30 permit ipv4 any any
   Router(config-ipv4-acl)# commit
   Router(config-ipv4-acl)# exit
   ```

2. Configure an IPv4 egress ACL.

   Example

   ```
   Router(config)# ipv4 access-list v4-acl-egress
   Router(config-ipv4-acl)# 10 deny ipv4 any any fragments log
   Router(config-ipv4-acl)# 20 deny tcp any any ack
   Router(config-ipv4-acl)# 30 permit ipv4 any any
   Router(config-ipv4-acl)# commit
   Router(config-ipv4-acl)# exit
   ```

3. Configure the HundredGigE interface that you attach to the BVI, and enable it for Layer 2 transport.

Example

```
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# l2transport
Router(config-if-l2)# commit
```

4. Attach the ingress and egress ACLs to the BVI.

Example

```
Router(config)# interface BVI1
Router(config-if)# ipv4 access-group v4-acl-ingress ingress
Router(config-if)# ipv4 access-group v4-acl-egress egress
Router(config-if)# commit
Router(config-if)# exit
```

5. Configure the bridge domain with the HundredGigE interface and BVI.

Example

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain B1
Router(config-l2vpn-bg-bd)# interface HundredGigE 0/0/0/0
Router(config-l2vpn-bg-bd-ac)# routed interface BVI1
Router(config-l2vpn-bg-bd)# commit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)# exit
```

6. Confirm that you have successfully committed the configuration.

Example

```
Router(config)# show run
...
!
ipv4 access-list v4-acl-egress
 10 deny ipv4 any any fragments log
 20 deny tcp any any ack
 30 permit ipv4 any any
!
ipv4 access-list v4-acl-ingress
 10 permit tcp any 10.1.1.0/24 dscp cs6
 20 deny udp any any eq ssh
 30 permit ipv4 any any
!
interface HundredGigE 0/0/0/0
 l2transport
 !
!
interface BVI1
 ipv4 address 209.165.200.224/27
 ipv4 access-group v4-acl-ingress ingress
 ipv4 access-group v4-acl-egress egress
!
l2vpn
 bridge group BG1
```

```
  bridge-domain B1
   interface HundredGigE 0/0/0/0
   !
   routed interface BVI1
  !
 !
!
end
```

7. Exit to the Executive Privileged mode and confirm that the ACLs are in operation.

   Example

```
Router# show access-lists interface bvi1
Tue May 9 10:01:25.732 EDT
Input ACL (common): HundredGigE 0/0/0/0 (interface): v4-acl-ingress
Output ACL: v4-acl-egress

Router# show access-lists summary
Tue May 9 10:02:01.167 EDT
ACL Summary:
Total ACLs configured: 2
Total ACEs configured: 6

Router# show access-lists ipv4 v4-acl-egress hardware egress location 0/0/CPU0
ipv4 access-list v4-acl-egress
10 deny ipv4 any any fragments log (15214 matches)
20 deny tcp any any ack (15214 matches)
30 permit ipv4 any any (15214 matches)
```

   The output clearly shows the configured ACLs, the total number of ACEs (three per ACL), and also the ACE matches in hardware.

   You have successfully configured and enabled IPv4 ingress and egress ACL on a BVI.

IPv4 and IPv6 ACLs in Layer 2

This topic describes how IPv4 and IPv6 access control lists (ACLs) operate on Layer 2 physical and bundle main interfaces on the Cisco 8000 Series Router to filter traffic at the data link layer.

Layer 2 ACLs are ethernet access control lists that operate at the data link layer of your network and filter traffic based on MAC addresses, VLAN tags, Ethernet type fields, and user or port-based authentication.

IPv4 and IPv6 ACLs are the Layer 3 access control lists that filter traffic based on IP addresses (IPv4 and IPv6) and other Layer 3 protocol information, such as protocol type, port numbers, and additional flags or control bits in the TCP header.

Feature History Table
Table 47: Feature History Table

Feature Name	Release Information	Description
IPv4 and IPv6 ACLs in Layer 2	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on: • 88-LC1-48Y8H-EM
IPv4 and IPv6 ACLs in Layer 2	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
IPv4 and IPv6 ACLs in Layer 2	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 and IPv6 ACLs in Layer 2	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 supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Description
IPv4 and IPv6 ACLs in Layer 2	Release 24.2.11	You can now configure both IPv4 and IPv6 ACLs on Layer 2 interfaces. This functionality is supported on the physical and bundle main Layer 2 interfaces, enabling Layer 3 ACLs. With this feature, you can implement traffic filtering at Layer 2, effectively preventing undesired traffic from progressing deeper into the network, like using an IPv6 ACL as an IPv6 router advertisement (RA) guard. Previously, IPv6 and IPv4 ACLs were not supported on Layer 2 interface.

IPv4 and IPv6 ACLs in Layer 2 interface

You can configure both IPv4 and IPv6 ACLs on the physical and bundle main Layer 2 interfaces. Additionally, you can enable Layer 2 (Ethernet) ACLs on the same interface simultaneously.

Advantages of IPv4 and IPv6 ACLs in Layer 2

- IPv4 and IPv6 ACLs can control which devices can communicate with each other on the same VLAN or between VLANs, thus segmenting and isolating traffic for security purposes.
- IP-based (IPv4 or IPv6) filtering at Layer 2 helps prevent unauthorized access to network resources.
- IP address-based ACLs can limit unnecessary broadcast traffic to specific segments of the network, thereby reducing congestion and improving overall network performance.
- IPv4 and IPv6 ACLs provide protocol-based traffic control, such as TCP, UDP, ICMP, and others.

Restrictions for IPv4 and IPv6 ACLs in Layer 2

This topic lists the restrictions to consider before you configure IPv4 and IPv6 access control lists (ACLs) on Layer 2 interfaces on the Cisco 8000 Series Router.

The Cisco 8000 Series Router supports IPv4 and IPv6 ACLs on Layer 2 interfaces only in the ingress direction.

Layer 2 interfaces support IPv4 and IPv6 ACLs on the following interface types:

- Physical main interfaces
- Bundle main interfaces

The Cisco 8000 Series Router does not support IPv4 and IPv6 ACLs on Layer 2 interfaces after you configure ACL permit statistics collection using the **hw-module profile stats acl-permit** command. Disable ACL permit statistics collection using the **no hw-module profile stats acl-permit** command before you configure IPv4 and IPv6 ACLs. You must reload the router after using the **no hw-module profile stats acl-permit** command.

Configure IPv4 and IPv6 ACL on Layer 2 interface

This topic describes how to configure IPv4, IPv6, and Layer 2 (Ethernet) access control lists (ACLs) on a Layer 2 physical interface on the Cisco 8000 Series Router, with the IPv6 ACL configured to act as an IPv6 router advertisement (RA) guard.

Before you begin

Ensure that the following prerequisites are met:

- Disable ACL statistics collection using the **no hw-module profile stats acl-permit** command.
- Identify the Layer 2 physical main interface on the router. In this example: HundredGigE 0/0/0/23.
- Identify the Layer 2 ACL. In this example: ETHERNET-ACL-INGRESS.
- Identify the IPv4 ACL. In this example: V4-ACL-INGRESS.
- Identify the IPv6 ACL. In this example: V6-ACL-INGRESS.

In this example, you apply IPv4, IPv6, and Layer 2 (Ethernet) ACLs to a Layer 2 physical interface, and the IPv6 ACL is configured to act as an IPv6 RA guard.

Procedure

1. Add access control entries (ACEs) to the IPv6 access list so that it acts as an IPv6 RA guard.

   Example

   ```
   Router(config)# ipv6 access-list V6-ACL-INGRESS
   Router(config-ipv6-acl)# 30 deny icmpv6 any any router-advertisement
   Router(config-ipv6-acl)# commit
   ```

2. Apply the IPv4, IPv6, and Layer 2 (Ethernet) ACLs to the Layer 2 physical main interface in interface configuration mode.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/0/23
   Router(config-if)# ipv4 access-group V4-ACL-INGRESS ingress
   Router(config-if)# ipv6 access-group V6-ACL-INGRESS ingress
   Router(config-if)# ethernet-services access-group ETHERNET-ACL-INGRESS ingress
   Router(config-if)# commit
   ```

3. Verify the configuration.

   Example

   ```
   Router# show run interface HundredGigE 0/0/0/23
   interface HundredGigE 0/0/0/23
    l2transport
    !
    ethernet-services access-group ETHERNET-ACL-INGRESS ingress
    ipv4 access-group V4-ACL-INGRESS ingress
    ipv6 access-group V6-ACL-INGRESS ingress
   !
   ```

ACL-based forwarding

This topic describes ACL-based forwarding (ABF) on the Cisco 8000 Series Router, which routes traffic through a specified next hop instead of the path computed by routing protocols.

The ACL-based forwarding (ABF) is a network feature that

- enables routing of specific traffic through designated paths instead of the paths computed by routing protocols
- uses next-hop addresses specified in ACL configurations for forwarding packets, and
- provides service selection from multiple providers for various types of traffic such as broadcast TV over IP, IP telephony, and data.

Converged networks carry voice, video, and data. Users may need to route certain traffic through specific paths instead of using the paths computed by routing protocols. You can achieve this by specifying the next-hop address in ACL configurations, so that the configured next-hop address from the ACL is used to forward the packet towards its destination instead of routing by packet-based destination address lookup. This feature of using next hop in ACL configurations for forwarding is called ACL-based forwarding (ABF).

Feature History Table
Table 48: Feature History Table

Feature Name	Release Information	Description
Permit statistics for ACL-based Forwarding (ABF)	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on: • 88-LC1-48Y8H-EM
Permit statistics for ACL-based Forwarding (ABF)	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
Permit statistics for ACL-based Forwarding (ABF)	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
Permit statistics for ACL-based Forwarding (ABF)	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Permit statistics for ACL-based Forwarding (ABF)	Release 7.3.2	This feature enables the inclusion of permitted packet count for ABF in the hardware profile statistics, thus providing the cumulative packet count of permit and deny ACL and ABF. This information helps you plan your traffic management and distribution more effectively. Modified command: • **hw-module profile stats acl-permit**

Restrictions for ACL-based forwarding

This topic lists the restrictions to consider before you configure ACL-based forwarding (ABF) on the Cisco 8000 Series Router.

- Traffic outages can occur during transitions from an existing next hop to another next hop.
- IPv4 and IPv6 ABF next hops routed over GRE interfaces are not supported.
- VRF-select (where only the VRF is configured for the next hop) is not supported in ABF for IPv4 and IPv6 addresses in releases before Cisco IOS XR Release 7.3.3.
- Logging of permit statistics for ABF is not supported in releases before Cisco IOS XR Release 7.3.2.
- ACL-based forwarding (ABF) is not supported over BVI in releases before Cisco IOS XR Release 7.5.3.
- Enabling the **log** keyword on access control entries (ACEs) configured for ABF next-hop redirection causes the ABF redirect to fail, resulting in traffic not being forwarded to the specified next hops.

```
! Working (no logging)
permit ipv4 any 52.108.1.32/30 \
  nexthop1 ipv4 101.15.52.1 \
  nexthop2 ipv4 101.15.52.11 \
  nexthop3 ipv4 101.15.52.13

! Problematic (logging enabled – redirect fails)
permit ipv4 any 52.108.1.32/30 log \
  nexthop1 ipv4 101.15.52.1 \
```

```
  nexthop2 ipv4 101.15.52.11 \
  nexthop3 ipv4 101.15.52.13
```

ACL-based forwarding behavior and logging

This topic describes the operational behavior of ACL-based forwarding (ABF) and the logging of permit statistics for ABF traffic.

- ABF supports next-hop modifications. You can modify a next hop, remove a next hop, or make changes between existing next hops.

  Note

  While you define an ACE rule, you must specify the VRF for all next hops unless the next hop is in the default VRF. This process ensures that the packets take the right path towards the next hop.

- ABF is ACL-based. Packets that do not match an existing rule (ACE) in the ACL are subject to the default ACL rule (drop all). If the ACL is used only for ABF redirect (not for security), include an explicit ACE rule at the end of the ACL (lowest user priority) to match and permit all traffic. This ensures that all traffic that does not match an ABF rule is permitted and forwarded normally.
- ABF is supported on permit rules only.
- ABF default route is not supported.
- Packets punted in the ingress direction from the NPU to the line card CPU are not subject to ABF treatment because ABF is not supported in the slow path. These packets are forwarded normally based on destination-address lookup by the software dataplane. Examples of these types of packets include, but are not limited to, packets with IPv4 options, IPv6 extension headers, and packets destined for glean (unresolved or incomplete) adjacencies.

Logging of permit statistics for ABF

Starting with Cisco IOS XR Release 7.3.2, ABF supports logging of permit statistics. This feature tracks the number of packets that an ACL permits in the routing traffic. To enable logging of permit statistics for ABF, configure hardware module statistics for ACL before you configure ACL-based forwarding. To enable hardware module statistics for ACL, use the **hw-module profile stats acl-permit** command in XR Config mode. To disable the tracking of permitted packet count, use the **no** form of this command.

Note

After you enable hardware module statistics for ACL, you must reboot the line cards or the router based on the requirement.

Use the **show access-lists ipv4** or **show access-lists ipv6** command to view the ABF statistics.

Configure logging of permit statistics

This topic describes how to configure logging of permit statistics for ACL-based forwarding (ABF) on the Cisco 8000 Series Router by enabling the hardware module statistics profile for ACL.

Use this procedure to enable the tracking of permitted packet count for ABF traffic and to verify the ABF statistics on a selected interface.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enable the hardware module statistics profile for ACL permit by using the **hw-module profile stats acl-permit** command.

   Example

   ```
   Router(config)# hw-module profile stats acl-permit
   Fri Aug  7 05:52:58.052 UTC
   In order to activate/deactivate this stats profile, you must manually reload
    the chassis/all line cards
   ```

3. Commit the configuration.

   Example

   ```
   Router(config)# commit
   Fri Aug 7 05:55:50.103 UTC
   LC/0/4/CPU0:Aug 7 05:55:50.218 UTC: fia_driver[245]:
   %FABRIC-FIA_DRVR-4-STATS_HW_PROFILE_MISMATCH : Mismatch found, reload LC to
   activate the new stats profile
   Router(config)#
   ```

4. Reload the line cards or the chassis to activate the stats profile.
5. Verify the ABF statistics on a selected interface by using the **show access-lists ipv4** or **show access-lists ipv6** command.

   Example

   ```
   Router# show access-lists ipv4 test-acl-ipv4 hardware ingress location 0/0/CPU0
   Tue Sep 21 07:37:41.297 UTC
   ipv4 access-list test-acl-ipv4
   10 permit ipv4 any any (14950)
   ```

   You have successfully configured logging of permit statistics for ACL-based forwarding.

Configure ACL-based forwarding

This topic describes how to configure ACL-based forwarding (ABF) to route traffic through a specified next hop instead of the path computed by routing protocols.

Use this procedure to configure ACL-based forwarding (ABF) by defining IPv4 and IPv6 ACLs with next-hop addresses, and to verify the running configuration and next-hop state.

Procedure

1. Enter IPv4 access list configuration mode and configure an ACL with the required ACEs and next hops for ABF.

   Example

```
Router# configure
Router(config)# ipv4 access-list abf-acl
Router(config-ipv4-acl)# 10 permit ipv4 192.168.18.0 0.255.255.255 any nexthop1
 ipv4 192.168.20.2
Router(config-ipv4-acl)# 15 permit ipv4 192.168.21.0 0.0.0.255 any
Router(config-ipv4-acl)# 20 permit ipv4 192.168.22.0 0.0.255.255 any nexthop1
 ipv4 192.168.23.2
Router(config-ipv4-acl)# 25 permit tcp any range 2000 3000 any range 4000
5000 nexthop1 ipv4 192.168.23.1 nexthop2 ipv4 192.168.24.1 nexthop3 ipv4
192.168.25.1
Router(config-ipv4-acl)# 30 permit tcp any eq www host 192.168.12.2 precedence
 immediate nexthop1 vrf vrf1_ipv4 ipv4 192.168.13.2 nexthop2 vrf vrf1_ipv4
ipv4 192.168.14.2
Router(config-ipv4-acl)# 35 permit ipv4 any any
Router(config-ipv4-acl)# commit
```

2. Enter IPv6 access list configuration mode and configure an ACL with the required ACEs and next hops for ABF.

   Example

```
Router# configure
Router(config)# ipv6 access-list abf-acl
Router(config-ipv6-acl)# 10 permit ipv6 2001:db8::/32 any nexthop1 ipv6
2001:db8::2
Router(config-ipv6-acl)# 25 permit tcp any range 2000 3000 any range 4000
5000 nexthop1 ipv6 2001:db8::3 nexthop2 ipv6 2001:db8::4 nexthop3 ipv6
2001:db8::5
Router(config-ipv6-acl)# 30 permit tcp any eq www host 2001:db8::8 precedence
 immediate nexthop1 vrf vrf1_ipv6 ipv6 2001:db8::7 nexthop2 vrf vrf1_ipv6
ipv6 2001:db8::6
Router(config-ipv6-acl)# 35 permit ipv6 any any
Router(config-ipv6-acl)# commit
```

3. Verify the running configuration.

   Example

```
Router# show access-lists ipv4
ipv4 access-list abf-acl
10 permit ipv4 192.168.18.0 0.255.255.255 any nexthop1 192.168.20.2
15 permit ipv4 192.168.21.0 0.0.0.255 any
20 permit ipv4 192.168.22.0 0.0.255.255 any nexthop1 192.168.23.2
25 permit tcp any range 2000 3000 any range 4000 5000 nexthop1 ipv4
192.168.23.1 nexthop2 ipv4 192.168.24.1 nexthop3 ipv4 192.168.25.1
30 permit tcp any eq www host 192.168.12.2 precedence immediate nexthop1 vrf
 vrf1_ipv4 ipv4 192.168.13.2 nexthop2 vrf vrf1_ipv4 ipv4 192.168.14.2
35 permit ipv4 any any
!
Router# show access-lists ipv6
ipv6 access-list abf-acl-ipv6
 10 permit ipv6 2001:db8::/32 any nexthop1 ipv6 2001:db8::2
 25 permit tcp any range 2000 3000 any range 4000 5000 nexthop1 ipv6
```

```
2001:db8::3 nexthop2 ipv6 2001:db8::4 nexthop3 ipv6 2001:db8::5
 30 permit tcp any eq www host 2001:db8::8 precedence immediate nexthop1 vrf
 vrf1_ipv6 ipv6 2001:db8::7 nexthop2 vrf vrf1_ipv6 ipv6 2001:db8::6
 35 permit ipv6 any any
```

4. Verify the IP next-hop state in ABF to ensure that the expected next hop is up.

   Example

```
Router# show access-lists ipv4 abf nexthops client pfilter_ea location 0/3/CPU0
Tue May 17 22:25:05.940 UTC

 ACL name : abf-acl
  ACE seq.           NH-1                  NH-2                  NH-3
 ---------  --------------------  --------------------  --------------------

       20    Global 192.168.23.2           Not present           Not present

    status                    UP           Not present           Not present

     exist                    No           Not present           Not present

    pd ctx               Present           Not present           Not present

                Track not present      Track not present
--
        25    Global 192.168.23.1    Global 192.168.24.1        Global
192.168.25.1
    status                     UP                    UP            UP
     exist                    Yes                   Yes           Yes
    pd ctx                Present               Present           Present
              Track not present      Track not present           Track not
 present
```

5. Verify whether ABF is attached to any interfaces on any line card.

   Example

```
Router# show access-lists usage pfilter location all
sh access-lists ipv4 abf nexthops client pfilter_ea loc 0/RP0/CPU0
Wed Jul 29 20:48:18.559 UTC

ACL name : abf-1
  ACE seq.        NH-1             NH-2             NH-3
 ---------  ---------------  ---------------  ---------------
        10    27.138.216.32         28.0.0.2      Not present
    status               UP               UP      Not present
at status       Not Present      Not Present      Not present
     exist               No              Yes      Not present
       vrf          default          default      Not present
     track      Not present      Not present      Not present
    pd ctx          Present          Present      Not present
```

Virtual Routing and Forwarding (VRF) redirect in ABF

This topic describes how ACL-based forwarding (ABF) supports Virtual Routing and Forwarding (VRF) redirect for IPv4 and IPv6 addresses on the Cisco 8000 Series Router.

The ACL-based forwarding (ABF) with VRF redirect is a network feature

- supports forwarding incoming IPv4 and IPv6 traffic to virtual routing and forwarding (VRF) instances by using VRF as the next hop
- enables two types of VRF redirect–VRF-select and VRF-aware, and
- supports up to three next hops per access control entry (ACE).

Feature History Table
Table 49: Feature History Table

Feature Name	Release Information	Description
Virtual Routing and Forwarding (VRF) redirect in ACL-based Forwarding (ABF)	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100])(select variants only*) This feature is now supported on Cisco 8712-MOD-M routers.
Virtual Routing and Forwarding (VRF) redirect in ACL-based Forwarding (ABF)	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Description
Virtual Routing and Forwarding (VRF) redirect in ACL-based Forwarding (ABF)	Release 7.3.3	With this feature, ABF supports VRF redirect for IPv4 and IPv6 addresses. You can enable this feature by configuring a VRF as the next hop in ABF ACEs. ABF supports two types of VRF redirect: • VRF-select • VRF-aware ABF with VRF redirect enables you to inject incoming traffic matching ACEs in the ABF to a VRF network and helps with load balancing and cost saving.

How VRF redirect in ABF works

This topic describes how the router processes incoming packets that match a permit ACE with VRF redirect in ACL-based forwarding (ABF).

ACL-based forwarding (ABF) supports VRF redirect for IPv4 and IPv6 addresses. When you configure a VRF as the next hop in an ABF ACE, the router forwards the matching traffic to the VRF for further route lookup instead of using the global routing table.

Summary

The following components take part in this process:

- Router: Evaluates incoming packets against the ABF ACL and forwards the matching traffic to the specified VRF next hop.
- Permit ACE with VRF redirect: Specifies up to three VRFs as next hops for matching traffic.
- VRF: Provides a separate routing instance in which the router performs route lookup for the redirected traffic.
- Global routing table: Provides the fallback route lookup when a default route in the VRF redirects the traffic back.

These stages describe how the router forwards packets that match a permit ACE with VRF redirect and how it handles the outcome of the VRF route lookup.

Workflow

These stages describe how VRF redirect in ABF processes an incoming packet:

1. The router matches the incoming packet against the ABF ACL and identifies a permit ACE with VRF redirect.
2. The router forwards the matching traffic to the first available VRF specified in the ACE as the next hop.
3. The router performs a route lookup for the redirected traffic in the selected VRF and handles the outcome as follows:

When...	Then...
The redirected traffic matches a route available in the VRF.	The router forwards the traffic by using the matched route in the VRF.
The redirected traffic fails to match any route available in the VRF.	A default route redirects the traffic back to the global routing table for lookup and forwarding.

When...	Then...
The VRFs in any ACEs with VRF are down.	The router drops the traffic matching such ACEs immediately.

Configure ABF with VRF-select

This topic describes how to configure ACL-based forwarding (ABF) with VRF-select, where only a VRF is configured as the next hop.

Use this procedure to configure ABF with VRF-select by adding ACEs that redirect matched traffic to one or more VRFs as next hops.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create an IPv4 access list `abf-vrf-select`.

   Example

   ```
   Router(config)# ipv4 access-list abf-vrf-select
   ```

3. Add ACEs that redirect matched traffic to one or more VRFs as next hops.

   Example

   ```
   Router(config-ipv4-acl)# 10 permit ipv4 60.1.1.5 0.0.0.255 any nexthop1 vrf
   VRF1 nexthop2 vrf VRF2 nexthop3 vrf VRF3
   Router(config-ipv4-acl)# 11 permit tcp 30.0.10.1 0.0.0.255 30.0.20.1 0.0.0.255
    nexthop1 vrf vrf3 nexthop2 vrf vrf2
   Router(config-ipv4-acl)# 12 permit tcp host 30.0.10.1 host 30.0.20.1 match-all
    +ack +psh +syn nexthop1 vrf vrf2 nexthop2 vrf vrf3 nexthop3 vrf vrf1
   Router(config-ipv4-acl)# 13 permit tcp host 30.0.10.1 host 30.0.20.1 precedence
    priority nexthop1 vrf vrf1 nexthop2 vrf vrf2 nexthop3 vrf vrf3
   ```

4. Commit the configuration.

   Example

   ```
   Router(config-ipv4-acl)# commit
   ```

   You have successfully configured ABF with VRF-select.

5. Verify the ABF-VRF configurations by using the **show access-lists ipv4** command for `abf-vrf-select` access lists.

Example

```
Router# show access-lists ipv4 abf-vrf-select
10 permit ipv4 60.1.1.5 0.0.0.255 any nexthop1 vrf VRF1 nexthop2 vrf VRF2
nexthop3 vrf VRF3
11 permit tcp 30.0.10.1 0.0.0.255 30.0.20.1 0.0.0.255 dscp af22 nexthop1 vrf
 vrf3 nexthop2 vrf vrf2
12 permit tcp host 30.0.10.1 host 30.0.20.1 match-all +ack +psh +syn set
nexthop1 vrf vrf2 nexthop2 vrf vrf3 nexthop3 vrf vrf1
13 permit tcp host 30.0.10.1 host 30.0.20.1 precedence priority nexthop1 vrf
 vrf1 nexthop2 vrf vrf2 nexthop3 vrf vrf3
```

You have successfully configured ABF with VRF-select and verified the ABF-VRF configurations on the router.

Configure ABF with VRF-aware

This topic describes how to configure ACL-based forwarding (ABF) with VRF-aware, where both a VRF and an IPv4 address are configured as the next hop.

Use this procedure to configure ABF with VRF-aware by adding ACEs that redirect matched traffic to one or more VRF and IPv4 next-hop combinations.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create an IPv4 access list `abf-vrf-aware`.

   Example

   ```
   Router(config)# ipv4 access-list abf-vrf-aware
   ```

3. Add ACEs that redirect matched traffic to one or more VRF and IPv4 address combinations as next hops.

   Example

   ```
   Router(config-ipv4-acl)# 10 permit tcp net-group network_object_group_host
   net-group network_object_group_host precedence priority nexthop1 vrf vrf1
   ipv4 40.1.5.2 nexthop2 vrf vrf2 ipv4 50.1.5.2 nexthop3 vrf vrf3 ipv4 60.1.5.2
   Router(config-ipv4-acl)# 11 permit tcp 30.0.10.1 0.0.0.255 net-group
   network_object_group dscp af22 nexthop1 vrf vrf3 ipv4 60.1.5.2 nexthop2 vrf
   vrf2 ipv4 50.1.5.2
   ```

4. Commit the configuration.

   Example

   ```
   Router(config-ipv4-acl)# commit
   ```

5. Verify the ABF-VRF configurations by using the **show access-lists ipv4** command for `abf-vrf-aware` access lists.

Example

```
Router# show access-lists ipv4 abf-vrf-aware
10 permit tcp net-group network_object_group_host net-group
network_object_group_host precedence priority nexthop1 vrf vrf1 ipv4 40.1.5.2
 nexthop2 vrf vrf2 ipv4 50.1.5.2 nexthop3 vrf vrf3 ipv4 60.1.5.2
11 permit tcp 30.0.10.1 0.0.0.255 net-group network_object_group dscp af22
nexthop1 vrf vrf3 ipv4 60.1.5.2 nexthop2 vrf vrf2 ipv4 50.1.5.2
```

You have successfully configured ABF with VRF-aware and verified the ABF-VRF configurations on the router.

ABF over BVI

This topic describes ACL-based forwarding (ABF) for IPv4 and IPv6 addresses on a Bridge Virtual Interface (BVI) on the Cisco 8000 Series Router.

Starting with Cisco IOS XR Release 7.5.3, you can enable ACL-based Forwarding for IPv4 and IPv6 addresses on Bridge Virtual Interfaces (BVIs). You can configure ABF to forward incoming traffic on the BVIs that matches the ACEs.

Feature History Table
Table 50: Feature History Table

Feature Name	Release Information	Description
ACL-based Forwarding (ABF) on Bridge Virtual Interfaces (BVIs)	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on: • 88-LC1-48Y8H-EM
ACL-based Forwarding (ABF) on Bridge Virtual Interfaces (BVIs)	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
ACL-based Forwarding (ABF) on Bridge Virtual Interfaces (BVIs)	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
ACL-based Forwarding (ABF) on Bridge Virtual Interfaces (BVIs)	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700); Modular Systems (8800 [ASIC: P100]) (select variants only*) *This feature is now supported on: • 8212-32FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 88-LC1-36EH
ACL-based Forwarding (ABF) on Bridge Virtual Interfaces (BVIs)	Release 7.5.3	This feature allows using Bridge Virtual Interfaces (BVIs) as next hop in ABF. You can apply a single ABF configuration to multiple interfaces that are part of the BVI. You can therefore divert user traffic for a group of interfaces with a particular purpose.

Restrictions for configuring ABF over BVI

- ABF for IPv4 and IPv6 addresses over BVI is supported only in ingress direction.

Configure ABF over BVI

This topic describes how to configure ACL-based forwarding (ABF) for IPv4 and IPv6 addresses on a Bridge Virtual Interface (BVI).

Use this procedure to configure ACL-based forwarding (ABF) for IPv4 and IPv6 addresses on a Bridge Virtual Interface (BVI), and to verify the running configuration.

Procedure

1. Enter IPv4 or IPv6 ACL configuration mode and configure an ACL.

   Example

```
/* IPv4 */
Router# configure
Router(config)# ipv4 access-list abf-acl1
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# exit

/* IPv6 */
Router# configure
Router(config)# ipv6 access-list abf-acl2
Router(config-ipv6-acl)# commit
Router(config-ipv6-acl)# exit
```

2. Create ACEs for the ACL and configure next-hop addresses for ABF.

   Example

```
/* IPv4 */
Router(config)# ipv4 access-list abf-acl1
Router(config-ipv4-acl)# 10 permit ipv4 192.168.18.0 0.255.255.255 any nexthop1
 ipv4 192.168.20.2
Router(config-ipv4-acl)# 15 permit ipv4 192.168.21.0 0.0.0.255 any
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# exit

/* IPv6 */
Router(config)# ipv6 access-list abf-acl2
Router(config-ipv6-acl)# 10 permit ipv6 2001:db8::/32 any nexthop1 ipv6
2001:db8::2
Router(config-ipv6-acl)# commit
Router(config-ipv6-acl)# exit
```

3. Configure the HundredGigE interface for the BVI, and enable it for Layer 2 transport.

   Example

```
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# l2transport
Router(config-if-l2)# commit
```

4. Enter the BVI mode and configure ABF.

   Example

```
Router(config)# interface BVI 1
Router(config-if)# ipv4 access-group abf-acl1 ingress
Router(config-if)# ipv6 access-group abf-acl2 ingress
Router(config-if)# commit
```

5. Configure the bridge domain with the HundredGigE interface and BVI.

   Example

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain B1
Router(config-l2vpn-bg-bd)# interface HundredGigE 0/0/0/0
Router(config-l2vpn-bg-bd-ac)# routed interface BVI1
```

6. Save the configuration.

```
Router(config-l2vpn-bg-bd)# commit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)# exit
```

Internal VRF-based forwarding

This topic describes how the internal VRF-based forwarding feature uses a forwarding-match access control entry (ACE) to redirect packets that do not match predefined access control entries to an internal VRF (iVRF) for deep inspection through GRE tunneling.

Forwarding capabilities in VRFs are enhanced in the ingress direction to allow VRFs to redirect incoming packets to a different destination using GRE tunneling.

Feature history for Internal VRF-based forwarding

Feature Name	Release Information	Description
Internal VRF based Forwarding	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on: • 88-LC1-48Y8H-EM
Internal VRF based Forwarding	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
Internal VRF based Forwarding	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.
Internal VRF based Forwarding	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 supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Description
Internal VRF based Forwarding	Release 24.2.11	Forwarding capabilities in VRFs are enhanced, allowing internal VRFs (iVRF) to redirect incoming packets to a different destination using GRE tunneling. This functionality can be used to examine packets that do not match the predefined access control entries. Instead of discarding these packets by default, a forwarding-match ACE sends them to a VRF that can forward them using GRE tunnels. This allows for a more thorough inspection of these discarded packets, helping to identify any hidden threats or attacks in the contents and improving network security.

Configure VRF-based forwarding

This topic describes how to configure internal VRF-based forwarding by using a forwarding-match ACE to redirect unmatched packets through a GRE tunnel to a destination for deep inspection.

In this example, a forwarding-match ACE is used for packets that do not match the predefined access control entries. These packets are then forwarded to a specific destination using GRE tunnels, which allow for deep inspection to better understand any underlying threats.

Procedure

1. Create an ingress VRF using the **vrf** command.

   Example

   ```
   Router(config)# vrf ivrf
   Router(config)# commit
   ```

2. Configure the ingress ACLs to send the discarded packets to the ingress VRF (iVRF) by using the **ipv4 access-list** and **permit** commands.

   Example

   ```
   Router(config)# ipv4 access-list V4-ACL-INGRESS
   Router(config-ipv4-acl)# 10 permit tcp 192.0.2.6 255.255.255.0 any
   Router(config-ipv4-acl)# 20 permit ipv4 any any nexthop1 vrf iVRF
   Router(config-ipv4-acl)# commit
   ```

   In this example, the entry **20 permit ipv4 any any nexthop1 vrf iVRF** is a forwarding-match ACE for packets that do not match the predefined access control entries.

3. Apply the ingress ACL to an interface by using the **ipv4 access-group** command.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/0/0
   Router(config-if)# ipv4 access-group V4-ACL-INGRESS
   Router(config-if)# commit
   ```

4. Create a GRE tunnel in the iVRF to the desired destination by using the **interface**, **vrf**, **tunnel mode**, **tunnel source**, and **tunnel destination** commands.

   Example

```
Router(config)# interface tunnel-ip1
Router(config-if)# vrf iVRF
Router(config-if)# ipv4 address 198.51.100.7 255.255.255.0
Router(config-if)# tunnel mode gre ipv4 encap
Router(config-if)# tunnel source 198.51.100.19
Router(config-if)# tunnel destination 203.0.113.1
Router(config-if)# commit
```

   For more information on GRE tunnels, see the *Configure GRE Tunnel* chapter in the *Interface and Hardware Component Configuration Guide for Cisco 8000 Series Routers*.

5. Add static routes in the iVRF to redirect traffic to the GRE tunnels by using the **router static**, **vrf**, and **address-family** commands.

   Example

```
Router(config)# router static
Router(config-static)# vrf iVRF
Router(config-static-vrf)# address-family ipv4 unicast
Router(config-static-vrf-afi)# 192.0.2.9/32 tunnel-ip1
Router(config-static-vrf-afi)# commit
```

6. Add rules to import BGP routes matching the community rules to the iVRF.

   Example

```
Router(config)# vrf iVRF
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-afi)# import from default-vrf route-policy COMMUNITY1
Router(config-vrf-afi)# exit
Router(config-vrf)# community-set 2000
Router(config-vrf-comm)# 2000:2000
Router(config-vrf-comm)# end set
Router(config-vrf)# commit
```

7. Rewrite the next-hop for community-matching rules to the route pointing to the GRE tunnel.

   Example

```
Router(config)# vrf iVRF
Router(config-vrf)# route-policy COMMUNITY1
Router(config-vrf-rpl)# if community in 2000 then
Router(config-vrf-rpl-if)# set next-hop 192.0.2.9 destination-vrf
Router(config-vrf-rpl-if)# end if
Router(config-vrf-rpl)# end policy
Router(config-vrf)# commit
```

10 Diagnostic, statistics, and policy control

Topics:

This chapter describes diagnostic and statistics features for ACLs on the Cisco 8000 Series Router, including user-defined TCAM keys, ACL counters and statistics, fragment control, packet length filtering, TTL matching, logging, and per-interface statistics.

This chapter contains the following sections:

User-defined TCAM keys for IPv4 and IPv6

This topic describes User-Defined TCAM Keys (UDKs) that provide flexibility to define a custom TCAM key for ingress IPv4 and IPv6 ACLs.

The TCAM (internal and external) Key Definition is a configuration component that

- specifies qualifier and action fields for packet lookups
- optimizes the utilization of constrained hardware TCAM space, and
- determines the available key width for ACL operations.

The key definitions are specific to a given ACL type, which can depend on the following attributes of the access list:

- Direction of attachment only for ingress
- Protocol type (IPv4/IPv6)

Because the default key definitions are constrained (do not include all qualifier/action fields), User-Defined Key (UDK) definitions are supported for the following types:

- Traditional Ingress IPv4 ACL (uncompressed)
- Traditional Ingress IPv6 ACL (uncompressed)

The User-Defined TCAM Key (UDK) functionality provides the flexibility to define your own TCAM key for ingress, traditional, or IPv4 and IPv6 ACL only. To include the well-known fields in the default TCAM key, see IPv4 and IPv6 key formats on page 241.

A User-Defined TCAM Key (UDK) can be defined globally or locally per line card. If both global and local UDK is available for a line card, then global UDK is ignored for the line card.

Configure UDK

A UDK can be configured using the following command:

```
hw-module profile tcam format access-list [ipv4 | ipv6] field1 field2[location
 rack/slot/cpu0]
```

To define UDK globally, you can ignore the location option.

```
hw-module profile tcam format access-list [ipv4 | ipv6] field1 field2
```

Note

Use global UDK for Cisco 8000 Series Routers Fixed platform.

IPv4 and IPv6 key formats

This topic describes the qualifier and action fields supported in the default IPv4 and IPv6 TCAM key formats.

Qualifier fields

The following table shows the qualifier fields that are supported in the IPv4 and IPv6 key formats.

Note

You cannot configure destination address and destination object group together for an ACL. Similarly, you cannot configure source address and source object group together for an ACL.

Table 51: Qualifier Fields Supported in IPv4 and IPv6 Key Formats

Parameter	Default TCAM Key	
	IPv4	IPv6
Destination Address	Supported	Supported
Destination Object Group	Supported	Supported
Destination Port	Supported	Supported
Fragment bit	Supported	Supported
Fragment offset	Supported	Not Supported
Fragment type	Supported	Not Supported
ICMP type and code	Supported	Supported
IGMP type and code	Supported	Supported
Packet Length	Supported	Supported
Protocol/Next Header	Supported	Supported
Precedence/DSCP	Supported	Supported
Source Object Group	Supported	Supported
Source Address	Supported	Supported
Source Port	Supported	Supported
TCP Flags	Supported	Supported for Ingress only
Time to live (TTL) Match	Supported	Not supported
UDF 1-8	Not supported	Not supported

> **Note**
>
> IGMP header match for IPv4 in v2 and v3 reports is not supported.

Action fields

The following table shows the action fields supported in the IPv4 and IPv6 key formats.

Table 52: Action Fields Supported in IPv4 and IPv6 Key Formats

Parameter	Default Action Field	
	IPv4	IPv6
Permit	Supported	Supported
Deny	Supported	Supported
Next Hop	Supported	Supported
Log	Supported for Ingress only	Supported for Ingress only
Capture	Supports only ingress Encapsulated Remote SPAN (ERSPAN)	Supports only ingress Encapsulated Remote Switch port Analyzer (ERSPAN)
Stats Counter	Deny stats is always enabled (permit stats is enabled by the **hw-module profile stats acl-permit** command)	Deny stats is always enabled (permit stats is enabled by the **hw-module profile stats acl-permit** command)

Access control list counters

This topic describes how access control list (ACL) counters are maintained in hardware and software on the Cisco 8000 Series Router.

In Cisco IOS XR software, ACL counters are maintained both in hardware and software. Hardware counters are used for packet filtering applications. Software counters are used by all the applications mainly involving software packet processing.

Software counters are updated for the packets processed in software, for example, exception packets punted to the LC CPU for processing, or ACL used by routing protocols, and so on. The counters that are maintained are an aggregate of all the software applications using that ACL. To display software-only ACL counters, use the **show access-lists ipv4** *access-list-name* [**sequence** *number*] command in EXEC mode.

ACL statistics counter

This topic describes how the ACL statistics counter feature on the Cisco 8000 Series Router tracks the count of packets that a router permits or denies based on the ACL rules configured on an interface.

The ACL statistics counter is an ACL monitoring feature that

- tracks the quantity of packets the router permits or denies
- facilitates ACL-based traffic mirroring, and
- identifies specific ACE matches for permitted traffic.

By default, the ACL statistics counter allows you to track only the count of packets that are denied. By configuring the **hw-module profile stats acl-permit** command, you can also track the count of packets that are permitted. Routers use this knowledge of the count of packets for ACL-based traffic mirroring.

Restrictions for the ACL statistics counter

This topic lists the restrictions to consider before you enable the ACL statistics counter for tracking permitted packet counts.

These restrictions apply for the ACL statistics counter:

- After you configure the **hw-module profile stats acl-permit** command on the router, you must reload the router or the line cards based on the requirement. Configuring the command followed by reloading the router or line cards enables tracking of the permitted packet count on the router or line cards.
- Starting with Cisco IOS XR Release 26.2.1, MPLS-to-IP packet-accounting counters may not update when `hw-module profile stats acl-permit` is enabled. If MPLS-to-IP packet accounting is required, do not enable ACL permit statistics.
- To restore MPLS-to-IP packet accounting, remove the ACL permit statistics profile and reload the affected line cards or router.
- If the hardware module statistics for ACL (**hw-module profile stats acl-permit**) is enabled, the flowspec drop counter functionality does not work for the NP1-based platforms.
- ACL named counter configuration is not supported. Although configurations with named counters are accepted, you cannot query counters by their names. Starting with Cisco IOS XR Release 25.4.1, a log message is displayed similar to `pfilter_ea[225]: %PKT_INFRA-DPA_FM-4-ACL_NAMED_COUNTER_NOT_SUPPORTED : ACL Named counter is not supported`.

Configure the ACL statistics counter

This topic describes how to enable tracking of the permitted packet count based on the ACL rules by configuring the hardware module statistics profile for ACL permit.

Use this procedure to configure an ACL, attach it to an interface, and enable tracking of the permitted packet count on the router.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Configure an ACL by using the **ipv4 access-list** *acl-name* command and add the required access control entries.

Example

```
Router(config)# ipv4 access-list TEST
Router(config-ipv4-acl)# 10 permit ipv4 any any
Router(config-ipv4-acl)# 20 deny udp any any
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# exit
```

3. Enter interface configuration mode and attach the configured ACL to the interface by using the **ipv4 access-group** *acl-name* **ingress** command.

Example

```
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv4 access-group TEST ingress
Router(config-if)# commit
Router(config-if)# exit
```

4. Enable tracking of the permitted packet count by configuring the **hw-module profile stats acl-permit** command.

Example

```
Router(config)# hw-module profile stats acl-permit
Router(config)# commit
```

5. Reload the router or the line cards to activate the stats profile.

Example

```
Router# reload location 0/0/CPU0
```

6. Verify whether tracking of the permitted packet count is enabled by using the **show access-lists ipv4** *acl-name* **hardware ingress location** *node-id* command.

Example

```
Router# show access-lists ipv4 TEST hardware ingress location 0/0/CPU0
ipv4 access-list test-v4-ttl
 10 permit ipv4 any any ttl eq 100
 20 deny ipv4 any any ttl eq 200 (44043 matches)
Router# configure
Router(config)# hw-module profile stats acl-permit
In order to activate/deactivate this stats profile, you must manually reload
 the chassis/all line cards
Router(config)# commit
Router# reload location 0/0/CPU0


Router# show access-lists ipv4 TEST hardware ingress location 0/0/CPU0
ipv4 access-list test-v4-ttl
 10 permit ipv4 any any ttl eq 100 (346318 matches)
 20 deny ipv4 any any ttl eq 200 (44043 matches)
```

You have successfully configured the ACL statistics counter and verified that the router tracks the count of permitted packets based on the ACL rules.

ACLs with fragment control

This topic describes how the IP Extended Access Lists with Fragment Control feature provides granular control over non-initial IP fragments when you apply an IP extended access list.

Non-fragmented packets and the initial fragments of a packet are processed by IP extended access lists (if you apply this access list), but non-initial fragments are permitted by default.

The IP Extended Access List with Fragment Control is a configuration feature that

- provides granular filtering control over non-initial IP fragments
- applies Layer 3-specific access-list entries to non-initial packet fragments, and
- utilizes an optional fragments keyword to manage how the system evaluates packet segments.

This feature adds the optional **fragments** keyword to these IP access list commands:

- **deny**
- **permit**

If you specify the **fragments** keyword in an access-list entry, that access-list entry applies only to non-initial fragments of packets. The fragment is either permitted or denied accordingly.

Behavior of access-list entries with and without the fragments keyword

If the access-list entry has...	Then...

No **fragments** keyword and all of the access-list entry information matches	For an access-list entry containing only Layer 3 information:
	• The entry is applied to non-fragmented packets, initial fragments, and non-initial fragments.
	
	For an access-list entry containing Layer 4 information:
	• The entry is applied to non-fragmented packets and initial fragments.
	  • If the entry matches and is a **permit** statement, the packet or fragment is permitted.
	  • If the entry matches and is a **deny** statement, the packet or fragment is denied.
	• The entry is also applied to non-initial fragments in the following manner. Because non-initial fragments contain only Layer 3 information, only the Layer 3 portion of an access-list entry can be applied. If the Layer 3 portion of the access-list entry matches:
	  • If the entry is a **permit** statement, the non-initial fragment is permitted.
	  • If the entry is a **deny** statement, the next access-list entry is processed.
	
	Note
	The deny statements are handled differently for non-initial fragments versus non-fragmented or initial fragments.

The **fragments** keyword and all of the access-list entry information matches	The access-list entry is applied only to non-initial fragments. **Note** If the **fragments** keyword is configured for an access-list entry, the Layer 4 information is ignored for the non-initial fragments.

Configuration guidelines and restrictions for ACLs with fragment control

This topic outlines the configuration guidelines to follow when you use the fragments keyword in ACL entries to control the handling of initial and non-initial IP packet fragments.

These guidelines apply for ACLs with fragment control:

- Do not add the **fragments** keyword to every access-list entry, because the first fragment of the IP packet is considered a non-fragment and is treated independently of the subsequent fragments.

- Because an initial fragment does not match an access-list permit or deny entry that contains the **fragments** keyword, the packet is compared to the next access-list entry until it is either permitted or denied by an access-list entry that does not contain the **fragments** keyword.
- You may need two access-list entries for every deny entry. The first deny entry of the pair does not include the **fragments** keyword and applies to the initial fragment. The second deny entry of the pair includes the **fragments** keyword and applies to the subsequent fragments.
- If there are multiple **deny** access-list entries for the same host but with different Layer 4 ports, a single deny access-list entry with the **fragments** keyword for that host is all that has to be added. Thus, all the fragments of a packet are handled in the same manner by the access list.
- Packet fragments of IP datagrams are considered individual packets, and each fragment counts individually as a packet in access-list accounting and access-list violation counts.
- The **fragments** keyword cannot be configured for an access-list entry that contains any Layer 4 information.
- Within the scope of ACL processing, Layer 3 information refers to fields located within the IPv4 header, for example, source, destination, and protocol. Layer 4 information refers to other data contained beyond the IPv4 header, for example, source and destination ports for TCP or UDP, flags for TCP, and type and code for ICMP.

Configure an IPv4 ACL to match on fragment type

This topic describes how to configure an IPv4 access control list (ACL) to match on the fragment type and perform an appropriate action to protect against fragmentation-based denial-of-service attacks.

Most denial-of-service (DoS) attacks work by flooding the network with fragmented packets. By filtering the incoming fragments of the packet in a network, you add an extra layer of protection against such attacks.

> **Note**
>
> IPv6 Extended Access Lists do not support the configuration of fragment types.

You can configure an IPv4 ACL to match on the fragment type and perform an appropriate action. Use the following configuration examples with the different fragment options.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create an IPv4 access list to match on the various fragment types by using the **ipv4 access-list** *acl-name* command.

   Example

   ```
   Router(config)# ipv4 access-list TEST
   ```

3. Add access control entries (ACEs) that match on the different fragment-type flags and forward traffic to the specified next hops.

- ACE 20 matches the **dont-fragment** flag (non-fragmented packet) and forwards the packet to the default (pre-configured) next hop.
- ACE 30 matches the **is-fragment** flag (fragmented packet) and forwards the packet to next hop 10.10.10.1.
- ACE 40 matches the **first-fragment** flag (first fragment of a fragmented packet) and forwards the packet to next hop 20.20.20.1.
- ACE 50 matches the **last-fragment** flag (last fragment of a fragmented packet) and forwards the packet to next hop 30.30.30.1.

Example

```
Router(config-ipv4-acl)# 10 permit tcp any any
Router(config-ipv4-acl)# 20 permit tcp any any fragment-type dont-fragment
default
Router(config-ipv4-acl)# 30 permit udp any any fragment-type is-fragment
nexthop1 ipv4 10.10.10.1
Router(config-ipv4-acl)# 40 permit ospf any any fragment-type first-fragment
 nexthop1 ipv4 20.20.20.1
Router(config-ipv4-acl)# 50 permit icmp any any fragment-type last-fragment
nexthop1 ipv4 30.30.30.1
```

4. Commit the configuration.

   Example

   ```
   Router(config-ipv4-acl)# commit
   Router(config-ipv4-acl)# exit
   ```

5. Create another IPv4 access list to forward the first fragment of a packet and drop the last fragment.

   The ACL checks the fragment offset value ('0' for the first fragment). If the fragment is the first fragment of the packet, the packet is forwarded. If the fragment is the last fragment of the packet, it is dropped at the interface.

   Example

   ```
   Router(config)# ipv4 access-list ACLFIRSTFRAG
   ```

6. Add an ACE to match the first fragment and forward it to next hop 192.168.1.2, and add an ACE to match the last fragment and drop it at the interface.

   Example

   ```
   Router(config-ipv4-acl)# 10 permit tcp any any fragment-type first-fragment
   nexthop1 ipv4 192.168.1.2
   Router(config-ipv4-acl)# 20 deny tcp any any fragment-type last-fragment
   ```

7. Commit the configuration.

   Example

   ```
   Router(config-ipv4-acl)# commit
   ```

8. Verify the configuration by using the **show access-lists** command.

Example

```
Router(config-ipv4-acl)# do show access-lists
ipv4 access-list ACLFIRSTFRAG
 10 permit tcp any any fragment-type first-fragment nexthop1 ipv4 192.168.1.20

 20 deny tcp any any fragment-type last-fragment
```

You have successfully configured an IPv4 ACL to match on the fragment type.

ACL matching by fragment offset

This topic describes the fragment-offset match feature in access control lists (ACLs) on the Cisco 8000 Series Router.

You can configure an access control list (ACL) rule to filter packets by the fragment-offset value. Depending on whether a packet matches the criteria in a permit or deny statement, the packet is either processed or dropped at the interface. Fragment-offset filtering is supported only in the ingress direction with compression mode of an ACL.

Configuration guidelines and restrictions for ACL matching by fragment offset

This topic outlines the configuration guidelines and restrictions to consider before you configure ACL matching by fragment offset.

These restrictions apply when you configure ACL matching by fragment offset:

- Fragment-offset filtering is supported for IPv4 packets in the default TCAM key for NC57-18DD-SE line cards, in traditional ACL mode, and not in compressed ACL mode.
- IPv6 Extended Access Lists do not support the configuration of ACL matching by fragment-offset values.

Configure ACL matching by fragment offset

This topic describes how to configure fragment-offset match in an access control list (ACL) on the Cisco 8000 Series Router.

To configure fragment-offset match in an ACL, use the **fragment-offset** option in the **permit** or **deny** command in IPv4 or IPv6 access-list configuration mode.

> **Note**
>
> For fragment-offset filtering, you must attach the specific ACL to an interface with compression level 2. Otherwise, the configuration is rejected.

Procedure

1. Configure an ACL rule to permit packets whose fragment-offset in the IPv4 header is within the range 300-400. The value *300-400* is based on the 8-byte unit, which is the same as a fragment-offset of *2400-3200* bytes.

Example

```
ipv4 access-list fragment-offset-acl
 10 permit ipv4 any any fragment-offset range 300 400
!
```

2. Verify the fragment-offset match in the ACL.

Example

```
Router# show access-lists ipv4 fragment-offset-acl usage pfilter loc 0/4/CPU0
Wed Apr 12 19:49:54.457 UTC
Interface : Bundle-Ether70
    Input  ACL : Common-ACL : N/A  ACL : fragment-offset-acl  (comp-lvl 3)
    Output ACL : N/A
```

```
Router# show access-lists ipv4 fragment-offset-acl hardware ing int
Bundle-Ether70 loc 0/4/CPU0
Wed Apr 12 19:51:07.837 UTC
ipv4 access-list fragment-offset-acl
 10 permit ipv4 any any fragment-offset range 300 400
```

ACL filtering by IP packet length

This topic describes the packet-length filtering feature in access control lists (ACLs) on the Cisco 8000 Series Router.

You can configure an access control list to filter packets by the packet length at an ingress interface. Depending on whether a packet matches the packet-length condition in a permit or deny statement, the packet is either processed or dropped at the interface.

To configure packet-length filtering in an ACL, use the **packet-length** option in the **permit** or **deny** command in IPv4 or IPv6 access-list configuration mode.

Restrictions for ACL filtering by packet length

This topic outlines the restrictions to consider before you configure ACL filtering by packet length.

These restrictions apply to packet-length filtering feature in ACLs:

- Packet-length filtering is supported only in the ingress direction, for both traditional (non-compression) and hybrid (compression) ACLs.
- IPv6 packet-length filtering is supported only for hybrid ACLs; not for traditional ACLs.
- Only quantized (value divisible by 16) packet-length filtering is supported for traditional ACLs on IPv4.
- For ACLs with compression level 2, by using the **hw-module profile tcam format** command, you can add packet length as a User Defined Field (UDF) along with **src-object-group** and **dst-object-group** as other UDFs.

Configure scaled IPv4 ACLs to filter by packet length

This topic describes the procedure to configure a scaled access control list (ACL) for IPv4 networks on the Cisco 8000 Series Router to filter packets by packet length.

To configure a scaled ACL to filter by packet length in IPv4 networks, use the following steps.

Procedure

1. Enable packet-length filtering in global configuration mode by using the **hw-module** command.

   Example

   ```
   Router# configure
   Router(config)# hw-module profile tcam format access-list ipv4 src-object-group
    dst-object-group dst-port proto packet-length frag-bit port-range
   ```

2. Enter global configuration mode and create an object group for configuring a scaled ACL.

   Example

   ```
   Router(config)# object-group network ipv4 netobject1
   Router(config-object-group-ipv4)# 50.0.0.0/24
   Router(config-object-group-ipv4)# commit
   ```

3. From global configuration mode, configure an IPv4 access list to filter packets by the packet-length value.

   In this example, you configure a statement to process only those packets that match the specified packet-length condition. All other packets are dropped when this ACL is applied to an ingress interface.

   Example

   ```
   Router# configure
   Router(config)# ipv4 access-list scaled_acl1
   Router(config-ipv4-acl)# 10 permit ipv4 net-group netobject1 any packet-length
    eq 1000
   ```

4. Commit the ACL and exit the IPv4 ACL configuration mode.

   Example

   ```
   Router(config-ipv4-acl)# commit
   Router(config-ipv4-acl)# end
   ```

5. Apply the ACL to the required HundredGigE interface.

   Example

   ```
   Router(config)# interface HundredGigE 0/5/0/3
   Router(config-if)# ipv4 access-group scaled_acl1 ingress compress level 2
   ```

6. Commit the configuration and exit the interface configuration mode.

   Example

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

7. Verify your configuration.

Example

```
Router# show access-lists scaled_acl1
ipv4 access-list scaled_acl1
10 permit ipv4 net-group netobject1 any packet-length eq 1000
```

8. Verify the ACL matches in hardware.

Example

```
Router# show access-lists scaled_acl1 hardware ingress location 0/5/CPU0
ipv4 access-list scaled_acl1
10 permit ipv4 net-group netobject1 any packet-length eq 1000 (1500 hw matches)
```

You have successfully configured a scaled IPv4 ACL to filter by packet length.

Configure scaled IPv6 ACLs to filter by packet length

This topic describes the procedure to configure a scaled access control list (ACL) for IPv6 networks on the Cisco 8000 Series Router to filter packets by packet length.

To configure a scaled ACL to filter by packet length in IPv6 networks, use the following steps.

Procedure

1. Enable packet-length filtering in global configuration mode by using the **hw-module** command.

Example

```
Router# configure
Router(config)# hw-module profile tcam format access-list ipv4 src-object-group
 dst-object-group dst-port proto packet-length frag-bit port-range
```

2. Enter global configuration mode and create an object group for configuring a scaled ACL.

Example

```
Router(config)# object-group network ipv6 netobject2
Router(config-object-group-ipv6)# 2001::0/128
Router(config-object-group-ipv6)# commit
```

3. From global configuration mode, configure a scaled IPv6 access list to filter packets by the packet-length value.

In this example, you configure a statement to process only those packets that match the specified packet-length condition. All other packets are dropped when this ACL is applied to an ingress interface.

Example

```
Router(config)# ipv6 access-list scaled_acl2
Router(config-ipv6-acl)# 10 permit ipv6 net-group netobject2 any packet-length
 eq 1000
Router(config-ipv6-acl)# commit
```

4. Commit the ACL and exit the IPv6 ACL configuration mode.

Example

```
Router(config-ipv6-acl)# commit
Router(config-ipv6-acl)# end
```

5. Apply the ACL to the required HundredGigE interface.

   Example

   ```
   Router# configure
   Router(config)# interface HundredGigE 0/5/0/3
   Router(config-if)# ipv6 access-group scaled_acl2 ingress compress level 3
   ```

6. Commit the configuration and exit the interface configuration mode.

   Example

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

7. Verify your configuration.

   Example

   ```
   Router# show access-lists ipv6 scaled_acl2
   ipv6 access-list scaled_acl2
   10 permit ipv6 net-group netobject2 any packet-length eq 1000
   ```

8. Verify the ACL matches in hardware.

   Example

   ```
   Router# show access-lists ipv6 scaled_acl2 hardware ingress location 0/5/CPU0
   ipv6 access-list scaled_acl2
   10 permit ipv6 net-group netobject2 any packet-length eq 1000 (2000 hw matches)
   ```

   You have successfully configured a scaled IPv6 ACL to filter by packet length.

TTL matching

This topic describes how IPv4 access control lists (ACLs) match on the Time-to-Live (TTL) value in the IPv4 header.

You can configure ACLs to match on the TTL value specified in the IPv4 header. You can specify the TTL match condition to be based on a single value or on multiple values.

TTL matching is supported for both ingress and egress ACLs.

Table 53: Feature History Table

Feature Name	Release Information	Description
TTL Matching	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on: • 88-LC1-48Y8H-EM
TTL Matching	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D

Configure TTL matching

This topic describes how to configure IPv4 access control lists (ACLs) on the Cisco 8000 Series Router to match on the Time-to-Live (TTL) value in the IPv4 header.

Use this procedure to configure an IPv4 ACL with TTL match conditions, attach the ACL to an interface, and verify the configuration.

Procedure

1. Configure an IPv4 ACL with the TTL parameters.

   Example

```
Router(config)# ipv4 access-list acl-v4
Router(config-ipv4-acl)# 10 deny tcp any any ttl eq 100
Router(config-ipv4-acl)# 20 permit tcp any any ttl range 1 50
Router(config-ipv4-acl)# 30 permit tcp any any ttl neq 100
Router(config-ipv4-acl)# commit
```

2. Attach the IPv4 ACL to the HundredGigE interface.

   Example

```
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv4 address 15.1.1.1 255.255.255.0
Router(config-if)# ipv4 access-group acl-v4 ingress
Router(config-if)# commit
```

3. Verify your configuration by using the **show run** command.

Example

```
Router(config)# show run
Building configuration...
!! IOS XR Configuration 0.0.0
!
ipv4 access-list acl-v4
 10 deny tcp any any ttl eq 100
 20 permit tcp any any ttl range 1 50
 30 permit tcp any any ttl neq 100
!
interface HundredGigE 0/0/0/0
 ipv4 address 15.1.1.1 255.255.255.0
 ipv4 access-group acl-v4 ingress
!
```

You have successfully configured TTL matching for IPv4 ACLs.

IP access list logging messages

This topic describes the logging messages that Cisco IOS XR Software generates when packets are permitted or denied by a standard IP access list on the Cisco 8000 Series Router.

Cisco IOS XR Software can provide logging messages about packets that are permitted or denied by a standard IP access list. Any packet that matches the access list causes an informational logging message about the packet to be sent to the console. The level of messages logged to the console is controlled by the **logging console** command in global configuration mode.

The first packet that triggers the access list causes an immediate logging message, and subsequent packets are collected over 5-minute intervals before they are displayed or logged. The logging message includes the access-list number, whether the packet was permitted or denied, the source IP address of the packet, and the number of packets from that source permitted or denied in the prior 5-minute interval.

You can use the **{ipv4 | ipv6} access-list log-update threshold** command to set the number of packets that, when they match an access list (and are permitted or denied), cause the system to generate a log message. Use this command to receive log messages more frequently than at 5-minute intervals.

> ⚠ **Caution**
>
> If you set the update-number argument to 1, a log message is sent right away, rather than cached; every packet that matches an access list causes a log message. A setting of 1 is not recommended because the volume of log messages could overwhelm the system.

Even if you use the **{ipv4 | ipv6} access-list log-update threshold** command, the 5-minute timer remains in effect, so each cache is emptied at the end of 5 minutes, regardless of the number of messages in each cache. Regardless of when the log message is sent, the cache is flushed and the count reset to 0 for that message the same way it is when a threshold is not specified.

> **Note**
>
> The logging facility might drop some logging message packets if there are too many to be handled or if more than one logging message is handled in 1 second. This behavior prevents the router from using excessive CPU cycles because of too many logging packets. Therefore, do not use the logging facility as a billing tool or as an accurate source of the number of matches to an access list.

Interface logging on ACLs

This topic describes how interface logging on IPv4 and IPv6 access control lists (ACLs) generates log messages that identify the interface through which traffic enters or exits the router.

You can enable interface logging on IPv4 and IPv6 access control lists (ACLs) so that log messages identify the interface through which traffic enters or exits the router. The router supports two types of interface logging on ACLs:

- Ingress interface logging: uses the **log-input** keyword on ACEs to include, in the log message, the ingress interface on which the router receives the packet. The router supports this feature for both IPv4 and IPv6 ingress ACLs on main interfaces, sub-interfaces, and bridge-group virtual interfaces (BVI).
- Egress interface logging: uses the **log** option on ACEs to identify the packet counts matching the ACEs. With this log option, for egress traffic, you can fetch information such as access list number, packets permitted or denied, and source or destination addresses of the packets.

Table 54: Feature History Table

Feature Name	Release Information	Feature Description
ACL Log Message Collection for Egress Traffic	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on: • 88-LC1-48Y8H-EM
ACL Log Message Collection for Egress 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
ACL Log Message Collection for Egress 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: • 8011-4G24Y4H-I • 8712-MOD-M

Feature Name	Release Information	Feature Description
ACL Log Message Collection for Egress 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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
ACL Log Message Collection for Egress Traffic	Release 7.10.1	We have made it easier for you to monitor ACL egress traffic, assess traffic load on an ACL, and troubleshoot issues. This is made possible by adding a **log** option to the ACEs that are associated with an interface and identify the packet counts matching the ACEs. With this log option, for an egress traffic, you can fetch information, such as access list number, packets permitted or denied, and source or destination addresses of the packets.
Enable Ingress Interface Logging on IPv4 and IPv6 ACLs	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
Enable Ingress Interface Logging on IPv4 and IPv6 ACLs	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
Enable Ingress Interface Logging on IPv4 and IPv6 ACLs	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Enable Ingress Interface Logging on IPv4 and IPv6 ACLs	Release 7.8.1	Using the **log-input** keyword, you can now enable Access Control Lists (ACLs) to generate log messages that help you identify the interface through which a particular traffic stream ingresses the routers. This information aids in optimizing traffic flow across the network. There was no option to enable logging of ingress interfaces with an ACL in earlier releases. This feature introduces an optional keyword **log-input** for the following commands: • deny (IPv4) • deny (IPv6) • permit (IPv4) • permit (IPv6)

Configuration guidelines and limitations for interface logging on ACLs

This topic outlines the configuration guidelines and limitations to consider before you enable interface logging on IPv4 and IPv6 ACLs.

These guidelines apply for interface logging on ACLs:

- The router supports this feature for both IPv4 and IPv6 egress ACLs on main interfaces, sub-interfaces, and bridge-group virtual interfaces (BVI).
- This feature is supported only on the Cisco 8000 Series Routers that use Q100 and Q200-based line cards.

Enable ingress interface logging

This topic describes how to enable ingress interface logging on IPv4 and IPv6 ACLs by using the log-input keyword so that log messages include the ingress interface on which the router receives the packet.

The **log-input** option provides the same functionality as the **log** keyword, except that the log message also includes the ingress interface on which the router receives the packet. The router supports this feature for both IPv4 and IPv6 ingress ACLs on main interfaces, sub-interfaces, and bridge-group virtual interfaces (BVI).

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create an IPv4 access list and add an access control entry (ACE) with the **log-input** keyword.

   Example

   ```
   Router(config)# ipv4 access-list test
   Router(config-ipv4-acl)# 10 deny udp 10.1.1.0 255.255.255.0 log-input
   Router(config-ipv4-acl)# exit
   ```

3. Attach the ACL to an ingress interface by using the **ipv4 access-group** *acl-name* **ingress** command and commit the configuration.

   Example

   ```
   Router(config)# interface FortyGigE0/0/0/22
   Router(config-if)# ipv4 access-group test ingress
   Router(config-if)# commit
   ```

4. Verify ingress interface logging by checking the generated log messages.

   The following snippet shows a sample log message when you enable this option on an ACE:

   Example

   ```
   Router: ipv4_acl_mgr[132]: %ACL-IPV4_ACL-6-IPACCESSLOGP : access-list test
   (10) deny udp
   10.1.1.2(0) FortyGigE0/0/0/22-> 10.2.2.2(0), 63782 packets
   ```

   For more information on how to configure IPv4 and IPv6 ACLs, see IPv4 ACLs on page 156 and IPv6 ACLs on page 160.

   You have successfully enabled ingress interface logging on the ACL.

Enable egress interface logging

This topic describes how to enable egress interface logging on IPv4 and IPv6 ACLs by using the log keyword on ACEs so that log messages provide information about egress traffic on an interface.

Use this procedure to configure an IPv4 ACL with the **log** keyword on ACEs and attach it to an egress interface. The log message provides the following information:

- access list number
- packet permitted or denied

- protocol used, such as TCP, UDP, or ICMP
- source and destination addresses
- source and destination port numbers

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create an IPv4 access list and add access control entries (ACEs) with the **log** keyword on the entries that you want to log.

   Example

   ```
   Router(config)# ipv4 access-list test-egress-acl
   Router(config-ipv4-acl)# 10 deny tcp host 30.0.10.1 host 30.0.20.1 precedence
    priority
   Router(config-ipv4-acl)# 11 permit tcp 30.0.10.1 0.0.0.255 30.0.20.1 0.0.0.255
    dscp af22 log
   Router(config-ipv4-acl)# 17 deny udp host 30.0.10.1 host 30.0.20.1 precedence
    priority log
   Router(config-ipv4-acl)# exit
   ```

3. Attach the ACL to an egress interface by using the **ipv4 access-group** *acl-name* **egress** command and commit the configuration.

   Example

   ```
   Router(config)# interface FourHundredGigE0/1/0/0
   Router(config-if)# ipv4 access-group test-egress-acl egress
   Router(config-if)# commit
   ```

4. Verify the running configuration of the IPv4 egress ACL and the interface by using the **show running-config** command.

   Example

   ```
   Router# show running-config
   Wed May  3 16:09:25.917 UTC
   ipv4 access-list test-egress-acl
    10 deny tcp host 30.0.10.1 host 30.0.20.1 precedence priority
    11 permit tcp 30.0.10.1 0.0.0.255 30.0.20.1 0.0.0.255 dscp af22 log
   17 deny udp host 30.0.10.1 host 30.0.20.1 precedence priority log
   !
   interface FourHundredGigE0/1/0/0
    mtu 1530
    ipv4 address 30.0.10.2 255.255.255.0
    ipv6 address 130:1::2/96
    ipv4 access-group test-egress-acl egress
   !
   ```

5. Verify the packet counts for the filtered egress packets on the interface by using the **show access-lists ipv4** *acl-name* **hardware egress location** *node-id* command.

   Example

```
Router# show access-lists ipv4 test-egress-acl hardware egress location
0/1/CPU0
Wed May  3 16:09:25.917 UTC
ipv4 access-list test-egress-acl
 10 deny tcp host 30.0.10.1 host 30.0.20.1 precedence priority
 11 permit tcp 30.0.10.0 0.0.0.255 30.0.20.0 0.0.0.255 dscp af22 log (1440
matches)
 17 deny udp host 30.0.10.1 host 30.0.20.1 precedence priority log
```

6. Verify the ACL logging messages for the egress traffic by using the **show logging** command.

   Example

```
Router# show logging | i permit
Wed May  3 16:12:30.162 UTC
Router:May 3 16:09:08.251 UTC: ipv4_acl_mgr[431]: %ACL-IPV4_ACL-6-IPACCESSLOGP
 : access-list test-egress-acl (11) permit tcp 30.0.10.1(1024) ->
30.0.20.1(1024), 1 packet
Router:May 3 16:10:08.396 UTC: ipv4_acl_mgr[431]: %ACL-IPV4_ACL-6-IPACCESSLOGP
 : access-list test-egress-acl (11) permit tcp 30.0.10.1(1024) ->
30.0.20.1(1024), 1000 packet
```

   You have successfully enabled egress interface logging on the ACL and verified the packet counts and log messages for the egress traffic.

Per-interface statistics

This topic describes per-interface ACL entry (ACE) drop counters, which you enable by using the interface-statistics keyword when you bind an ACL to an interface.

When you bind an ACL to interfaces, you can configure ACE drop counts by using the **interface-statistics** keyword in per-interface mode.

You can allocate up to 8 stats counters per NPU. This also limits the number of interfaces that can support the same ACL in per-interface-stats mode to 8. Additional binding of the same ACL in per-interface-stats mode is rejected.

Note

For a specific direction, all interfaces on a line card using the same ACL must be configured in the same stats mode; otherwise, the subsequent binding is rejected.

This behavior applies to bundle interfaces too. If you add any member to the existing bundle interface with a different stats mode, the binding is rejected.

Table 55: Feature History Table

Feature Name	Release Information	Feature Description
Per Interface Statistics	Release 26.2.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]):* *This feature is supported on: • 88-LC1-48Y8H-EM
Per Interface Statistics	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D

Configure per-interface statistics

This topic describes how to enable per-interface ACL entry (ACE) drop counters by using the interface-statistics keyword when you bind an ACL to an interface, and how to verify the deny statistics on the interface.

Use this procedure to bind an ACL to an interface with the **interface-statistics** keyword and verify the per-interface deny statistics.

Procedure

1. Bind the ACL to the interface with the **interface-statistics** keyword.

   Example

   ```
   Router(config)# interface HundredGigE 0/0/0/0
   Router(config-if)# ipv4 address 1.1.1.1 255.255.0.0
   Router(config-if)# ipv6 address 2001::1/64
   Router(config-if)# ipv4 access-group test ingress interface-statistics
   Router(config-if)# commit
   ```

2. Verify the per-interface deny statistics by using the **show access-lists ipv4** *acl-name* **hardware ingress interface** *interface* **location** *node-id* command.

Example

```
Router# show access-lists ipv4 test hardware ingress interface FourHundredGigE
 0/1/0/0 location 0/1/CPU0
ipv6 access-list test
10 permit ipv6 any 200:23::/64
20 deny udp any any (2356 matches)
```

You have successfully enabled per-interface ACE drop counters and verified the deny statistics on the interface.

11 Cisco Express Forwarding

Topics:

Describes CEF operation, route programming, forwarding scale, load distribution, and diagnostics. Provides requirements and tasks for configuring and verifying supported forwarding features.

Cisco Express Forwarding (CEF) is a Layer 3 forwarding architecture that uses forwarding and adjacency information to switch IPv4 and IPv6 packets. It supports route programming, hardware scale profiles, and load distribution. CEF also provides operational commands for verification and forwarding diagnostics.

This chapter covers CEF fundamentals, static routes, proactive adjacency resolution, route-scale profiles, shortened IPv6 prefixes, per-NPU hash rotation, and FIB diagnostics.

Cisco Express Forwarding fundamentals

Explains how CEF uses forwarding and adjacency data to provide scalable Layer 3 packet forwarding and covers its operation, supported functions, benefits, and access requirement.

Cisco Express Forwarding (CEF) is an advanced Layer 3 IP switching technology that

- optimizes performance for large and dynamic traffic patterns
- maintains forwarding and adjacency information for software and hardware forwarding engines, and
- operates as an inherent feature without requiring an enablement configuration.

Cisco 8000 Series Routers support single-stage forwarding.

How Cisco Express Forwarding works

Describes how CEF processes routing updates, resolves routes, maintains forwarding and adjacency tables, and forwards IPv4 and IPv6 unicast packets.

CEF always operates in CEF mode in Cisco IOS XR Software.

Summary

CEF uses two key components:

- Forwarding Information Base (FIB): Maintains IPv4 and IPv6 unicast forwarding tables on the route processor and each line card.
- Adjacency Information Base (AIB): Maintains protocol-independent adjacency information.

Workflow

These stages describe CEF packet forwarding:

1. Each routing protocol selects suitable routes and installs them in the Routing Information Base (RIB).
2. The FIB process on each node receives RIB updates, resolves routes, and independently maintains its forwarding table.
3. CEF uses the FIB and AIB data to forward packets through software or hardware forwarding engines.

Result

The route processor and line cards maintain forwarding tables that can differ slightly because each node processes updates independently.

CEF features and benefits

Describes the forwarding capabilities, operational features, maintained IPv4 and IPv6 databases, and performance benefits that CEF provides on Cisco IOS XR Software.

Use this information to identify the functions and benefits of CEF.

CEF supports these functions and operational features:

- Software switching paths
- Forwarding and adjacency tables for software and hardware forwarding engines
- Bundle interfaces and multiple paths
- Route consistency

- Packaging, restartability, and out-of-resource handling
- OSPFv2 shortest path first prefix prioritization
- Border Gateway Protocol (BGP) attribute downloads

CEF provides these benefits:

- Performance: Uses less processor capacity than fast-switching route caching.
- Scalability: Provides full switching capacity at each line card.
- Resilience: Uses a complete FIB lookup table to avoid route-cache maintenance during routing changes.

Table 56: CEF forwarding databases

Database	Function
IPv4 CEF database	Stores IPv4 unicast routes for forwarding IPv4 unicast packets.
IPv6 CEF database	Stores IPv6 unicast routes for forwarding IPv6 unicast packets.

Verify CEF tables

Verifies IPv4 or IPv6 CEF table contents and resolution state by displaying summary information and detailed forwarding entries for specific addresses.

Verify the contents and resolution state of an IPv4 or IPv6 CEF table.

Before you begin

Confirm that your task group permits the required CEF show commands.

Procedure

1. Display a summary of an IPv4 or IPv6 CEF table.

Example

```
Router# show cef ipv4 summary
Router ID is 216.1.1.1

IP CEF with switching (Table Version 0) for node0_RP0_CPU0

  Load balancing: L4
  Tableid 0xe0000000 (0x8cf5b368), Vrfid 0x60000000, Vrid 0x20000000, Flags
0x1019
  Vrfname default, Refcount 4129
  56 routes, 0 protected, 0 reresolve, 0 unresolved (0 old, 0 new), 7616 bytes

  13 rib, 0 lsd, 0:27 aib, 1 internal, 10 interface, 4 special, 1 default
routes
  56 load sharing elements, 24304 bytes, 1 references
  1 shared load sharing elements, 432 bytes
  55 exclusive load sharing elements, 23872 bytes
  0 route delete cache elements
  13 local route bufs received, 1 remote route bufs received,  0 mix bufs
received
  13 local routes, 0 remote routes
```

```
  13 total local route updates processed
  0 total remote route updates processed
  0 pkts pre-routed to cust card
  0 pkts pre-routed to rp card
  0 pkts received from core card
  0 CEF route update drops, 0 revisions of existing leaves
  0 CEF route update drops due to version mis-match
  Resolution Timer: 15s
  0 prefixes modified in place
  0 deleted stale prefixes
  0 prefixes with label imposition, 0 prefixes with label information
  0 LISP EID prefixes, 0 merged, via 0 rlocs
 28 next hops
  1 incomplete next hop

 0 PD backwalks on LDIs with backup path
```

Use **show cef ipv6 summary** for the IPv6 table.

2. Display detailed CEF information for an address.

Example

```
Router# show cef 203.0.113.2 detail
203.0.1.2/32, version 102239, internal 0x1000001 0x0 (ptr 0xa932b408) [1],
0x0 (0xaf4a6ad8), 0xa20 (0xc22c6da8)
 Updated Jul  3 21:40:17.827
 local adjacency 203.1.104.2
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
  gateway array (0xb9061e70) reference count 1982, flags 0x8068, source lsd
(5), 1 backups
                [1983 type 4 flags 0x108401 (0x943df068) ext 0x0 (0x0)]
  LW-LDI[type=1, refc=1, ptr=0xaf4a6ad8, sh-ldi=0x943df068]
  gateway array update type-time 1 Jul  3 20:23:36.957
 LDI Update time Jul  3 20:23:36.964
 LW-LDI-TS Jul  3 21:40:17.834
   via 203.1.104.2/32, Bundle-Ether104, 11 dependencies, weight 0, class 0
[flags 0x0]
    path-idx 0 NHID 0x0 [0xa446b0a8 0x0]
    next hop 203.1.104.2/32
    local adjacency
   via 203.1.114.2/32, Bundle-Ether114, 9 dependencies, weight 0, class 0
[flags 0x0]
    path-idx 1 NHID 0x0 [0xa446ac18 0x0]
    next hop 203.1.114.2/32
    local adjacency

    Load distribution: 0 1 (refcount 1983)

    Hash  OK  Interface                 Address
    0     Y   Bundle-Ether104           203.1.104.2
    1     Y   Bundle-Ether114           203.1.114.2
```

Configuration status of CEF hardware modules

Explains how hardware-module status identifies configured CEF profiles, whether each profile is applied, and whether a reload or commit action remains pending.

CEF hardware-module status is operational information that

- identifies whether each hardware profile is configured
- reports whether the router applied each profile, and
- identifies any reload or commit action needed to apply a profile.

Table 57: Feature History Table

Feature name	Release Information	Feature Description
Configuration Status of Cisco Express Forwarding (CEF) Hardware Modules	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
Configuration Status of Cisco Express Forwarding (CEF) Hardware Modules	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
Configuration Status of Cisco Express Forwarding (CEF) Hardware Modules	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Configuration Status of Cisco Express Forwarding (CEF) Hardware Modules	Release 24.1.1	Introduced in this release on: Cisco 8011-4G24Y4H-I router.

Feature name	Release Information	Feature Description
Configuration Status of Cisco Express Forwarding (CEF) Hardware Modules	Release 7.3.1	This feature enables you to view pending actions, such as a reload or a commit action, which is applicable to CEF hardware-modules. The show hw-module profile cef command is introduced for this feature.

View CEF hardware-module status

Displays the configuration, application, and pending-action status of CEF hardware-module profiles in XR EXEC mode and explains the reported status columns.

Determine whether each CEF hardware-module profile is configured and applied.

Procedure

Display CEF hardware-module profile status.

Example

```
Router# show hw-module profile cef
-------------------------------------------------------------------------------
Knob                          Status      Applied    Action
-------------------------------------------------------------------------------
BGPLU                          Configured    No      Reload
Dark Bandwidth                 Unconfigured  Yes     None
MPLS Per Path Stats            Unconfigured  Yes     None
Tunnel TTL Decrement           Configured    Yes     None
High-Scale No-LDP-Over-TE      Unconfigured  Yes     None
```

The Status column reports configuration state, the Applied column reports application state, and the Action column reports any required action.

Static routes and Cisco Express Forwarding

Explains when static routes provide predictable paths for CEF forwarding and why dynamic routing is more suitable for large or frequently changing networks.

A static route is a manually configured route-table entry that

- defines an explicit path between routers
- uses less bandwidth than a dynamically calculated route, and
- requires manual reconfiguration when the network path changes.

Suitable uses

Use static routes in simple networks with predictable traffic or to define a gateway of last resort. Use dynamic routing for large or frequently changing networks.

Configure a static route

Configures an IPv4 unicast static route through a HundredGigE interface.

Create a static route to destination prefix 203.0.113.0/24 through next hop 192.0.2.1.

Procedure

1. Configure the destination prefix, outgoing interface, and next hop in IPv4 unicast static-router configuration mode.

   Example

```
Router# configure
Router(config)# router static address-family ipv4 unicast
Router(config-static-afi)# 203.0.113.0/24 HundredGigE0/0/0/0 192.0.2.1
Router(config-static-afi)# commit
```

2. Display the static IPv4 unicast configuration.

   Example

```
Router# show running-config router static address-family ipv4 unicast
router static
 address-family ipv4 unicast
  203.0.113.0/24 HundredGigE0/0/0/0 192.0.2.1
 !
!
```

BGP attribute downloads

Explains how downloaded Border Gateway Protocol (BGP) attributes become visible in CEF and how global or local attribute identifiers select a specific installed attribute for inspection.

Border Gateway Protocol (BGP) attribute download is a CEF capability that

- makes installed BGP attributes visible through CEF commands
- identifies attributes by a BGP attribute ID, and
- supports lookup by a local attribute ID.

View downloaded BGP attributes

Displays all BGP attributes installed in CEF or selects one installed attribute by its global attribute identifier or local attribute identifier.

Inspect the BGP attributes that CEF installed.

Procedure

1. Display all installed BGP attributes.

Example

```
Router# show cef bgp-attribute
VRF: default

_____________
Table ID: 0xe0000000. Total number of entries: 1
OOR state: GREEN. Number of OOR attributes: 0

BGP Attribute ID: 0x6, Local Attribute ID: 0x1
    Aspath      :    2
    Community   :
    Origin AS   :    2
    Next Hop AS :    2
```

2. To inspect one attribute, select it by its BGP attribute ID or local attribute ID.

   Use **show cef bgp-attribute attribute-id** or **show cef bgp-attribute local-attribute-id** command.

The output identifies installed attributes and their resource state.

Proactive Address Resolution Protocol and Neighbor Discovery

Explains how proactive ARP and ND preemptively resolve incomplete Layer 2 next-hop information for CEF routes, reducing first-packet software forwarding and helping prevent resolution failures and traffic drops.

Proactive Address Resolution Protocol (ARP) and Neighbor Discovery (ND) are CEF adjacency-resolution capabilities that

- detect routes with incomplete Layer 2 next-hop information
- start ARP or ND resolution before the first packet requires the adjacency, and
- retry resolution every 15 seconds until the next-hop information is complete.

Incomplete next-hop information

When Cisco Express Forwarding (CEF) installs a route without Layer 2 adjacency information, it creates an incomplete Layer 3 next hop and programs the entry in hardware.

Without proactive resolution, the first packet uses the software forwarding path. Feature-specific information in the Layer 2 header can prevent the software path from removing the header completely, which can cause ARP or ND resolution to fail and traffic to be dropped.

How proactive ARP and ND work

Explains how proactive address resolution works by having CEF detect incomplete next hops, start ARP or ND resolution before traffic arrives, and retry until Layer 2 adjacency information is complete.

Without proactive resolution, a packet with feature-specific Layer 2 information can reach the software forwarding path before CEF has complete adjacency information.

Summary

The process involves these components:

- CEF: Detects the incomplete next hop and programs it in hardware.
- Address Resolution Protocol (ARP) or Neighbor Discovery (ND): Resolves the missing Layer 2 adjacency.

Workflow

These stages describe proactive address resolution:

1. CEF installs a route without complete Layer 2 adjacency information.
2. CEF proactively starts ARP or ND resolution instead of waiting for the first packet.
3. CEF retries resolution every 15 seconds until it resolves the next hop.

Result

CEF obtains the adjacency information before traffic depends on the incomplete next hop.

Enable proactive ARP and ND

Enables proactive ARP and ND so that CEF resolves incomplete Layer 2 next-hop information before packet forwarding depends on the adjacency.

Enable CEF to start ARP or ND resolution when a static route has incomplete next-hop information.

Procedure

1. Enable proactive ARP and ND resolution in global configuration mode.

   Example

   ```
   Router# configure
   Router(config)# cef proactive-arp-nd enable
   Router(config)# commit
   ```

2. Display the running configuration.

   Example

   ```
   Router# show running-config
   cef proactive-arp-nd enable
   end
   ```

Route-scale profiles

Explains how a route-scale profile reallocates hardware resources to increase IPv4 and IPv6 FIB capacity. Covers capacity changes, classification tradeoffs, configuration, and restoration.

A route-scale profile is a hardware resource allocation that

- increases the number of IPv4 and IPv6 routes that the FIB can store
- reassigns existing router resources to the forwarding database, and
- reduces the resources available to some packet-classification features.

Table 58: Feature History Table

Feature Name	Release Information	Description
Route Scale Improvements	Release 7.9.1	This feature enables you to increase the number of Forwarding Information Base (FIB) entries supported for IPv4 traffic from 2 million to 3 million and IPv6 traffic from 0.5 million to 1 million. The increased FIB entries allow the router to route more traffic streams. It also helps the router to achieve a faster switch or process-switch forwarding scenario by eliminating the frequent need for route cache maintenance due to fewer route entries in the FIB database. This feature introduces the **hw-module profile route scale** command.

Route-scale capacity and resource allocation

Describes the IPv4 and IPv6 FIB capacity increases and the corresponding reduction in ternary content-addressable memory resources available for packet classification features.

Use the capacity values to evaluate the route-scale profile before you enable it.

Table 59: Route-scale resource changes

Traffic type	Default FIB capacity	Increased FIB capacity	Classification resource reduction
IPv4	2 million entries	3 million entries	512 ternary content-addressable memory entries
IPv6	0.5 million entries	1 million entries	256 ternary content-addressable memory entries

Restrictions of route-scale profiles

Provides the resource limitation that affects access control lists, BGP FlowSpec, quality of service, and local packet transport services after route-scale expansion.

Account for reduced packet-classification resources before you enable route-scale improvements.

The restriction affects security access control lists, quality-of-service access control lists, BGP FlowSpec, and local packet transport services.

The profile reallocates hardware resources from packet classification to the FIB.

IPv4 expansion reduces available classification space by 512 entries. IPv6 expansion reduces it by 256 entries.

The Cisco 8010 Series Routers do not support this feature.

Configure the route-scale profile

Configures the route-scale profile to increase IPv4 and IPv6 FIB capacity, reloads all locations, and identifies how to restore the default capacity.

Increase IPv4 capacity to 3 million entries and IPv6 capacity to 1 million entries.

Before you begin

Confirm that the router supports route-scale improvements and that reduced packet-classification resources are acceptable.

Procedure

1. Configure the route-scale profile in global configuration mode.

   Example

   ```
   Router# configure
   Router(config)# hw-module profile route scale lpm tcam-banks
   Router(config)# commit
   ```

2. Reload all locations.

   Example

   ```
   Router# reload location all
   ```

3. Display the running configuration.

   Example

   ```
   Router# show running-config
   hw-module profile route scale lpm tcam-banks
   ```

What to do next

To restore default capacity, remove the profile with the **no** form of the command and reload all locations.

Full-scale longest-prefix-match mode

Explains how full-scale longest-prefix-match (LPM) mode reallocates memory and ternary content-addressable memory to maximize IPv4 and IPv6 route capacity.

Full-scale longest-prefix-match (LPM) mode is a hardware routing profile that

- maximizes the supported IPv4 and IPv6 route scale
- reallocates memory and ternary content-addressable memory resources, and
- requires a manual chassis or line-card reload after a configuration change.

Table 60: Feature History Table

Feature Name	Release Information	Description
Full-scale route LPM mode	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8711-48Z-M routers.
Full-scale route LPM mode	Release 25.2.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100]) This feature enables you to increase the number of route entries supported for IPv4 and IPv6 traffic with the full-scale capability. You can now maximize the route scale for IPv4 and IPv6 traffic and this helps to reallocate hardware resources to support a higher number of routing entries. CLI: The full-scale keyword is introduced in the hw-module profile route scale lpm command. This feature is supported on Cisco 8712-MOD-M routers.

Benefits of full-scale longest-prefix-match mode

Describes how full-scale LPM mode supports extensive IPv4 and IPv6 routing tables by reallocating hardware resources to the maximum available LPM route capacity.

Use full-scale LPM mode in environments that require extensive IPv4 and IPv6 routing tables.

Full-scale LPM mode provides these benefits:

- Allocates more memory and ternary content-addressable memory to IP routes.
- Supports larger service-provider or enterprise routing tables.
- Uses the maximum LPM route capacity of the supported hardware.

Configure full-scale LPM mode

Configures full-scale LPM mode, commits the hardware-profile change, reloads the affected chassis or line cards, and verifies that the profile is applied.

Maximize the IPv4 and IPv6 LPM route scale on supported hardware.

The default LPM profile operates in half-scale mode.

Procedure

1. Configure full-scale LPM mode.

Example

```
Router# config
Router(config)# hw-module profile route scale lpm full-scale
Router(config)# commit
```

2. Reload the chassis or all affected line cards.

Example

```
Router# reload location all
```

3. Verify the route-scale profile.

Example

```
Router# show hw-module profile route-scale
-------------------------------------------------------------
Knob                      Status           Applied     Action
LPM-Full-Scale            Configured       Yes          None
```

The LPM full-scale profile reports a configured and applied state.

What to do next

To restore half-scale mode, remove the profile with the **no** form of the command and reload all locations.

Shortened IPv6 routing prefixes

Explains how shortening wide IPv6 prefixes improves LPM memory use and mitigates out-of-resource conditions. Covers operation, platform restrictions, configuration, reload, and verification.

Shortened IPv6 routing prefixes are wide prefixes that the router stores in a shorter LPM representation to

- reduce the number of LPM entries used by prefixes longer than 64 bits
- accommodate more wide IPv6 routing prefixes, and
- mitigate out-of-resource conditions on supported hardware.

Table 61: Feature History Table

Feature Name	Release Information	Description
Shortened IPv6 Routing Prefixes	Release 25.4.1	Introduced in this release on: Fixed Systems (010 [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 Information	Description
Shortened IPv6 Routing Prefixes	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
Shortened IPv6 Routing Prefixes	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Shortened IPv6 Routing Prefixes	Release 24.1.1	We've improved the memory utilization of the Longest Prefix Match (LPM) table, allowing you to accommodate more IPv6 routing prefixes with wider IPv6 prefix lengths by storing them as shorter routing prefixes. This approach conserves router resources by reducing the number of entries in the routing table and is particularly helpful in mitigating out-of-resource (OOR) situations. This feature is supported only on Cisco Silicon One Q200-based routers and line cards. This feature modifies the **hw-module profile route scale** command with the new **lpm wide-entries shortened** keyword.

How shortened IPv6 routing prefixes work

Describes how the router transforms IPv6 routing prefixes longer than 64 bits into shorter LPM entries to conserve forwarding resources.

An IPv6 address is 128 bits long and contains a routing prefix, subnet ID, and interface ID. The LPM table stores routing-prefix bits for forwarding decisions.

Summary

The router uses the LPM entry to determine the egress interface and next destination for a packet.

Workflow

These stages describe shortened-prefix storage:

1. The router receives an IPv6 route with a prefix longer than 64 bits.
2. The enabled hardware profile converts the wide prefix into a shorter routing-prefix representation.
3. The router stores the shortened representation in the LPM table and uses it for forwarding lookup.

Result

The LPM table accommodates more wide IPv6 routes with the available resources.

Guidelines and restrictions of shortened IPv6 routing prefixes

Provides the reload requirement, supported-hardware restriction, and feature-compatibility restrictions that apply when you shorten wide IPv6 routing prefixes in the LPM table.

- Reload the router after you enable shortened IPv6 routing prefixes.
- Use the feature only on supported Cisco Silicon One Q200-based routers and line cards.
- Do not enable IPv6 prefix scale expansion with shortened IPv6 routing prefixes.
- Do not enable unicast Reverse Packet Forwarding (uRPF) with shortened IPv6 routing prefixes.

Configure shortened IPv6 routing prefixes

Enables shortened storage for wide IPv6 prefixes, reloads all locations, and verifies that the Wide-Entries-Shortened profile is configured and applied.

Reduce the LPM resources used by IPv6 prefixes longer than 64 bits.

Before you begin

Confirm that the router or line card uses supported Cisco Silicon One Q200 hardware.

Procedure

1. Enable shortened wide entries in global configuration mode.

   Example

   ```
   Router# configure
   Router(config)# hw-module profile route scale lpm wide-entries shortened
   Router(config)# commit
   ```

2. Reload all locations.

   Example

   ```
   Router# reload location all
   ```

3. Verify the route-scale profile.

Example

```
Router# show hw-module profile route-scale
-------------------------------------------------------------
Knob                          Status          Applied   Action
-------------------------------------------------------------
Wide-Entries-Shortened  Configured   Yes     None
```

The route-scale profile reports that shortened wide entries are configured and applied.

What to do next

To disable the feature, remove the profile with the **no** form of the command and reload all locations.

FIB object-ID diagnostics

Explains how FIB object-ID diagnostics expose allocated identifiers in the default and encapsulation-ID tables for targeted forwarding-object troubleshooting and inspection.

Forwarding Information Base (FIB) object-ID diagnostics are command outputs that

- list all allocated object IDs in the default table
- list all allocated object IDs in the encap-id table, and
- display a selected object ID from either table.

Table 62: Feature History Table

Feature name	Release Information	Feature Description
Enhance FIB debuggability	Release 25.3.1	You can now enhance the debuggability of FIB objects by requesting information like allocated object-ids within the required table. This feature introduces these changes: CLI: • **show cef global object-id summary** • **show cef global object-id table**

View FIB object IDs

Displays summary or table-level information about object IDs allocated to FIB tables so that you can isolate forwarding objects during debugging.

Inspect the object IDs allocated within the default or encap-id FIB table.

Procedure

Select the required diagnostic view.

- Use **show cef global object-id summary** to display an allocation summary.
- Use **show cef global object-id table** to display table-level object-ID information.

The command output displays object-ID allocation information for the selected FIB scope.

Hash rotation for individual NPUs

Explains how per-NPU hash rotation varies path-selection inputs to reduce traffic polarization. Covers the forwarding process, deployment restriction, configuration priorities, and hardware verification.

Per-network-processing-unit (NPU) hash rotation is a Silicon One load-balancing technique that

- changes the hashing function by applying a rotation value
- allows a different value for each NPU, and
- reduces repeated selection of the same path in highly symmetric networks.

You can configure a global hash rotation value, configure a value for an individual NPU, or let the router calculate values automatically.

Table 63: Feature History Table

Feature Name	Release Information	Feature Description
Per-NPU hash rotation	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) (select variants only*) This feature is supported on Cisco 8712-MOD-M routers.
Per-NPU hash rotation	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]) You can now configure the hash rotation value for each Network Processing Unit (NPU) to improve traffic load balancing and minimize traffic polarization. Alternatively, the value can be automatically calculated, eliminating the need for manual configuration.

How hash rotation reduces traffic polarization

Describes how varied hash-rotation values change path selection across routers or NPUs to distribute traffic more evenly in a symmetric network.

A hash function maps packet-header information to an available path. Reusing the same function and inputs across symmetric devices can repeatedly select one path and leave other paths underused.

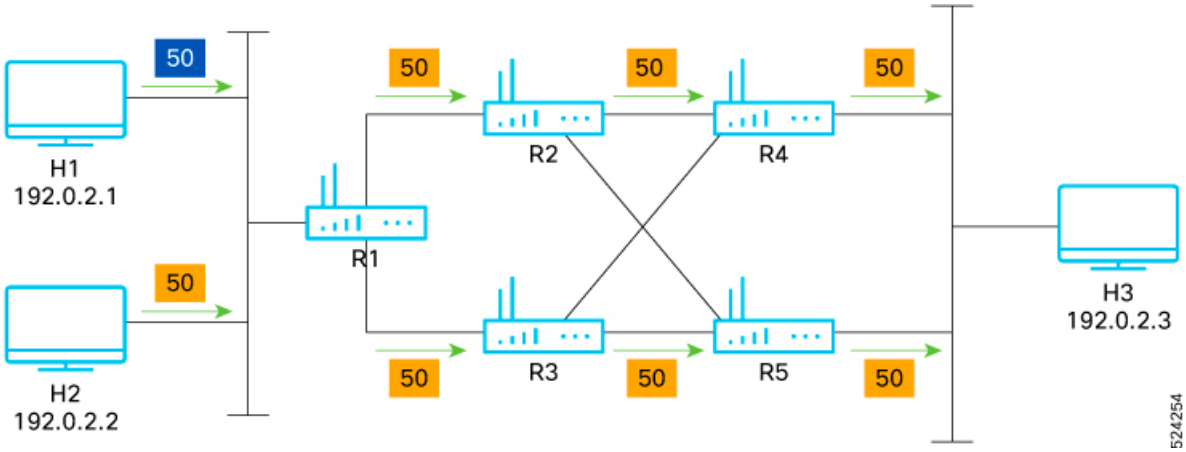
Summary

The process uses these elements:

- Hash function: Produces the value that selects a path or interface.
- Traffic polarization: Concentrates similar traffic on the same path.
- Hash rotation: Changes the hash parameters that influence path selection.

Workflow

Figure 14: Network topology for optimized hash rotation and load balancing



Routers R1, R2, and R3 forward traffic from hosts H1 and H2 toward host H3 through multiple paths. Identical hashing can leave the R2-to-R5 and R3-to-R4 links unused.

These stages describe load balancing with hash rotation:

1. An ingress router hashes packet-header fields and distributes packets across available paths.
2. Downstream routers that reuse the same hash inputs can map each flow to the same outgoing path.
3. Different rotation values at the router or NPU level alter the hash results and distribute traffic across more paths.

Result

The network uses links more evenly, which reduces traffic polarization and congestion.

Restriction for per-NPU hash rotation

Provides the deployment condition for changing hash-rotation values on a router or an individual NPU in a highly symmetric network.

Use hash rotation only when the network shows traffic polarization and underused bandwidth.

Configure per-NPU hash rotation

Configures an NPU-specific or global hash-rotation value, or uses automatic calculation, to reduce traffic polarization in a highly symmetric network.

Adjust the hashing function to improve path distribution across NPUs.

Before you begin

Enable the **hw-module profile pbr vrf-redirect** profile and reload the router.

Procedure

Select one hash-rotation configuration method.

- Configure an NPU-specific value from 1 to 35. This value has the highest priority.

```
Router# configure
Router(config)# cef load-balancing algorithm adjust 7 instance 0 location
0/0/CPU0
Router(config)# end
```

- Configure a global value for all NPUs.

```
Router# configure
Router(config)# cef load-balancing algorithm adjust 2
Router(config)# end
```

- Use automatic calculation by leaving the global and NPU-specific values unconfigured. The router derives values from the router ID, line-card ID, and NPU ID.

The default value is 1 when the required policy-based routing profile is enabled.

The hardware uses the highest-priority configured method to determine the hash-rotation value.

Verify programmed hash-rotation values

Displays the hash-rotation values programmed in hardware for every NPU on a selected line card or for one selected NPU on that card.

Confirm the hash-rotation values that the router programmed in hardware.

Procedure

Select the required verification scope.

- Display values for all NPUs on a line card.

```
Router# show controllers npu debugshell <npu-id> 'script
device_hash_rotate_info get_val_all_npu' location <line-card-node>
```

- Display the value for one NPU on a line card.

```
Router# show controllers npu debugshell <npu-id> 'script
device_hash_rotate_info get_val' location <line-card-node>
```

The command output reports the hash-rotation value programmed for the selected hardware scope.

Unified CEF error counters

Explains how a unified command consolidates FIB error counters for rapid forwarding diagnostics. Covers benefits, debugging guidance, supported hardware, and location-specific or systemwide views.

Unified Cisco Express Forwarding (CEF) error counters are a diagnostic view that

- centralizes diverse FIB error counters into a single, unified view
- exposes error conditions that affect FIB operations, and
- reduces the number of commands needed for an initial forwarding assessment.

Table 64: Feature History Table

Feature Name	Release Information	Feature Description
Unified CEF error counters	Release 25.4.1	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: Q100, Q200, P100]) Having a unified view of CEF operational health streamlines the evaluation of the FIB operations. Earlier, error counters were scattered across various CLI commands, making it difficult to monitor FIB health and pinpoint network forwarding issues. With this enhancement, all relevant metrics are consolidated into a single command, allowing you to quickly detect and resolve potential problems while greatly reducing troubleshooting time. This feature introduces these changes: CLI: **show cef global error** Yang DM: `Cisco-IOS-XR-fib-common-oper.yang` (see GitHub, YANG Data Models Navigator)

Benefits of unified CEF error counters

Describes how a consolidated FIB error view simplifies forwarding troubleshooting, improves the initial operational assessment, and reduces the number of diagnostic commands.

Use the unified counters for an initial assessment of FIB errors.

Unified CEF error counters provide these benefits:

- Simplified troubleshooting: Consolidate multiple counters in one view.
- Improved visibility: Present a quick view of FIB process errors.
- Improved efficiency: Reduce the commands needed to gather diagnostic information.

Recommendation for unified CEF error counter usage

Provides guidance for selecting the appropriate command for temporary CEF debugging or persistent system-health monitoring without creating unnecessary operational alerts.

Use **show cef global error** for debugging, not for health monitoring.

Some reported errors are temporary or noncritical. Treating every counter as a health alert can create unnecessary alerts and noise.

Use **show healthcheck metric** for health monitoring.

View unified CEF error counters

Displays a unified summary of FIB error counters for one location or all locations to support an initial forwarding diagnosis.

Inspect consolidated CEF and FIB error counters during debugging.

Procedure

1. Select the required location scope.
   - Display counters for one location.

     ```
     Router# show cef global error total location 0/RP0/CPU0
     ```

   - Display counters for all locations.

     ```
     Router# show cef global error total location all
     ```

2. Review the unified counter summary.

   Example

   ```
   FIB Objects in OOR state: 6
   FIB Objects with Platform errors: 5
   GRID IDs in OOR: 0
   FIB Next Hops with No Adjacency: 2
   CEF (Prefixes/Labels) Unresolved: 2
   MPLS Local-Label Conflicts: 0
   Encap-ID Conflicts: 1
   LDI/NHG/ECMP-FEC DROP: 0
   FIB Has Objects in Unsuccessful Retry: Yes
   ```

The output summarizes FIB resource, platform, adjacency, resolution, label, encapsulation, and retry errors for the selected locations.

12 Local Packet Transport Services

Topics:

Explains LPTS packet delivery, flow capacity, policers, NPU traps, and control and management plane ACLs. Provides configuration, monitoring, verification, tuning, and traffic-filtering guidance for protecting router-bound traffic.

Local Packet Transport Services (LPTS) is a Cisco IOS XR subsystem that

- delivers packets that are destined for router applications to the correct route processor
- protects control plane resources through flow policers and Network Processing Unit (NPU) trap policers, and
- supports flow-capacity controls and ACLs for router-bound traffic.

This chapter describes LPTS components, dynamic flows, policers, NPU traps, and user-managed control and management plane ACLs.

LPTS components and packet delivery

Describes the port arbitrator, flow managers, and IFIB that identify router-bound packet flows and deliver accepted packets to the correct route processor for application processing.

Local Packet Transport Services (LPTS) components maintain the packet-flow information that

- identifies traffic destined for the secure domain router
- associates accepted traffic with the intended router application, and
- delivers packets to the correct route processor.

LPTS operates without required customer configuration. You can customize policer values and use operational commands to monitor flow managers and the port arbitrator.

Primary LPTS components

LPTS uses these components:

- Port arbitrator: Maintains packet-flow tables for a logical router.
- Flow managers: Coordinate with applications that receive packets from outside the router and maintain flow entries.
- Internal Forwarding Information Base (IFIB): Describes packet flows and directs accepted packets to the correct route processor.

How LPTS packet delivery works

Describes how LPTS identifies an incoming router-bound flow, applies the programmed traffic treatment, uses IFIB information to select its destination, and delivers an accepted packet for application processing.

Router applications receive control and management traffic from outside the router. Local Packet Transport Services (LPTS) maintains the information required to identify and deliver this traffic securely.

Summary

The packet-delivery process uses these components:

- Flow managers: Maintain flow information for applications.
- Port arbitrator: Maintains packet-flow tables for the logical router.
- Internal Forwarding Information Base (IFIB): Identifies the destination for an accepted packet.

Workflow

These stages describe how LPTS packet delivery works:

1. The router examines an incoming packet that is destined for a router application and identifies its flow type from packet-header information.
2. The programmed LPTS treatment determines whether the packet is accepted or dropped.
3. For an accepted packet, LPTS uses the IFIB entry to select the correct route processor and application destination.
4. The route processor receives the packet for application processing.

Result

Accepted router-bound packets reach their intended applications without allowing excess traffic to consume route processor resources.

LPTS flow types and TCAM capacity

Explains how configurable LPTS flow types use per-line-card TCAM capacity, how maximum entry values enable or disable flow types, and how to configure and verify dynamic flow limits.

Dynamic Local Packet Transport Services (LPTS) flow configuration is a per-line-card capability that

- selects configurable flow types for programming in ternary content-addressable memory (TCAM)
- sets the maximum number of entries for each selected flow type, and
- allows different flow-capacity profiles across line cards.

The router programs default LPTS flow types and limits in TCAM during startup. A configured maximum of zero disables a configurable flow type.

TCAM capacity controls

Use **show lpts pifib dynamic-flows statistics location** to view configurable flow types, default limits, configured limits, hardware entries, software entries, and pending software entries.

- Def_Max: Default maximum entry limit.
- Conf_Max: Configured maximum entry limit.
- HWCnt/ActLimit: Hardware entry count and actual maximum limit.
- SWCnt: Software entry count.
- P, (+): Pending software entries.

Important

The combined maximum for all configured flow types must not exceed 16,000 entries per line card.

Hardware entry planning

Set the dynamic scale so that entries for the flow type remain in hardware. Adequate hardware capacity can help prevent session flaps for protocols such as BGP and OSPF during events such as route processor failover.

LPTS flow types

From IOS XR Release 26.3.1 onwards, the router supports these flow types:

- IFIB_FT_IPSEC_KNOWN
- FIB_FT_NTP_DEFAULT
- IFIB_FT_TCP_LISTEN
- IFIB_FT_PCEP
- IFIB_FT_RADIUS
- IFIB_FT_TACACS
- IFIB_FT_RIP
- IFIB_FT_EIGRP
- IFIB_FT_SHTTP_DEFAULT

- IFIB_FT_HTTP_DEFAULT

Configure the dynamic LPTS flow limits

Configures maximum TCAM entries for selected LPTS flow types on a line card, disables a flow type when capacity is needed, and verifies hardware and software entry usage.

Use this procedure to adjust per-line-card capacity for configurable Local Packet Transport Services (LPTS) flow types.

Before you begin

Use **show lpts pifib dynamic-flows statistics location** to record the current limits and available ternary content-addressable memory (TCAM) capacity.

Confirm that the combined maximum values do not exceed the capacity reported for the line card.

Procedure

1. Configure the maximum entry value for each required flow type at the target location. Set a configurable flow type to zero when you must disable it to release capacity.

   Example

   ```
   Router# configure
   Router(config)# lpts pifib hardware dynamic-flows location 0/1/CPU0
   Router(config-pifib-flows-per-node)# flow bgp known max 1800
   Router(config-pifib-flows-per-node)# flow rsvp known max 0
   Router(config-pifib-flows-per-node)# commit
   ```

   The line card reserves up to 1,800 entries for the BGP-known flow type and disables the RSVP-known flow type.

2. Verify the committed dynamic-flow configuration.

   Example

   ```
   Router# show running-config lpts pifib hardware dynamic-flows location 0/1/CPU0
   lpts pifib hardware dynamic-flows location 0/1/CPU0
    flow bgp known max 1800
    flow rsvp known max 0
   !
   ```

3. Verify the configured limits and entry counts for the location.

   Example

   ```
   Router# show lpts pifib dynamic-flows statistics location 0/1/CPU0
   Dynamic-flows Statistics:
    -------------------------
    (C - Configurable, T - TRUE, F - FALSE, * - Configured)
    Def_Max  - Default Max Limit
    Conf_Max - Configured Max Limit
    HWCnt    - Hardware Entries Count
    ActLimit - Actual Max Limit
    SWCnt    - Software Entries Count
    P, (+)   - Pending Software Entries


     FLOW-TYPE           C  Def_Max Conf_Max     HWCnt/ActLimit      SWCnt  P
   ```

```
 -------------------- -- ------- --------   -------/--------    -------- -
 ..
 OSPF-uc-default      F        0       --         0/0                 1  +
 BFD-default          F        2       --         2/2                 2
 BFD-MP-known         T       40       --         1/40                0
 BGP-known            T*    2400     1800         6/900               6
 ..
 RSVP-known           T*     300        0         0/0                 1  +
 SNMP                 T      300       --         8/300               8
 SSH-known            T       40       --         0/40                0
 SSH-default          T        1       --         1/1                 2  +
 HTTP-known           T       40       --         0/40                0
 ..
-----------------------------------------------------
 Active TCAM Usage : 13421/16000 [Platform MAX: 16000]
 HWCnt/SWCnt       : 65/88
-----------------------------------------------------
```

An asterisk identifies a configured flow type. Confirm that BGP-known reports the configured maximum and that RSVP-known reports an actual limit of zero.

The target line card uses the configured dynamic-flow limits without exceeding its reported TCAM capacity.

LPTS policers

Explains how ingress policers protect the route processor by rate-limiting control traffic by flow type. Covers global and per-node configuration, effective hardware rates, CLI verification, and YANG statistics.

Local Packet Transport Services (LPTS) policers are ingress traffic controls that

- classify route-processor-bound control packets by flow type
- apply statically programmed rates at incoming ports, and
- allow rate customization globally or for a specific node.

LPTS programs default policer rates during startup. Hardware optimization can produce an effective rate that is close to, but different from, the configured rate without affecting policer operation.

Table 65: Feature History Table

Feature Name	Release Information	Description
Monitor LPTS Host Path Drops via YANG Data Model	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

Feature Name	Release Information	Description
Monitor LPTS Host Path Drops via YANG Data Model	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
Monitor LPTS Host Path Drops via YANG Data Model	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Monitor LPTS Host Path Drops via YANG Data Model	Release 7.3.2	This feature allows you to use the `Cisco-IOS-XR-lpts-pre-ifib-oper.yang` data model to monitor the policer action for Local Packet Transport Services (LPTS) flow type for all IOS XR platforms. To access this data model, see the Github repository.

Effective policer rates

Use **show lpts pifib hardware police** to compare current hardware rates with configured rates and to review accepted and dropped packet counts.

Note

The hardware can program a rate that differs from the requested value. The programmed rate is optimized for hardware performance and remains close to the configured value.

Configure the global LPTS policer rates

Configures OSPF and BGP LPTS policer rates for all nodes and verifies the committed configuration and effective hardware rates for the selected flow types.

Use global Local Packet Transport Services (LPTS) policer configuration when every node requires the same flow-type rates.

Procedure

1. Configure and commit the global policer rates for the required flow types.

   Example

```
Router# configure
Router(config)# lpts pifib hardware police
Router(config-lpts-policer-global)# flow ospf unicast default rate 3000
Router(config-lpts-policer-global)# flow bgp default rate 4000
Router(config-lpts-policer-global)# commit
```

2. Verify the committed global configuration.

   Example

```
Router# show running-config lpts
lpts pifib hardware police
 flow ospf unicast default rate 3000
 flow bgp default rate 4000
!
```

3. Verify the effective hardware rates and packet counters.

   Example

```
Router# show lpts pifib hardware police
-------------------------------------------------------------
                Node 0/RP0/CPU0:
-------------------------------------------------------------
FlowType              Policer Type    Cur. Rate Burst     Accepted     Dropped
      npu
---------------------- ------- ------- --------- --------- ------------
------------ ---------
Fragment               2       np      542       1000      0            0
         0
OSPF-mc-known          3       np      1627      1000      0            0
         0
OSPF-mc-default        4       np      1084      1000      0            0
         0
OSPF-uc-known          5       np      542       1000      0            0
         0
OSPF-uc-default        6       np      2878      1000      0            0
         0
BFD-default            10      np      8136      1000      0            0
         0
BFD-MP-known           11      np      8136      1000      0            0
         0
BGP-known              16      np      17000     1000      0            0
         0
BGP-cfg-peer           17      np      1627      1000      0            0
         0
BGP-default            18      np      3880      1000      0            0
         0
```

Confirm that OSPF-uc-default and BGP-default have effective rates close to the configured values. Hardware rate optimization can cause a small difference.

All nodes use the global LPTS policer configuration for the selected flow types.

Configure LPTS policer rates for a node

Configures OSPF and BGP LPTS policer rates for one node and verifies the node-specific running configuration, effective per-NPU rates, and packet counters.

Use location-specific Local Packet Transport Services (LPTS) configuration when a node requires rates that differ from the global policer configuration.

Before you begin

Identify the node location where the policer rates must apply.

Procedure

---

1. Configure the policer rates at the target node.

   Example

   ```
   Router# configure
   Router(config)# lpts pifib hardware police location 0/0/CPU0
   Router(config-lpts-policer-local)# flow ospf unicast default rate 3000
   Router(config-lpts-policer-local)# flow bgp default rate 4000
   Router(config-lpts-policer-local)# commit
   ```

2. Verify the location-specific running configuration.

   Example

   ```
   Router# show running-config lpts
   lpts pifib hardware police location 0/0/CPU0
    flow ospf unicast default rate 3000
    flow bgp default rate 4000
   !
   ```

   The output lists the OSPF and BGP rates under the policer configuration for location 0/0/CPU0.

3. Verify the effective rates and counters for each Network Processing Unit (NPU) at the node.

   Example

   ```
   Router# show lpts pifib hardware police location 0/0/CPU0
   ------------------------------------------------------------
                  Node 0/0/CPU0:
   ------------------------------------------------------------
   FlowType              Policer Type    Cur. Rate Burst     Accepted     Dropped
         npu
   --------------------- ------- ------- --------- --------- ------------
   ------------ ---------
   Fragment                 2       np      542       1000      0            0
           0
   Fragment                 2       np      542       1000      0            0
   ```

```
          1
OSPF-mc-known           3       np      1627      1000      0             0
          0
OSPF-mc-known           3       np      1627      1000      0             0
          1
OSPF-mc-default         4       np      1084      1000      0             0
          0
OSPF-mc-default         4       np      1084      1000      0             0
          1
OSPF-uc-known           5       np      542       1000      0             0
          0
OSPF-uc-known           5       np      542       1000      0             0
          1
OSPF-uc-default         6       np      2878      1000      0             0
          0
OSPF-uc-default         6       np      2878      1000      0             0
          1
BFD-default             10      np      8136      1000      0             0
          0
BFD-default             10      np      8136      1000      0             0
          1
BFD-MP-known            11      np      8136      1000      0             0
          0
BFD-MP-known            11      np      8136      1000      0             0
          1
BGP-known               16      np      17000     1000      0             0
          0
BGP-known               16      np      17000     1000      0             0
          1
BGP-cfg-peer            17      np      1627      1000      0             0
          0
BGP-cfg-peer            17      np      1627      1000      0             0
          1
BGP-default             18      np      3880      1000      0             0
          0
BGP-default             18      np      3880      1000      0             0
          1
```

Confirm that OSPF-uc-default and BGP-default appear for each NPU and have effective rates close to the configured values.

The selected node uses its location-specific LPTS policer rates.

YANG data model fields for LPTS policer statistics

Describes the operational YANG request for LPTS hardware flow-policer statistics, the returned policer fields, and the aggregate NPU identifier used to summarize counters by flow type.

Use this reference to retrieve and interpret Local Packet Transport Services (LPTS) flow-type policer statistics through NETCONF.

Cisco IOS XR Release 7.3.2 and later supports the `Cisco-IOS-XR-lpts-pre-ifib-oper.yang` operational data model across Cisco IOS XR platforms.

This sample request retrieves hardware flow-policer statistics for node 0/0/CPU0:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
```

```
      <lpts-pifib
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-lpts-pre-ifib-oper">
        <nodes>
          <node>
            <node-name>0/0/CPU0</node-name>
            <pifib-hw-flow-policer-stats/>
          </node>
        </nodes>
      </lpts-pifib>
    </filter>
  </get>
</rpc>
```

The response can include these values for each flow type:

- flow-type and flow-name: Numeric and descriptive flow identifiers.
- type and type-name: Policer type identifiers.
- domain-id and domain-name: LPTS domain identifiers.
- npu-id: Network Processing Unit (NPU) identifier for the statistics.
- policer-rate and burst-size: Programmed traffic-control values.
- accepted and dropped: Packet counters.

This is a sample response snippet:

```
<police-info>
       <flow-type>23</flow-type>
       <flow-name>ICMP-local</flow-name>
       <type>2</type>
       <type-name>Global</type-name>
       <domain-id>0</domain-id>
       <domain-name>default</domain-name>
       <npu-id>255</npu-id>
       <policer-rate>0</policer-rate>
       <burst-size>750</burst-size>
       <accepted>2000</accepted>
       <dropped>1000</dropped>
      </police-info>
      <police-info>
```

An NPU ID of 255 represents the aggregate of all NPU counters. This aggregate provides one policer-statistics view for each flow type.

NPU traps

Explains the exception, error, and non-LPTS control packets represented by NPU traps. Covers per-NPU policing, accepted and dropped counters, trap-statistics monitoring, and configurable trap policers.

Network Processing Unit (NPU) traps are packet classifications that

- identify exception packets such as glean-adjacency traffic and packets with IPv4 options
- identify error packets such as packets with an invalid checksum or hop count, and
- identify control packets such as LACP and LLDP that do not pass through LPTS flow processing.

NPU traps complement Local Packet Transport Services (LPTS) flow policing. LPTS protects recognized router-bound flows, whereas NPU traps classify exception, error, and non-LPTS control traffic for NPU-level handling.

Trap-statistics indicators

Trap-statistics output uses these indicators:

- (D): The NPU drops the trap packet.
- (D*): The NPU drops the trap packet but retains statistics for analysis.
- Accepted: The packet is available for additional processing or analysis.
- Dropped: The packet exceeded the applicable treatment or policer rate.

How NPU trap policing works

Describes how an NPU classifies trapped packets, applies a per-trap policer, forwards accepted packets that require additional processing, and drops excess traffic before it reaches CPU resources.

A Network Processing Unit (NPU) raises traps for packets that require exception handling, contain errors, or represent control traffic that does not use Local Packet Transport Services (LPTS) flow processing.

Summary

The process uses these elements:

- Trap classification: Identifies the reason for the exception or punt.
- Trap policer: Limits traffic for the trap on each NPU.
- Processing destination: Receives accepted packets that require additional handling.

Workflow

These stages describe how NPU trap policing works:

1. The NPU classifies an incoming packet and raises the applicable trap.
2. The per-NPU trap policer compares the packet rate with the programmed rate for that trap.
3. The NPU accepts packets that are within the policer rate and sends packets that require additional processing toward the CPU.
4. The NPU drops excess packets according to the system design and increments the dropped counter.

Result

Trap policing protects CPU resources while preserving accepted exception and control packets for required processing or analysis.

Monitor the NPU trap statistics

Displays trap statistics for one or all NPUs at a route processor or line-card location and identifies accepted packets, dropped packets, policer rates, and trap dispositions.

Use trap statistics to identify packet exceptions, drops, and policer activity at a Network Processing Unit (NPU) location.

Before you begin

Identify the route processor or line-card location and the NPU instance to monitor.

Procedure

1. Display trap statistics for one NPU or all NPUs at the selected location.

   Use location 0/RP0/CPU0 for a fixed system. Use a line-card location, such as 0/1/CPU0, for a distributed system.

   Example

```
Router# show controllers npu stats traps-all instance all location 0/RP0/CPU0
Trap Type                           NPU  Trap TrapStats  Policer Policer
Packet    Packet

                                    ID   ID   ID                 Rate
Accepted  Dropped
=============================================================================
ETHERNET_ACL_DROP(D)                0     0    0x0        1       0
0         0
ETHERNET_ACL_FORCE_PUNT(D*)         0     1    0x0        1       0
0         0
ETHERNET_VLAN_MEMBERSHIP(D*)        0     2    0x0        1       0
0         0
ETHERNET_ACCEPTABLE_FORMAT          0     3    0x0        258     100
0         0
UNKNOWN_VLAN_OR_BUNDLE_MEMBER(D*)   0     4    0x0        259     100
0         0
NOT_MY_MAC(D*)                      0     5    0x0        260     100
0         0
..                                  .     .    .          .       .
.         .
ETHERNET_SA_MULTICAST(D*)           0     13   0x0        268     100
0         0
DHCPV4_SERVER                       0     14   0x0        269     542
0         0
DHCPV4_CLIENT                       0     15   0x0        270     200
0         0
ETHERNET_INGRESS_STP_BLOCK(D*)      0     18   0x0        1       0
0         0
PTP_OVER_ETHERNET                   0     19   0x0        274     4000
0         0
..                                  .     .    .          .       .
.         .
OAMP_PFC_DROP_INVALID_RX(D*)        0    166   0x0        1       0
0         0
APP_SGACL_DROP(D*)                  0    168   0x0        1       0
0         0
```

2. Review the trap type, NPU ID, policer rate, accepted count, and dropped count.

   Entries marked (D) are dropped by the NPU. Entries marked (D*) are dropped but remain available for statistics analysis.

3. Clear trap counters when you need a new measurement interval.

   Example

```
Router# clear controller npu stats traps-all instance all location 0/RP0/CPU0
```

   Subsequent output reports trap activity accumulated after the counters were cleared.

The output identifies the traps and NPUs that accept or drop packets during the measurement interval.

Configurable trap policers

Explains per-trap rate and average-packet-size controls for adapting control plane protection to observed traffic. Includes supported values, tuning guidance, configuration, and trap-statistics verification.

Configurable trap policers are control plane protection settings that

- set a packet-per-second rate for an individual trap
- define the expected average packet size for that trap, and
- allow traffic treatment to be tuned for deployment-specific traffic patterns.

Per-trap controls help prevent excessive or burst traffic from overwhelming system resources while preserving legitimate protocol and exception traffic.

Table 66: Feature History Table

Feature Name	Release Information	Description
Configurable trap policers	Release 26.2.1	Introduced in this release on: Fixed Systems(8200, 8700)(select variants only*); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q200, P100]) Configurable trap policers help prevent unnecessary packet drops, improve control-plane traffic handling, and support efficient scaling across diverse deployment scenarios. Per-trap rate limiting and average packet size tuning provide precise control over control-plane traffic.

Operational benefits

Per-trap configuration provides these operational benefits:

- Reduces unintended drops of legitimate control plane traffic.
- Adjusts individual trap settings without waiting for system-wide default changes.
- Supports different traffic profiles across large-scale deployments.
- Uses trap statistics to refine rate and packet-size settings.

Trap policer parameters

Lists the configurable rate and average-packet-size parameters for NPU trap policers, their valid ranges, and the effect of using either or both values.

Use these values when you configure a policer for an individual Network Processing Unit (NPU) trap.

The table lists the supported trap policer parameters and ranges.

Table 67: Trap policer configuration parameters

Parameter	Description	Configurable value
rate	Sets the number of packets allowed each second.	0–50,000 packets per second
avg-pkt-size	Sets the expected average size of packets for the trap.	64–65,535 bytes

You can configure the rate, the average packet size, or both. Configuring both values aligns policing more closely with the observed packet rate and size.

Best practice: Tune trap policers from observed traffic

Provides recommendations for setting conservative per-trap rates, using average packet size when traffic varies, monitoring trap counters, and revisiting settings after traffic or topology changes.

Recommendation:

Ensure that you tune trap policers based on actual traffic patterns to avoid unnecessary packet drops.

- Start with conservative rate values and increase gradually when statistics show that legitimate traffic is being dropped.
- Configure **avg-pkt-size** when traffic characteristics vary significantly from the default traffic profile.
  - Tune traps individually instead of applying one rate to every trap.
  - Use regular trap-statistics measurements to validate the settings.
- Monitor trap statistics regularly to validate configuration effectiveness.
- Adjust per trap instead of applying uniform settings across all traps.
- Revisit configurations after topology or traffic changes.

Configure a trap policer

Configures a rate and optional average packet size for a specific NPU trap at a node location and verifies the configured and hardware rates through trap statistics.

Use this procedure to adapt an individual Network Processing Unit (NPU) trap policer to the traffic profile at a node.

Before you begin

Identify the trap name and node location.

Record the current accepted and dropped counters before changing the policer.

Procedure

1. Configure and commit the packet rate for the trap at the target location.

   Example

   ```
   Router# configure
   Router(config)# lpts punt police location 0/0/CPU0
   Router(config-lpts-policer)# exception isis rate 100
   Router(config-lpts-policer)# commit
   ```

2. Optionally configure and commit the expected average packet size with the trap rate.

   Example

   ```
   Router# configure
   Router(config)# lpts punt police location 0/0/CPU0
   Router(config-lpts-policer)# exception isis rate 100 avg-pkt-size 512
   Router(config-lpts-policer)# commit
   ```

3. Verify the configured rate, hardware rate, average packet size, and packet counters.

Example

```
Router# show controllers npu stats traps-all instance all location 0/0/CPU0
```

Confirm that the IS-IS trap reports the expected configured rate and average packet size. Use accepted and dropped counters to evaluate the setting.

The selected trap uses the configured per-location policer values.

What to do next

Monitor trap statistics over a representative traffic interval and adjust the values when legitimate traffic is dropped.

Control and management plane ACLs

Explains how a virtual LPTS interface and compressed hybrid ACLs filter and police control and management plane traffic. Covers limitations, best practices, configuration, verification, and IPv6 AH and ESP support.

User-managed control and management plane access control lists (ACLs) are traffic-security controls that

- create a virtual Local Packet Transport Services (LPTS) interface for router-bound traffic
- apply compressed hybrid IPv4 and IPv6 ACLs to that interface, and
- filter, police, prioritize, and log control and management plane traffic.

The data plane forwards user packets between interfaces. The control plane determines packet paths through routing protocols and related processes. The management plane provides device configuration and monitoring functions.

Table 68: Feature History Table

Feature Name	Release Information	Description
User Managed Control Plane and Management Plane ACL	Release 7.3.3 Release 7.5.2	You can create a virtual LPTS interface and apply hybrid ACLs to it for inspecting traffic. This functionality lets you use the hybrid ACLs to filter and customize the control plane and management plane traffic. This feature modifies the following command: • **hw-module profile cef**

LPTS ACL mode

Use **hw-module profile cef lpts acl** to enable LPTS ACL mode. Apply one IPv4 ACL and one IPv6 ACL to the virtual LPTS interface with compression level 2.

Use **no hw-module profile cef lpts acl** to disable LPTS ACL mode.

Control and management plane ACL limitations

Lists BFD offload behavior, object-group compression requirements, unsupported protocol and port combinations, and reload requirements that affect user-managed control and management plane ACLs.

Review these limitations before you apply an access control list (ACL) to the virtual Local Packet Transport Services (LPTS) interface.

User-managed control and management plane ACLs have these limitations:

- Use only object groups with compression level 2 in LPTS ACL mode.
- Do not rely on these ACLs to filter hardware-offloaded BFD control packets.
- An ACL lookup applies only during initial BFD session setup when the remote discriminator is zero. After the system learns the discriminator, established-session packets do not use this lookup.
- A deny ACE with logging does not drop hardware-offloaded BFD packets. A deny ACE without logging can drop packets during initial session setup.
- Reload the router after you add AH or ESP entries to an IPv6 ACL.

Do not use these protocol and port combinations to permit or deny packets in this ACL mode:

Table 69: Unsupported protocol and port combinations

Protocol	Ports
PTP	UDP ports 319 and 320
BFD	UDP ports 3784 and 3785
IPLSA	TCP port 1167 and UDP ports 1167 and 1967
TWAMP	TCP port 862

To deny traffic that uses an unsupported protocol and port combination, use a general **deny ipv4 any any** or **deny ipv6 any any** ACE.

Best practice: Protect control and management plane traffic

Provides recommendations for using one virtual LPTS interface, compressed hybrid ACLs, object groups, policer and priority actions, logging, and deny-packet handling to protect router-bound traffic.

Recommendation: Use one virtual LPTS interface

Create one virtual Local Packet Transport Services (LPTS) interface and apply the required IPv4 and IPv6 hybrid access control lists (ACLs) to it.

- Create all required object groups before you apply an ACL to the interface.
- Apply each object-group ACL with compression level 2.

This recommendation applies when user-managed ACLs filter IPv4 or IPv6 control and management plane traffic.

One virtual interface provides a consistent attachment point for policies that protect router-bound traffic.

The router evaluates control and management plane traffic through the intended compressed ACLs.

Verify the hardware ACL entries on interface `lpts0` after you apply the configuration.

Recommendation: Use explicit traffic actions

Apply policer, priority, and logging actions to the access control entries (ACEs) that require traffic control or operational visibility.

- Enable logging on relevant ACEs to monitor accepted or denied traffic.

- Use the **icmp-on** option to punt packets that match deny ACEs. Without this option, the router drops the packets by default.

This recommendation applies when an ACL must rate-limit, prioritize, log, or punt matched traffic.

Explicit ACE actions define the required traffic treatment and provide counters for operational verification.

The ACL controls matched traffic and records the information required for operational review.

Use hardware ACL statistics to confirm the accepted and dropped packet counts for each relevant ACE.

Configure an ACL for control and management plane traffic

Enables LPTS ACL mode, creates compressed IPv4 and IPv6 hybrid ACLs, applies them to the virtual LPTS interface, reloads the router, and verifies hardware policy status and counters.

Use this procedure to configure an access control list (ACL) that filters, polices, and prioritizes router-bound IPv4 and IPv6 traffic.

Before you begin

Create the network object groups referenced by the ACL entries.

Review the unsupported protocol and port combinations before you create the ACLs.

> **Warning**
>
> Enabling LPTS ACL mode and applying the policy requires a router reload. Schedule a maintenance window before you begin.

Procedure

1. Enable Local Packet Transport Services (LPTS) ACL mode.

   Example

```
Router# configure
Router(config)# hw-module profile cef lpts acl
Router(config)# commit
```

2. Create the IPv4 hybrid ACL.

   Example

```
Router(config)# ipv4 access-list test-umpp-v4-filter
Router(config-ipv4-acl)# 10 permit icmp net-group ACL_GROUP_1 any police 67
pps
Router(config-ipv4-acl)# 20 permit icmp net-group ACL_GROUP_2 any priority
Medium
Router(config-ipv4-acl)# 30 permit icmp net-group ACL_GROUP_3 any priority
High
Router(config-ipv4-acl)# 40 permit icmp net-group ACL_GROUP_4 any police 100
 pps
Router(config-ipv4-acl)# 50 permit icmp any any 0
Router(config-ipv4-acl)# 60 permit icmp any any 3
```

```
Router(config-ipv4-acl)# priority-timeout 25
Router(config-ipv4-acl)# commit
```

3. Create the IPv6 hybrid ACL.

   Example

```
Router(config)# ipv6 access-list test-umpp-v6-filter
Router(config-ipv6-acl)# 10 permit icmpv6 net-group ACL_GROUP_1 any priority
 Medium
Router(config-ipv6-acl)# 20 permit icmpv6 net-group ACL_GROUP_2 any police
67 pps
Router(config-ipv6-acl)# 30 permit icmpv6 net-group ACL_GROUP_5 any priority
 Low
Router(config-ipv6-acl)# 40 permit icmpv6 net-group ACL_GROUP_6 any police
100 pps
Router(config-ipv6-acl)# 50 permit icmpv6 any any echo
Router(config-ipv6-acl)# 60 permit icmpv6 any any echo-reply
Router(config-ipv6-acl)# priority-timeout 25
Router(config-ipv6-acl)# commit
```

4. Apply both ACLs to the virtual LPTS interface with compression level 2.

   Example

```
Router(config)# interface lpts 0
Router(config-if)# ipv4 access-group test-umpp-v4-filter ingress compress
level 2
Router(config-if)# ipv6 access-group test-umpp-v6-filter ingress compress
level 2
Router(config-if)# commit
```

5. Reload the router to activate LPTS ACL mode.

   Example

```
Router# reload
```

6. Verify that LPTS ACL mode is applied and review the hardware ACL counters.

   Example

```
Router# show hw-module profile cef
-----------------------------------------------------------------
Knob                          Status          Applied   Action
-----------------------------------------------------------------
CBF                           Unconfigured    N/A       None
BGPLU                         Unconfigured    N/A       None
LPTS ACL                      Configured      Yes       None
..
```

```
Router# show access-lists test-umpp-v4-filter hardware ingress interface lpts
 0 location 0/RP0/CPU0
10 permit icmp net-group ACL_GROUP_1 any police 67 pps (Accepted: 14 packets,
```

```
 Dropped: 0 packets)
20 permit icmp net-group ACL_GROUP_2 any priority Medium
30 permit icmp net-group ACL_GROUP_3 any priority High
40 permit icmp net-group ACL_GROUP_4 any police 100 pps (Accepted: 25 packets,
 Dropped: 0 packets)
50 permit icmp any any 0
60 permit icmp any any 3
```

```
Router# show access-lists ipv6 test-umpp-v6-filter hardware ingress interface
 lpts 0 location 0/RP0/CPU0
10 permit icmp net-group ACL_GROUP_1 any priority Medium
20 permit icmp net-group ACL_GROUP_2 any police 67 pps (Accepted: 3 packets,
 Dropped: 0 packets)
30 permit icmp net-group ACL_GROUP_5 any priority Low
40 permit icmp net-group ACL_GROUP_6 any police 100 pps (Accepted: 35 packets,
 Dropped: 0 packets)
50 permit icmp any any echo
60 permit icmp any any echo-reply
```

The profile output reports LPTS ACL as configured and applied. The ACL output reports accepted and dropped packets for ACEs with policer actions.

The virtual LPTS interface filters control and management plane traffic through the configured IPv4 and IPv6 ACLs.

AH and ESP headers in IPv6 ACLs

Explains how IPv6 ACL entries match AH and ESP traffic on the virtual LPTS interface, the security purpose of each header, supported releases and platforms, configuration, reload, and verification.

Authentication Header (AH) and Encapsulating Security Payload (ESP) support in user-managed IPv6 access control lists (ACLs) is an IPsec traffic-filtering capability that

- matches Authentication Header traffic with the **ahp** protocol keyword
- matches Encapsulating Security Payload traffic with the **esp** protocol keyword, and
- permits or denies matched packets at the virtual LPTS interface.

AH provides data integrity and data-origin authentication. ESP provides data confidentiality.

Table 70: Feature History Table

Feature Name	Release Information	Description
Authentication Header (AH) and Encapsulating Security Payload (ESP) Headers Support in User Managed Control Plane and Management Plane ACLs	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

Feature Name	Release Information	Description
Authentication Header (AH) and Encapsulating Security Payload (ESP) Headers Support in User Managed Control Plane and Management Plane ACLs	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
Authentication Header (AH) and Encapsulating Security Payload (ESP) Headers Support in User Managed Control Plane and Management Plane ACLs	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Authentication Header (AH) and Encapsulating Security Payload (ESP) Headers Support in User Managed Control Plane and Management Plane ACLs	Release 7.10.1	This feature enhances traffic security by introducing the Authentication Header (AH) and Encapsulating Security Payload (ESP) IPv6 headers in the IPv6 ACLs. While AH provides data integrity and data origin authentication, ESP is for data confidentiality. You can configure ingress IPv6 ACL extensions for AH and ESP headers to permit or deny packets. These protocols ensure that the sensitive information travelling on the network reaches its destination safely.

Activation requirement

Important

Reload the router after you add AH or ESP entries to an IPv6 ACL that is applied to the virtual Local Packet Transport Services (LPTS) interface.

Configure an IPv6 ACL for AH and ESP headers

Creates IPv6 ACL entries that permit AH and ESP packets, applies the ACL to the virtual LPTS interface with compression level 2, reloads the node, and verifies hardware counters.

Use this procedure to permit Authentication Header (AH) and Encapsulating Security Payload (ESP) packets through a user-managed control and management plane IPv6 access control list (ACL).

Before you begin

Confirm that the router release and platform support AH and ESP protocol matching in IPv6 ACLs.

> **Warning**
>
> This configuration requires a node reload. Schedule a maintenance window before you begin.

Procedure

1. Enable Local Packet Transport Services (LPTS) ACL mode if it is not already enabled.

   Example

   ```
   Router# configure
   Router(config)# hw-module profile cef lpts acl
   Router(config)# commit
   ```

2. Create IPv6 ACL entries that permit AH and ESP packets.

   Example

   ```
   Router(config)# ipv6 access-list ipv6_umpp_access_list
   Router(config-ipv6-acl)# 12 permit ahp any any
   Router(config-ipv6-acl)# 14 permit esp any any
   Router(config-ipv6-acl)# commit
   ```

3. Apply and commit the IPv6 ACL to the virtual LPTS interface with compression level 2.

   Example

   ```
   Router(config)# interface lpts0
   Router(config-if)# ipv6 access-group ipv6_umpp_access_list ingress compress
   level 2
   Router(config-if)# commit
   ```

4. Reload the node to activate AH and ESP matching.

   Example

   ```
   Router# reload location 0/0/CPU0
   ```

5. Verify that the hardware ACL permits AH and ESP packets.

   Example

   ```
   Router# show access-lists ipv6 ipv6_umpp_access_list hardware ingress interface
    lpts0 location 0/RP0/CPU0
   ipv6 access-list ipv6_umpp_access_list
   12 permit ahp any any (Accepted: 246524 packets, Dropped: 0 packets)
   14 permit esp any any (Accepted: 246524 packets, Dropped: 0 packets)
   ```

   The output lists the **permit ahp** and **permit esp** entries with accepted and dropped packet counters.

The virtual LPTS interface permits AH and ESP packets that match the configured IPv6 ACL entries.

13 Implement VRRP

Topics:

Describes how to implement the Virtual Router Redundancy Protocol (VRRP) to enable transparent failover at the first-hop IP router by allowing a group of routers to act as a single virtual router.

VRRP

This topic describes the Virtual Router Redundancy Protocol (VRRP), which allows a group of routers to form a single virtual router and provides transparent failover at the first-hop IP router.

To implement VRRP on Cisco IOS XR software, you need to understand the following concepts:

- Multiple virtual router support
- VRRP router priority
- VRRP advertisements
- Benefits of VRRP
- Hot restartability for VRRP

Table 71: Feature History Table

Feature Name	Release Information	Feature Description
Virtual Router Redundancy Protocol	Release 26.3.1	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.
Virtual Router Redundancy Protocol	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
Virtual Router Redundancy Protocol	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is supported on: • 8011-4G24Y4H-I
Enabling Virtual Router Redundancy Protocol over BVI	Release 25.1.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) You can now enable Virtual Router Redundancy Protocol (VRRP) over BVI on Cisco Silicon One P100 ASIC-based systems. This feature, enabled by default, ensures continuous network availability by automatically switching to a standby router if the active router fails. This minimizes downtime and maintains network reliability. *The feature is supported on: • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Feature Name	Release Information	Feature Description
Virtual Router Redundancy Protocol	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 improve network reliability by enabling the Virtual Router Redundancy Protocol (VRRP) to automatically assign available IP routers to participating hosts, which enhances reliability. This improvement reduces downtime by allowing a backup router to take over if the primary router fails, ensuring continuous network availability. VRRP also supports load balancing, distributing traffic across multiple routers for better efficiency. It integrates seamlessly with existing network configurations, offering a streamlined setup process without compromising security or performance. This feature is especially beneficial for critical applications that require high availability. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

A LAN client can use a dynamic process or static configuration to determine which router should be the first hop to a particular remote destination. The client examples of dynamic router discovery are as follows:

- Proxy ARP—The client uses Address Resolution Protocol (ARP) to get the destination it wants to reach, and a router responds to the ARP request with its own MAC address.
- Routing protocol—The client listens to dynamic routing protocol updates (for example, from Routing Information Protocol [RIP]) and forms its own routing table.
- IRDP (ICMP Router Discovery Protocol) client—The client runs an Internet Control Message Protocol (ICMP) router discovery client.

The drawback to dynamic discovery protocols is that they incur some configuration and processing overhead on the LAN client. Also, in the event of a router failure, the process of switching to another router can be slow.

An alternative to dynamic Cisco Discovery Protocols is to statically configure a default router on the client. This approach simplifies client configuration and processing, but creates a single point of failure. If the default gateway fails, the LAN client is limited to communicating only on the local IP network segment and is cut off from the rest of the network.

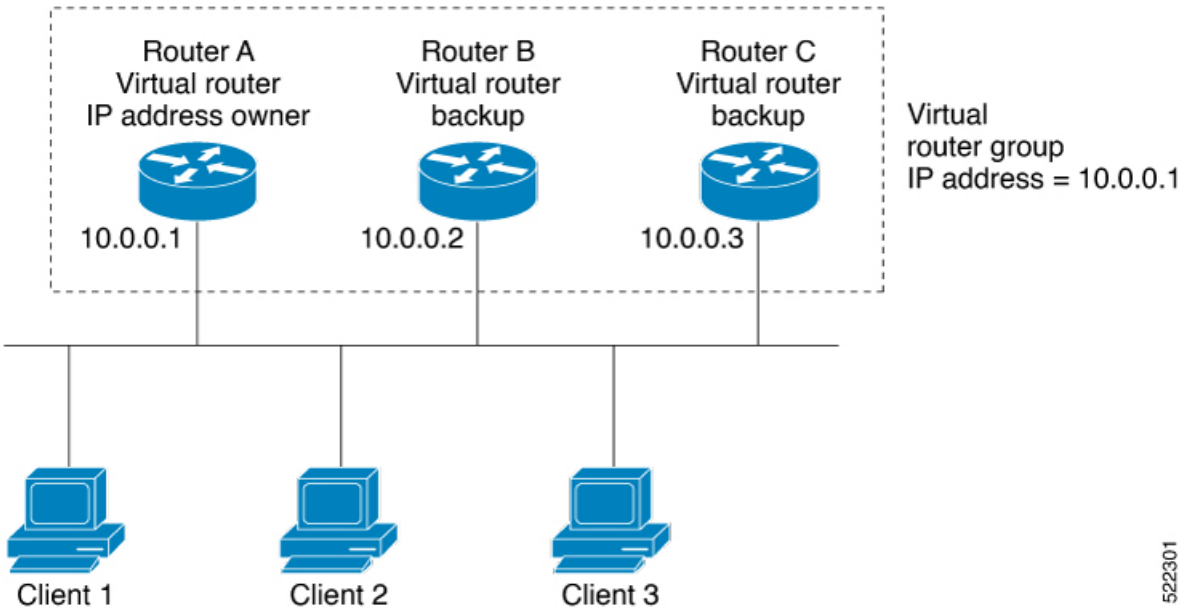
The Virtual Router Redundancy Protocol (VRRP) feature can solve the static configuration problem. VRRP is an IP routing redundancy protocol designed to allow for transparent failover at the first-hop IP router. VRRP enables a group of routers to form a single *virtual router*. The LAN clients can then be configured with the virtual router as their default gateway. The virtual router, representing a group of routers, is also known as a *VRRP group*.

When the virtual router group IP address is the same as the IP address of the physical interface of any router in the VRRP group, then such router becomes the *IP address owner* and the VRRP group operates in the *Owner* mode. When a VRRP group operates in Owner mode, the IP address owner is responsible for forwarding packets that are sent to the VRRP group.

For operating in Owner mode in case of IPv6 VRRP sessions, the link-local address that is configured for the VRRP session must be the same as the link-local address of the physical interface in a router. The link-local address can be autoconfigured by the router or can be an address that is configured by the administrator.

For example, the following topology shows a LAN in which VRRP is configured. In this example, Routers A, B, and C are *VRRP routers* (routers running VRRP) that compose a virtual router. The IP address of the virtual router is the same as that configured for the interface of Router A (10.0.0.1).

Figure 15: Basic VRRP topology



Because the virtual router uses the IP address of the physical interface of Router A, Router A assumes the role of the *IP address owner* and is responsible for forwarding packets that are sent to the VRRP group IP address. Clients 1 through 3 are configured with the default gateway IP address of 10.0.0.1.

Routers B and C function as *backup virtual routers*. If the router that is IP address owner fails, the router that is configured with the higher priority becomes the IP address owner and provides uninterrupted service for the LAN hosts. When Router A recovers, it becomes the IP address owner again.

Note

We recommend that you disable Spanning Tree Protocol (STP) on switch ports to which the virtual routers are connected. Enable RSTP or rapid-PVST on the switch interfaces if the switch supports these protocols.

Restrictions for VRRP configuration

This topic lists the generic restrictions to consider before you configure VRRP on supported interfaces.

Consider these restrictions before you configure VRRP:

- ICMP redirects are not supported.
- Protocol Independent Multicast (PIM) is not supported with VRRP.
- Before Release 24.2.1, the Cisco 8000 Series Routers do not support the configuration of Virtual IP (VIP) addresses for VRRP on subinterfaces. As an alternative, Bridge-Group Virtual Interfaces (BVIs) can be utilized for VRRP implementation on subinterfaces.

- Route Processor Fail Over (RPFO) is not supported for VRRP configuration on Cisco 8608 and Cisco 8404-SYS-D routers.

Multiple virtual router support

This topic describes support for multiple virtual routers on a router interface, including the factors that limit how many virtual routers a single interface can host.

You can configure up to 255 virtual routers on a router interface. The actual number of virtual routers that a router interface can support depends on the following factors:

- Router processing capability
- Router memory capability
- Router interface support of multiple MAC addresses

In a topology where multiple virtual routers are configured on a router interface, the interface can act as an IP address owner for one or more virtual routers and as a backup for one or more virtual routers.

VRRP router priority

This topic describes VRRP router priority, which determines the role of each VRRP router and the failover behavior when the IP address owner virtual router fails.

An important aspect of the VRRP redundancy scheme is VRRP router priority. Priority determines the role that each VRRP router plays and what happens if the IP address owner virtual router fails.

If a VRRP router owns the IP address of the virtual router and the IP address of the physical interface, this router functions as an IP address owner virtual router.

If no VRRP router owns the IP address, the priority of a VRRP router, combined with the preempt settings, determines if a VRRP router functions as an IP address owner router or a backup virtual router. By default, the highest priority VRRP router functions as IP address owner router, and all the others function as backups. Priority also determines the order of ascendancy to becoming an IP address owner virtual router if the IP address owner virtual router fails. You can configure the priority of each backup virtual router with a value of 1 through 254, using the **vrrp priority** command.

For example, if Router A, the IP address owner virtual router in a LAN topology, fails, an election process takes place to determine if backup virtual Routers B or C should take over. If Routers B and C are configured with the priorities of 101 and 100, respectively, Router B is elected to become IP address owner virtual router because it has the higher priority. If Routers B and C are both configured with the priority of 100, the backup virtual router with the higher IP address is elected to become the IP address owner virtual router.

By default, a preemptive scheme is enabled whereby a higher-priority backup virtual router that becomes available takes over from the current IP address owner virtual router. You can disable this preemptive scheme using the **vrrp preempt disable** command. If preemption is disabled, the backup virtual router that is elected to become IP address owner router upon the failure of the original higher priority IP address owner router, remains the IP address owner router even if the original IP address owner virtual router recovers and becomes available again.

VRRP advertisements

This topic describes VRRP advertisements, which the IP address owner virtual router sends to communicate its priority and state to other VRRP routers in the same group.

The IP address owner virtual router sends VRRP advertisements to other VRRP routers in the same group. The advertisements communicate the priority and state of the IP address owner virtual router. The VRRP advertisements are encapsulated in IP packets and sent to the IP Version 4 multicast address assigned to the VRRP group. The advertisements are sent every second by default; the interval is configurable.

Benefits of VRRP

This topic describes the benefits of VRRP, including redundancy, load sharing, support for multiple virtual routers and IP addresses, preemption, text authentication, and a standard advertisement protocol.

These are the benefits of VRRP:

- Redundancy—VRRP enables you to configure multiple routers as the default gateway router, which reduces the possibility of a single point of failure in a network.
- Load Sharing—You can configure VRRP in such a way that traffic to and from LAN clients can be shared by multiple routers, thereby sharing the traffic load more equitably among available routers.
- Multiple Virtual Routers—VRRP supports up to 100 virtual routers (VRRP groups) on a router interface, subject to the platform supporting multiple MAC addresses. You can configure up to 256 virtual routers on a router interface. Multiple virtual router support enables you to implement redundancy and load sharing in your LAN topology.
- Multiple IP Addresses—The virtual router can manage multiple IP addresses, including secondary IP addresses. Therefore, if you have multiple subnets configured on an Ethernet interface, you can configure VRRP on each subnet.
- Preemption—The redundancy scheme of VRRP enables you to preempt a backup virtual router that has taken over for a failing IP address owner virtual router with a higher-priority backup virtual router that has become available.
- Text Authentication—You can ensure that VRRP messages received from VRRP routers that comprise a virtual router are authenticated by configuring a simple text password.
- Advertisement Protocol—VRRP uses a dedicated Internet Assigned Numbers Authority (IANA) standard multicast address (224.0.0.18) for VRRP advertisements. This addressing scheme minimizes the number of routers that must service the multicasts and allows test equipment to accurately identify VRRP packets on a segment. The IANA assigns VRRP the IP protocol number 112.

Hot restartability for VRRP

This topic describes hot restartability for VRRP, which supports warm RP failover without causing forced failovers to peer VRRP routers.

In the event of failure of a VRRP process in one group, forced failovers in peer VRRP IP address owner router groups should be prevented. Hot restartability supports warm RP failover without incurring forced failovers to peer VRRP routers.

VRRP over physical and bundle interfaces

This topic describes VRRP support for IPv4 and IPv6 networks on physical interfaces, physical sub-interfaces, bundle interfaces, and bundle sub-interfaces, including supported scale and topology.

You can configure multiple routers from a VRRP router group and use one of these routers as the default gateway router to reduce the possibility of a single point of network failure. It also ensures that the traffic load received from and sent to LAN clients is equally shared between the available routers for optimal performance.

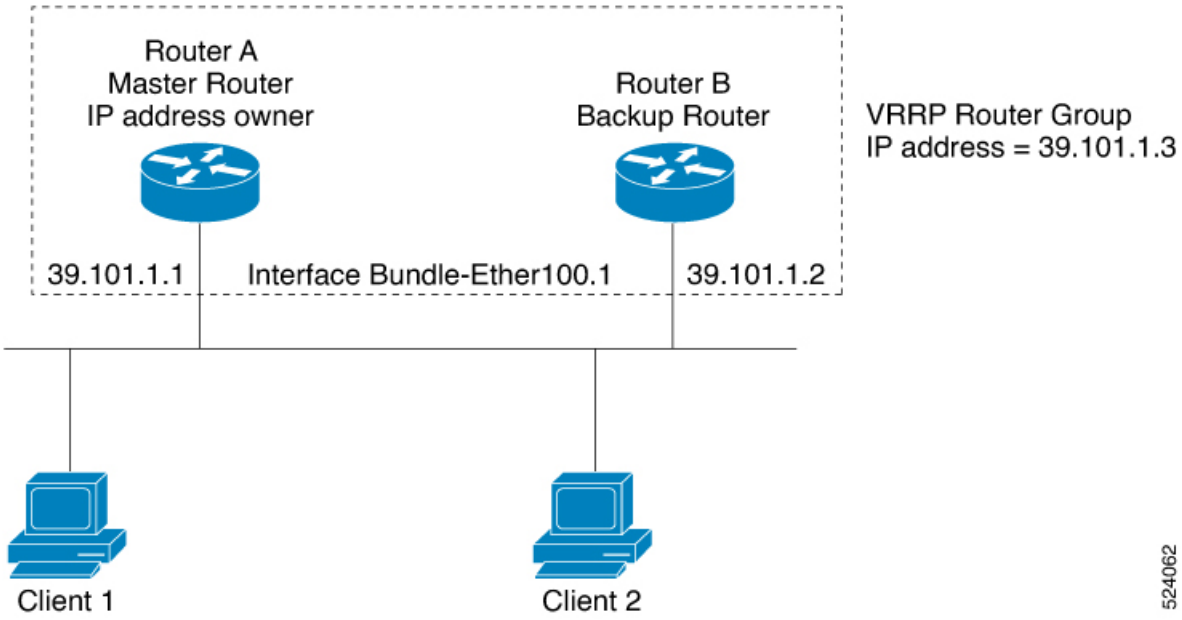
You can configure up to 510 IPv4 and IPv6 VRRP groups on the Cisco Silicon One Q200 and P100 ASIC-based systems.

Table 72: Feature History Table

Feature Name	Release Information	Feature Description
VRRP over Physical interfaces and Bundle 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
VRRP over Physical interfaces and 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
VRRP over Physical interfaces and Bundle interfaces	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
VRRP over Physical interfaces and Bundle interfaces	Release 24.2.11	This feature ensures high availability of routing paths by mitigating any failure in the primary interfaces within a group of physical or bundle interfaces or sub-interfaces in a network with a failover to a backup physical or bundle interface. The feature achieves this failover through a backup router in the VRRP router group configured on the physical or bundle interfaces or sub-interfaces. The virtual IP addresses from the failed primary router are handed over to this backup router.

Topology

This topology provides the VRRP implementation over a Bundle sub-interface on a network segment.



VRRP topology and underlying network elements:

- Interface Bundle-Ether100.1: This is the Bundle sub-interface which is configured on the Master router (IP Address: 39.101.1.1) and the Backup router (IP Address: 39.101.1.2).
- VRRP Router Group: The VRRP router group consists of two or more routers configured with VRRP. This group provides backup services to the backup router (Router B) in the event of the Master router (Router A) failure. The virtual IP address of the VRRP router group in this topology is 39.101.1.3.
- Router A: Router A is the Master router on which the Bundle sub-interface Bundle-Ether100.1 is configured. This router receives and routes packets destined for the MAC address of the group. This router is known as the IP address owner of the VRRP router group.
- Router B: When the designated Master router fails, VRRP detects it and selects another router with the highest priority in the VRRP router group as the Backup router. This Backup router with the Bundle sub-interface Bundle-Ether100.1 configuration takes over the job of Router A and assumes control of the MAC and IP Addresses of the VRRP router group.
- Client 1 and Client 2: These are source hosts from which traffic is forwarded through the Master router or Backup router to the destination hosts.

Guidelines for VRRP configuration

This topic describes the LPTS flow parameter adjustments required when you configure VRRP for both IPv4 and IPv6 groups with more than 510 sessions or with aggressive timer intervals below 1 second.

These guidelines apply when you configure VRRP:

When setting up VRRP for both IPv4 and IPv6 groups with more than 510 sessions, or when using aggressive timer intervals below 1 second, it is crucial to adjust the LPTS flow parameters accordingly. Specifically, increase the LPTS policer rate to a minimum of 3000 pps. This ensures optimal performance and avoids any LPTS drops for VRRP traffic to maintain network stability and efficiency. For detailed instructions on configuring these settings, refer to the *Usage Guidelines* section within the **lpts pifib hardware police** command documentation.

Configure VRRP for IPv4 networks on physical and bundle interfaces

This topic describes how to configure and verify VRRP for IPv4 networks on physical interfaces, physical sub-interfaces, bundle interfaces, and bundle sub-interfaces.

Perform the following configuration on both the master router and the backup router. On each router, first configure the main bundle interface (*Bundle-Ether100*) and add the member links to *Bundle-Ether100*.

> **Note**
>
> Apply customizations to control the behavior of the VRRP group when you commit the VRRP configuration on the router. Without these customizations, the router seizes control of the VRRP group and immediately assumes the role of the IP address owner virtual router.

Procedure

1. On the master router, enter the interface configuration mode and configure an IPv4 address for the interface.

   Example

   ```
   Router(config)# interface Bundle-Ether100.1
   Router(config-if)# ipv4 address 39.101.1.1/24
   Router(config-if)# encapsulation dot1q 1
   Router(config-if)# no shut
   Router(config-if)# commit
   Router(config-if)# exit
   ```

2. Enter the VRRP configuration mode, add the configured interface, and configure VRRP for IPv4.

   Example

   ```
   Router(config)# router vrrp
   Router(config-vrrp)# interface Bundle-Ether100.1
   Router(config-vrrp-if)# address-family ipv4 vrrp 1
   Router(config-vrrp-virtual-router)# address 39.101.1.3
   ```

3. Set a priority for the virtual router.

   Example

   ```
   Router(config-vrrp-virtual-Router)# priority 100
   ```

4. Configure a preempt delay value that controls the selection of the IP address owner virtual router, and commit the configuration.

   Example

   ```
   Router(config-vrrp-virtual-Router)# preempt delay 30
   Router(config-vrrp-virtual-Router)# commit
   ```

5. On the backup router, repeat the preceding steps by using an IPv4 address that is unique to the backup router but sharing the same VRRP virtual address.

Example

```
Router(config)# interface Bundle-Ether100.1
Router(config-if)# ipv4 address 39.101.1.2/24
Router(config-if)# encapsulation dot1q 1
Router(config-if)# no shut
Router(config-if)# commit
Router(config-if)# exit
Router(config)# router vrrp
Router(config-vrrp)# interface Bundle-Ether100.1
Router(config-vrrp-if)# address-family ipv4 vrrp 1
Router(config-vrrp-virtual-router)# address 39.101.1.3
Router(config-vrrp-virtual-Router)# priority 100
Router(config-vrrp-virtual-Router)# preempt delay 30
Router(config-vrrp-virtual-Router)# commit
```

6. Verify the interface configuration on the master router by using the **show run interface** command.

Example

```
Router(config-vrrp-virtual-router)# do show run interface Bundle-Ether100.1
interface Bundle-Ether100.1
 ipv4 address 39.101.1.1/24
 encapsulation dot1q 1
!
```

7. Verify VRRP operation by using the **show vrrp detail** command.

Example

```
Router# show vrrp detail
Bundle-Ether100.1 - IPv4 vrID 1
  State is Master
    16 state changes, last state change 00:05:26
    State change history:
    Feb 23 05:09:11.811 UTC  Backup   -> Init     Interface Down update
    Feb 23 05:13:25.018 UTC  Init     -> Backup   Delay timer expired
    Feb 23 05:13:28.990 UTC  Backup   -> Master   Master down timer expired
    Feb 23 05:13:59.911 UTC  Master   -> Backup   Higher priority advert
received
    Feb 23 05:28:20.344 UTC  Backup   -> Master   Master down timer expired
  Last resign sent:     Never
  Last resign received: Feb 23 05:02:47.544 UTC
  Virtual IP address is 39.101.1.3
  Virtual MAC address is 0000.5E00.0101, state is active
  Master router is local
  Version is 2
  Advertise time 1 secs
    Master Down Timer 3.960 (3 x 1 + (246 x 1/256))
  Minimum delay 1 sec, reload delay 5 sec
  Current priority 10
    Configured priority 10, may preempt
      minimum delay 30 secs
```

You have successfully configured and verified VRRP for IPv4 networks on the master and backup routers.

Configure VRRP for IPv6 networks on physical and bundle interfaces

This topic describes how to configure and verify VRRP for IPv6 networks on physical interfaces, physical sub-interfaces, bundle interfaces, and bundle sub-interfaces.

Perform the following configuration on both the master router and the backup router. On each router, first configure the main bundle interface (*Bundle-Ether100*) and add the member links to *Bundle-Ether100*.

> **Note**
>
> Apply customizations to control the behavior of the VRRP group when you commit the VRRP configuration on the router. Without these customizations, the router seizes control of the VRRP group and immediately assumes the role of the IP address owner virtual router.

Procedure

1. On the master router, enter the interface configuration mode and configure an IPv6 address.

   Example

   ```
   Router# interface Bundle-Ether100.1
   Router(config-if)# ipv6 address 39:100:0:1::1:1/64
   Router(config-if)# encapsulation dot1q 1
   Router(config-if)# no shut
   ```

2. Exit the interface configuration mode, enter the VRRP configuration mode, and add the configured interface for VRRP.

   Example

   ```
   Router(config-if)# exit
   Router(config)# Router vrrp
   Router(config-vrrp)# interface Bundle-Ether100.1
   ```

3. Enable the IPv6 global and link local address family on the interface.

   Example

   ```
   Router(config-vrrp-if)# address-family ipv6 vrrp 1
   Router(config-vrrp-virtual-Router)# address global 39:100:0:1::1:3
   Router(config-vrrp-virtual-Router)# address linklocal autoconfig
   ```

4. Set a priority for the virtual router.

   Example

   ```
   Router(config-vrrp-virtual-Router)# priority 100
   ```

5. Configure a preempt delay value that controls the selection of the IP address owner virtual router, and commit the configuration.

Example

```
Router(config-vrrp-virtual-Router)# preempt delay 30
Router(config-vrrp-virtual-Router)# commit
```

6. On the backup router, repeat the preceding steps by using an IPv6 address that is unique to the backup router but sharing the same VRRP virtual address.

Example

```
Router# interface Bundle-Ether100.1
Router(config-if)# ipv6 address 39:100:0:1::1:2/64
Router(config-if)# encapsulation dot1q 1
Router(config-if)# no shut
Router(config-if)# exit
Router(config)# Router vrrp
Router(config-vrrp)# interface Bundle-Ether100.1
Router(config-vrrp-if)# address-family ipv6 vrrp 1
Router(config-vrrp-virtual-Router)# address global 39:100:0:1::1:3
Router(config-vrrp-virtual-Router)# address linklocal autoconfig
Router(config-vrrp-virtual-Router)# priority 100
Router(config-vrrp-virtual-Router)# preempt delay 30
Router(config-vrrp-virtual-Router)# commit
```

7. Verify the interface configuration on the master router by using the **show run interface** command.

Example

```
Router(config-vrrp-virtual-router)# do show run interface Bundle-Ether100.1
interface Bundle-Ether100.1
 ipv6 address 39:100:0:1::1:1/64
 encapsulation dot1q 1
!
```

8. Verify the VRRP configuration by using the **show running-config router vrrp** command.

Example

```
Router(config-vrrp-virtual-router)# do show running-config router vrrp
router vrrp
 interface Bundle-Ether100.1
  address-family ipv6
   vrrp 1
    priority 100
    preempt delay 30
    address global 39:100:0:1::1:3
    address linklocal autoconfig
  !
 !
!
```

9. Verify VRRP operation by using the **show vrrp detail** command.

Example

```
Router# show vrrp detail
Bundle-Ether100.1 - IPv6 vrID 1
```

```
  State is Master
    18 state changes, last state change 00:05:26
    State change history:
    Feb 23 05:09:11.811 UTC  Backup   -> Init     Interface Down update
    Feb 23 05:13:25.018 UTC  Init     -> Backup   Delay timer expired
    Feb 23 05:13:28.990 UTC  Backup   -> Master   Master down timer expired
    Feb 23 05:13:59.911 UTC  Master   -> Backup   Higher priority advert
received
    Feb 23 05:28:20.344 UTC  Backup   -> Master   Master down timer expired
  Last resign sent:     Never
  Last resign received: Feb 22 16:57:09.690 UTC
   Virtual IP address is fe80::200:5eff:fe00:201
     Secondary Virtual IP address is 39:100:0:1::1:3
   Virtual MAC address is 0000.5E00.0201, state is active
   Master router is local
   Version is 3
  Advertise time 1 secs
    Master Down Timer 3.960 (3 x 1 + (246 x 1/256))
  Minimum delay 1 sec, reload delay 5 sec
  Current priority 10
    Configured priority 10, may preempt
      minimum delay 30 secs
```

You have successfully configured and verified VRRP for IPv6 networks on the master and backup routers.

Unicast VRRP

This topic describes how VRRP supports Layer 3 unicast transport mode to enable high availability in cloud network scenarios where Layer 2 multicast or broadcast transports are not supported.

The Virtual Router Redundancy Protocol (VRRP) configuration for Layer 3 unicast transport is a network mechanism that

- supports Layer 3 unicast transport
- enables high availability for cloud networks, and
- utilizes event-driven telemetry for state notifications.

Network architects implement this feature to extend gateway redundancy into environments that restrict Layer 2 multicast and broadcast traffic. Because cloud-native functions require a stable default route, this configuration provides a virtual IP address (VIP) to manage traffic without relying on pre-designated active members.

Table 73: Feature History Table

Feature Name	Release Name	Description
Unicast VRRP	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

Feature Name	Release Name	Description
Unicast VRRP	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
Unicast VRRP	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Name	Description
Unicast VRRP	Release 7.11.1	We have now enabled Layer 3 unicast transport mode in VRRP, allowing it to enhance its capacity to send data to other networks, including cloud networks. Pairwise router redundancy enables high availability in cloud network scenarios. However, a virtual IP (VIP) address is required by the default route of the cloud native function because there is no pre-designated active member in paired routers. HSRP can provide a VIP, but cloud networks do not support Layer 2 multicast or broadcast transports. You can configure VRRP to support Layer 3 unicast transport to overcome the limitation of Layer 2 multicast and broadcast transports. The feature introduces these changes: New Command: CLI: • **unicast-peer** Modified Commands: • **show vrrp** command is modified to support new fields: **Mcast packet in Ucast mode**, **IPv4 Unicast Peer**, and **IPv4 Unicast Peer**. YANG Data Model: New Xpaths for: • `Cisco-IOS-XR-ipv4-vrrp-cfg.yang` • `Cisco-IOS-XR-ipv4-vrrp-oper.yang` (see GitHub, YANG Data Models Navigator)

Benefits of unicast VRRP

The VRRP configuration for Layer 3 unicast transport provides several benefits for modern network environments:

- It allows you to deploy redundant gateway solutions in cloud environments that lack Layer 2 multicast support.
- It uses a virtual IP address to ensure seamless failover between paired routers.
- It sends state transition notifications through event-driven telemetry to improve monitoring capabilities.

VRRP over BVI

This topic describes how VRRP runs over Bridge-Group Virtual Interfaces (BVIs) to provide default gateway redundancy for a LAN, including supported scale and topology.

The VRRP session over a Bridge-Group Virtual Interface (BVI) is a redundancy configuration that

- provides default gateway protection for a LAN
- enables a group of routers to behave as a single virtual gateway, and
- utilizes a routed interface to manage bridge group traffic.

Layer 2 functionality is applicable to the interfaces which are part of a bridge group and BVI is the routed interface for that bridge group.

The network architecture assigns the primary role to the router with the highest priority, while all other routers function as backups. The Bridge-Group Virtual Interface (BVI) provides Layer 3 functionality to a bridge group, which allows the system to treat a bridged network as a routable entity. Configure VRRP sessions on the BVI to ensure that the redundancy protocol functions correctly within the bridge group environment.

These are the requirements for configuring these sessions:

- The system designates the router with the highest priority as the primary gateway.
- The BVI acts as the routed interface that enables Layer 3 functionality for the bridge group.
- Administrators restrict the configuration of VRRP sessions to the BVI to maintain compatibility.

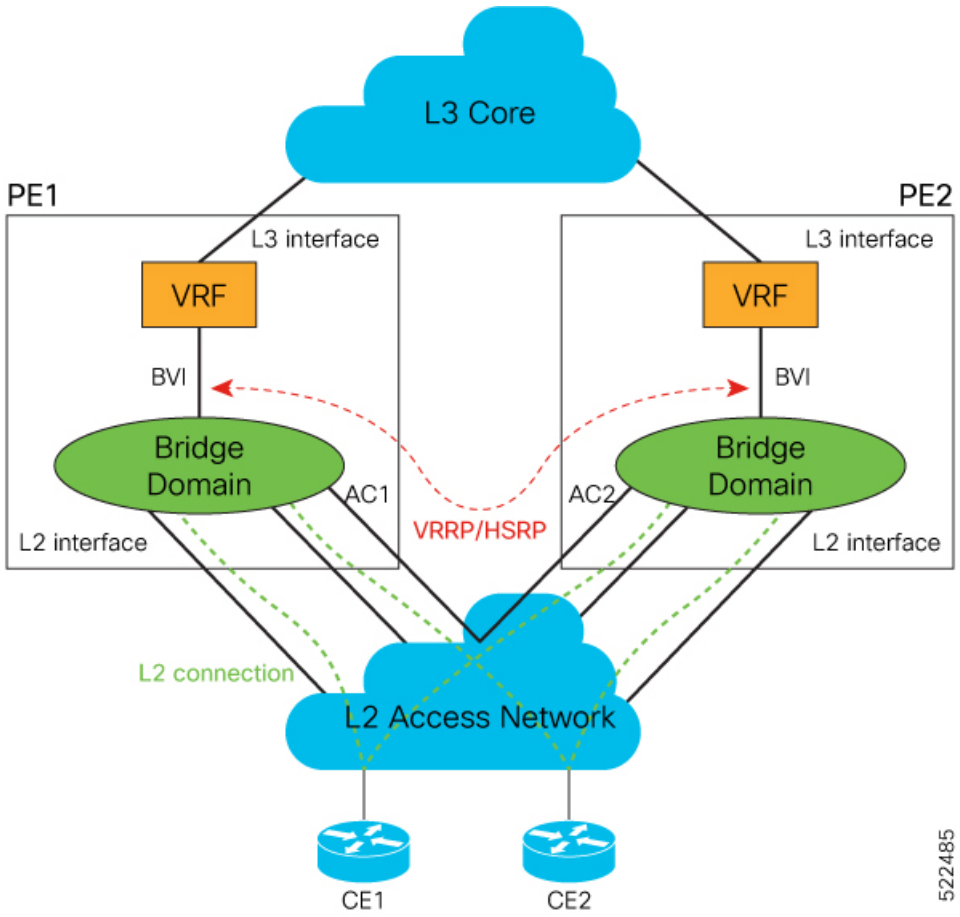
Table 74: Feature History Table

Feature Name	Release Information	Feature Description
VRRP over BVI	Release 26.2.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is supported on: • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
VRRP over BVI	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.
VRRP over BVI	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) * This feature is supported on: • 8712-MOD-M • 8711-48Z-M

Feature Name	Release Information	Feature Description
VRRP over BVI	Release 25.2.1	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH
VRRP over BVI	Release 25.1.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
VRRP over BVI	Release 7.5.2	Virtual Router Redundancy Protocol (VRRP) runs on top of interfaces of multiple routers in the same home network that has both Cisco and other vendor routers. It allows a group of routers to behave as a single virtual default gateway router, thereby providing default gateway redundancy and minimizing traffic loss. VRRP now supports Bridge-Group Virtual Interface (BVI), which means that VRRP sessions can run between BVI interfaces of multiple routers.

Topology

This topology showcases how VRRP functions over BVI.



In this topology, PE1 and PE2 are paired in a redundant group. This group provides Layer 3 gateway service to CE1 and CE2. VRRP is configured over BVI interfaces on PE1 and PE2. VRRP ensures one BVI is the active gateway. The other is the standby gateway.

You can configure one of the BVIs to be active and the other BVI as standby by setting the VRRP priority value. The active BVI is programmed with the virtual MAC address chosen by VRRP. Hosts, CE1 and CE2 send the traffic to the virtual destination MAC address and the active BVI forwards the traffic.

During failover, the standby BVI becomes active and is programmed with the virtual MAC address. The traffic from the hosts is forwarded through this active BVI.

Supported scale and systems

VRRP over Bridge Virtual Interfaces (BVIs) is supported:

- You can configure upto 512 VRRP groups (IPv4 and IPv6 combined) over BVIs on:
  - Cisco Silicon One Q100 ASIC-based systems
  - Cisco Silicon One Q200 ASIC-based systems
  - Cisco Silicon One P100 ASIC-based systems
- Where the underlay IRB bridge domains consist of bridge members on L2 main or subinterfaces. Only physical and bundle interfaces are supported for L2 bridging in IRB.
- When both IPv4 and IPv6 are configured on a BVI interface, IPv4 and IPv6 each requires a session. A total of two sessions are consumed on a BVI interface.
- For IPv4 and IPv6 configurations, in both the default and VRF tables.

- On both the fixed and distributed systems.
- VRRP on BVI is supported on the 88-LC1-12TH24FH-E line card from Cisco IOS XR Release 25.1.1.

Restrictions for VRRP over BVI

This topic lists the restrictions to consider before you configure VRRP over Bridge-Group Virtual Interfaces (BVIs).

Consider these restrictions before you configure VRRP over BVIs:

- The minimum supported VRRP Hello timer is 100 ms. At the minimum timer, a total of 50 sessions are supported. Above 100 ms timers, the sessions scale goes up proportionately. A maximum of 255 VRRP groups and 510 sessions are supported.
- VRRP on BVI is not supported on the 88-LC1-12TH24FH-E line card in Cisco IOS XR Release 24.3.1.

Configure VRRP over BVI

This topic describes how to configure and verify VRRP sessions over Bridge-Group Virtual Interfaces (BVIs) on PE1 and PE2 routers for IPv4 and IPv6 addresses.

To configure VRRP sessions over BVI, complete the following configurations on PE1 and PE2 in the specified order:

1. Configure a set of interfaces as Layer 2 interfaces and a set of VLAN sub-interfaces.
2. Configure a bridge group.
3. Configure a BVI.
4. Configure VRRP over BVI.

Procedure

1. Enter the global configuration mode and configure a set of interfaces as Layer 2 interfaces and a set of VLAN sub-interfaces.

   Example

```
Router# configure
Router(config)# interface HundredGigE0/0/1/0.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# interface HundredGigE0/0/1/1.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
```

2. Enter the Layer 2 VPN configuration mode and configure a bridge group.

   Example

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 5
Router(config-l2vpn-bg)# bridge-domain 5
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/1/0.1
```

```
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/1/1.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# routed interface BVI 10
Router(config-l2vpn-bg-bd-bvi)# commit
Router(config-l2vpn-bg-bd-bvi)# exit
```

3. Configure a BVI in the global configuration mode.

   Example

```
Router(config)# interface BVI 10
Router(config-if)# ipv4 address 209.165.200.225 255.255.255.0
Router(config-if)# ipv6 address 2001:DB8:A:B::1/64
Router(config-if)# commit
```

4. Configure VRRP over BVI in the global configuration mode for the IPv4 address.

   Example

```
Router(config)# router VRRP
Router(config-vrrp)# interface BVI 10
Router(config-vrrp-if)# address-family ipv4
Router(config-vrrp-address-family)# VRRP 10
Router(config-vrrp-virtual-router)# priority 101
Router(config-vrrp-virtual-router)# address 209.165.200.226
Router(config-vrrp-virtual-router)# commit
```

5. Configure VRRP over BVI in the global configuration mode for the IPv6 address.

   Example

```
Router(config)# router VRRP
Router(config-vrrp)# interface BVI 10
Router(config-vrrp-if)# address-family ipv6
Router(config-vrrp-address-family)# VRRP 11
Router(config-vrrp-virtual-router)# address global 2001:DB8:A:B::2
Router(config-vrrp-virtual-router)# address linklocal autoconfig
Router(config-vrrp-virtual-router)# commit
```

6. Verify the bridge domain details by using the **show l2vpn bridge-domain detail** command.

   Example

```
Router# show l2vpn bridge-domain detail

Legend: pp = Partially Programmed.
Bridge group: 5, bridge-domain: 5, id: 1, state: up, ShgId: 0, MSTi: 0
Coupled state: disabled
VINE state: BVI Resolved
MAC learning: enabled
MAC withdraw: enabled
MAC withdraw for Access PW: enabled
MAC withdraw sent on: bridge port up
MAC withdraw relaying (access to access): disabled
Flooding:
Broadcast & Multicast: enabled
```

```
Unknown unicast: enabled
MAC aging time: 300 s, Type: inactivity
MAC limit: 32768, Action: none, Notification: syslog
MAC limit reached: no, threshold: 75%
MAC port down flush: enabled
MAC Secure: disabled, Logging: disabled
Split Horizon Group: none
Dynamic ARP Inspection: disabled, Logging: disabled
IP Source Guard: disabled, Logging: disabled
DHCPv4 Snooping: disabled
DHCPv4 Snooping profile: none
IGMP Snooping: disabled
IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control: disabled
Bridge MTU: 1500
MIB cvplsConfigIndex: 2
Filter MAC addresses:
P2MP PW: disabled
Multicast Source: Not Set
Create time: 26/05/2020 17:08:54 (00:11:30 ago)
No status change since creation
ACs: 3 (3 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
List of ACs:
AC: BVI10, state is up
Type Routed-Interface
MTU 1514; XC ID 0x80000001; interworking none
BVI MAC address:
c472.95a6.8b90
Virtual MAC addresses:
0000.5e00.010a
0000.5e00.020b
Split Horizon Group: Access
AC: HundredGigE0/0/1/0.1, state is up
Type VLAN; Num Ranges: 1
Rewrite Tags: []
VLAN ranges: [1, 1]
MTU 1500; XC ID 0x1; interworking none
MAC learning: enabled
```

7. Verify VRRP details for IPv4 by using the **show vrrp ipv4 detail** command.

Example

```
Router# show vrrp ipv4 detail

BVI10 - IPv4 vrID 10
State is Master
2 state changes, last state change 00:11:57
State change history:
May 26 17:08:59.470 UTC Init -> Backup Delay timer expired
May 26 17:09:03.075 UTC Backup -> Master Master down timer expired
Last resign sent: Never
Last resign received: Never
Virtual IP address is 209.165.200.226
Virtual MAC address is 0000.5E00.010a, state is active
Master router is local
Version is 2
Advertise time 1 secs
Master Down Timer 3.605 (3 x 1 + (155 x 1/256))
Minimum delay 1 sec, reload delay 5 sec
```

```
Current priority 101
Configured priority 101, may preempt
minimum delay 0 secs
```

8. Verify VRRP details for IPv6 by using the **show vrrp ipv6 detail** command.

Example

```
Router# show vrrp ipv6 detail

BVI10 - IPv6 vrID 11
State is Master
2 state changes, last state change 00:04:29
State change history:
May 26 17:16:43.476 UTC Init -> Backup Virtual IP configured
May 26 17:16:47.085 UTC Backup -> Master Master down timer expired
Last resign sent: Never
Last resign received: Never
Virtual IP address is fe80::200:5eff:fe00:20b
Secondary Virtual IP address is 2001:db8:a:b::2
Virtual MAC address is 0000.5E00.020b, state is active
Master router is local
Version is 3
Advertise time 1 secs
Master Down Timer 3.609 (3 x 1 + (156 x 1/256))
Minimum delay 1 sec, reload delay 5 sec
Current priority 100
Configured priority 100, may preempt
minimum delay 0 secs

Router# show vrrp interface BVI10 detail
BVI10 - IPv4 vrID 10
State is Master
2 state changes, last state change 00:12:35
State change history:
May 26 17:08:59.470 UTC Init -> Backup Delay timer expired
May 26 17:09:03.075 UTC Backup -> Master Master down timer expired
Last resign sent: Never
Last resign received: Never
Virtual IP address is 209.165.200.226
Virtual MAC address is 0000.5E00.010a, state is active
Master router is local
Version is 2
Advertise time 1 secs
Master Down Timer 3.605 (3 x 1 + (155 x 1/256))
Minimum delay 1 sec, reload delay 5 sec
Current priority 101
Configured priority 101, may preempt
minimum delay 0 secs
BVI10 - IPv6 vrID 11
State is Master
2 state changes, last state change 00:04:51
State change history:
May 26 17:16:43.476 UTC Init -> Backup Virtual IP configured
May 26 17:16:47.085 UTC Backup -> Master Master down timer expired
Last resign sent: Never
Last resign received: Never
Virtual IP address is fe80::200:5eff:fe00:20b
Secondary Virtual IP address is 2001:db8:a:b::2
Virtual MAC address is 0000.5E00.020b, state is active
Master router is local
```

```
Version is 3
Advertise time 1 secs
Master Down Timer 3.609 (3 x 1 + (156 x 1/256))
Minimum delay 1 sec, reload delay 5 sec
Current priority 100
Configured priority 100, may preempt
minimum delay 0 secs
```

You have successfully configured and verified VRRP over BVI for IPv4 and IPv6 addresses.

VRRP statistics

This topic describes the VRRP statistics that you can view or clear for one or all VRRP groups or Virtual Router IDs (VRIDs) to monitor VRRP health and debug VRRP issues.

You can view or clear statistics of one or all Virtual Router Redundancy Protocol (VRRP) groups or Virtual Router IDs (VRIDs). This information helps you monitor VRRP health in the routers. It is also helpful in debugging VRRP issues like packet exchange failures when all virtual routers in the VRRP topology function as backup virtual routers and there is no IP address owner.

Table 75: Feature History Table

Feature Name	Release Information	Feature Description
View VRRP statistics in Router	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
View VRRP statistics in Router	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 Information	Feature Description
View VRRP statistics in Router	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 supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
View VRRP statistics in Router	Release 7.9.1	With this feature, you can view or clear statistics of one or all Virtual Router Redundancy Protocol (VRRP) groups or Virtual Router IDs (VRIDs). This information helps you monitor VRRP health in the routers. It is also helpful in debugging VRRP issues like packet exchange failures when all virtual routers in the VRRP topology function as backup virtual routers and there is no IP address owner. This feature introduces the following commands: • CLI: • **show vrrp statistics** • **clear vrrp statistics** • YANG Data Model: `Cisco-IOS-XR-ipv4-vrrp-oper.yang` Cisco native model (see GitHub, YANG Data Models Navigator)

View VRRP statistics

This topic describes how to view statistics of one or all VRRP virtual routers by using the show vrrp statistics command.

Use this procedure to view statistics of one or all Virtual Router Redundancy Protocol (VRRP) virtual routers.

Procedure

Display VRRP statistics by using the **show vrrp statistics** command.

Example

```
Router# show vrrp statistics
Invalid packets:
  Invalid checksum:                     0
  Unknown/unsupported versions:         3
  Invalid vrID:                         1
  Too short:                            7
Protocol:
  Transitions to Master                 4
Packets:
  Total received:                       54
  Adverts sent:                         0
  Bad TTL:                              0
  Short Packets:                        6
  Failed authentication:                0
  Unknown authentication:               2
  Conflicting authentication:           0
  Unknown Type field:                   1
  Conflicting Advertise time:           0
  Conflicting Addresses:                0
  Received with zero priority:          9
  Sent with zero priority:              0
```

You have successfully viewed the VRRP statistics.

Clear VRRP statistics

This topic describes how to clear the VRRP statistics by using the clear vrrp statistics command and how to verify that the statistics are reset.

Use this procedure to clear the Virtual Router Redundancy Protocol (VRRP) statistics and verify that the counters have been reset to zero.

Procedure

1. Clear the VRRP statistics by using the **clear vrrp statistics** command.

   Example

   ```
   Router# clear vrrp statistics
   ```

2. Verify that the statistics are reset by using the **show vrrp statistics** command.

   Example

   ```
   Router# show vrrp statistics
   Invalid packets:
     Invalid checksum:                     0
     Unknown/unsupported versions:         0
     Invalid vrID:                         0
     Too short:                            0
   Protocol:
     Transitions to Master                 0
   Packets:
     Total received:                       0
     Adverts sent:                         0
   ```

```
  Bad TTL:                            0
  Short Packets:                      0
  Failed authentication:              0
  Unknown authentication:             0
  Conflicting authentication:         0
  Unknown Type field:                 0
  Conflicting Advertise time:         0
  Conflicting Addresses:              0
  Received with zero priority:        0
  Sent with zero priority:            0
```

You have successfully cleared the VRRP statistics.

14 Implement HSRP

Topics:

Describes how to implement the Hot Standby Router Protocol (HSRP) to provide transparent failover at the first-hop IP router by allowing a group of routers to act as a single virtual router.

HSRP

This topic describes the Hot Standby Router Protocol (HSRP), an IP routing redundancy protocol that provides transparent failover at the first-hop IP router by allowing a group of routers to select an active and a standby router.

The Hot Standby Router Protocol (HSRP) is an IP routing redundancy protocol that

- provides transparent failover at the first-hop router
- increases network availability, and
- manages traffic routing without relying on a single device.

HSRP operates by selecting an active router to forward traffic and a standby router to assume responsibilities during a failure. This architecture ensures continuous network connectivity by maintaining a ready backup within the router group.

HSRP employs specific roles to maintain network continuity:

- The active router serves as the primary device for forwarding IP traffic.
- The standby router monitors the active device and takes over if a failure occurs or if specific conditions are met.

From Release 24.2.1, you can configure HSRP VIP on subinterfaces.

Table 76: Feature History Table

Feature Name	Release Information	Feature Description
Hot Standby Router Protocol	Release 26.3.1	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.
Hot Standby Router Protocol	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
Hot Standby Router Protocol	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
Enabling Hot Standby Router Protocol over BVI	Release 25.1.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) You can now enable Hot Standby Router Protocol (HSRP) over BVI on Cisco Silicon One P100 ASIC-based systems. This feature, enabled by default, ensures continuous network availability by automatically switching to a standby router if the active router fails. This minimizes downtime and maintains network reliability. *The feature is supported on: • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

To implement HSRP on Cisco IOS XR software, you need to understand these concepts:

- HSRP is useful for hosts that do not support a router discovery protocol (such as Internet Control Message Protocol [ICMP] Router Discovery Protocol [IRDP]) and cannot switch to a new router when their selected router reloads or loses power. Because existing TCP sessions can survive the failover, this protocol also provides a more transparent recovery for hosts that dynamically choose a next hop for routing IP traffic.
- When HSRP is configured on a network segment, it provides a virtual MAC address and an IP address that is shared among a group of routers running HSRP. The address of this HSRP group is referred to as the *virtual IP address*. One of these devices is selected by the protocol to be the *active router*. The active router receives and routes packets destined for the MAC address of the group. For *n* routers running HSRP, *n* + 1 IP and MAC addresses are assigned.
- HSRP detects when the designated active router fails, at which point a selected standby router assumes control of the MAC and IP addresses of the HSRP group. A new *standby router* is also selected at that time.
- Devices that are running HSRP send and receive multicast User Datagram Protocol (UDP) based hello packets to detect router failure and to designate active and standby routers.

Generic restrictions for HSRP configuration

This topic lists the generic restrictions to consider before you implement HSRP on supported interfaces.

Consider these restrictions before you implement HSRP:

- HSRP is not supported with bidirectional forwarding detection (BFD).
- Before Release 24.2.1, the Cisco 8000 Series Routers do not support the configuration of Virtual IP (VIP) addresses for HSRP on subinterfaces. As an alternative, Bridge-Group Virtual Interfaces (BVIs) can be utilized for HSRP implementation on subinterfaces.
- Route Processor Fail Over (RPFO) is not supported for HSRP configuration on Cisco 8608 and Cisco 8404-SYS-D routers.

HSRP overview

This topic describes HSRP behavior for hosts that do not support a router discovery protocol, including the virtual IP address, virtual MAC address, active and standby router roles, and hello packet exchange.

To implement HSRP on Cisco IOS XR software, you need to understand the following concepts.

HSRP is useful for hosts that do not support a router discovery protocol (such as Internet Control Message Protocol [ICMP] Router Discovery Protocol [IRDP]) and cannot switch to a new router when their selected router reloads or loses power. Because existing TCP sessions can survive the failover, this protocol also provides a more transparent recovery for hosts that dynamically choose a next hop for routing IP traffic.

When HSRP is configured on a network segment, it provides a virtual MAC address and an IP address that is shared among a group of routers running HSRP. The address of this HSRP group is referred to as the *virtual IP address*. One of these devices is selected by the protocol to be the *active router*. The active router receives and routes packets destined for the MAC address of the group. For *n* routers running HSRP, *n* + 1 IP and MAC addresses are assigned.

HSRP detects when the designated active router fails, at which point a selected standby router assumes control of the MAC and IP addresses of the HSRP group. A new *standby router* is also selected at that time.

Devices that are running HSRP send and receive multicast User Datagram Protocol (UDP) based hello packets to detect router failure and to designate active and standby routers.

HSRP groups

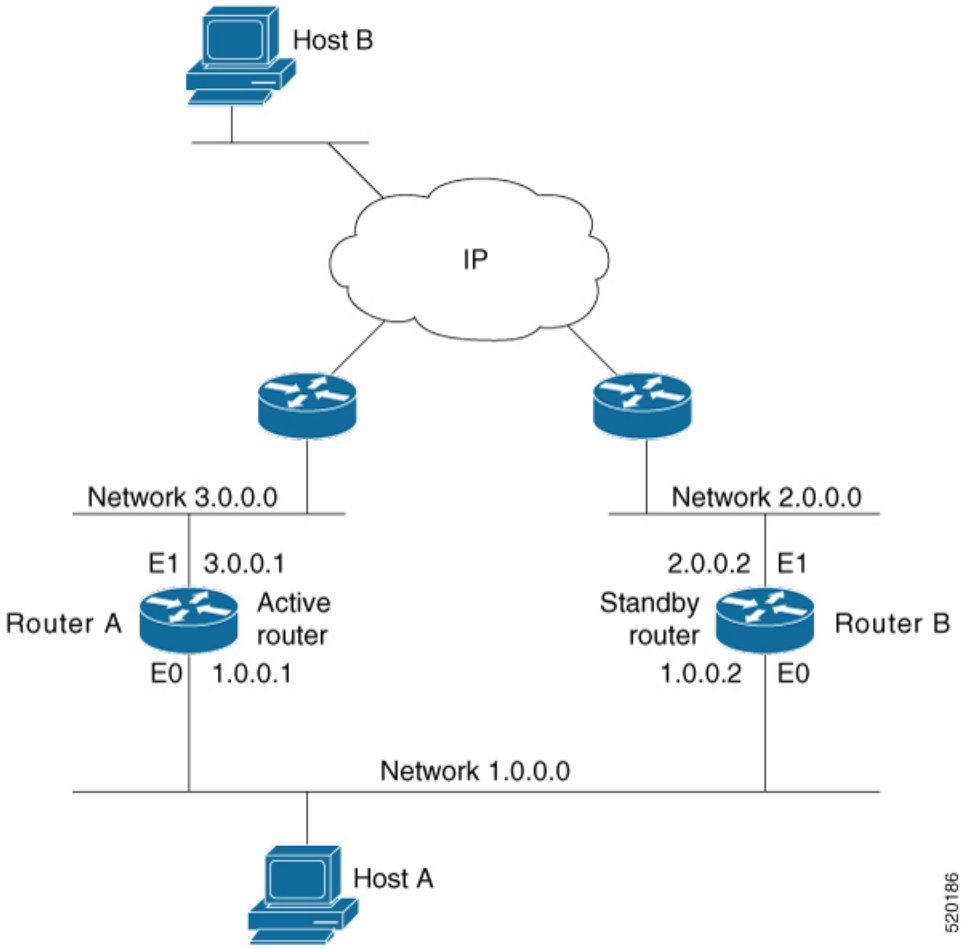
This topic describes HSRP groups, how HSRP-configured routers use priority to select the active router, and how a single router interface can belong to multiple HSRP groups.

An HSRP group consists of two or more routers running HSRP that are configured to provide hot standby services for one another. HSRP uses a priority scheme to determine which HSRP-configured router is to be the default active router. To configure a router as the active router, you assign it a priority that is higher than the priority of all the other HSRP-configured routers. The default priority is 100, so if you configure just one router to have a higher priority, that router will be the default active router.

HSRP works by the exchange of multicast messages that advertise priority among the HSRP group. When the active router fails to send a hello message within a configurable period of time, the standby router with the highest priority becomes the active router. The transition of packet-forwarding functions between routers is completely transparent to all hosts on the network.

The following figure shows routers configured as members of a single HSRP group.

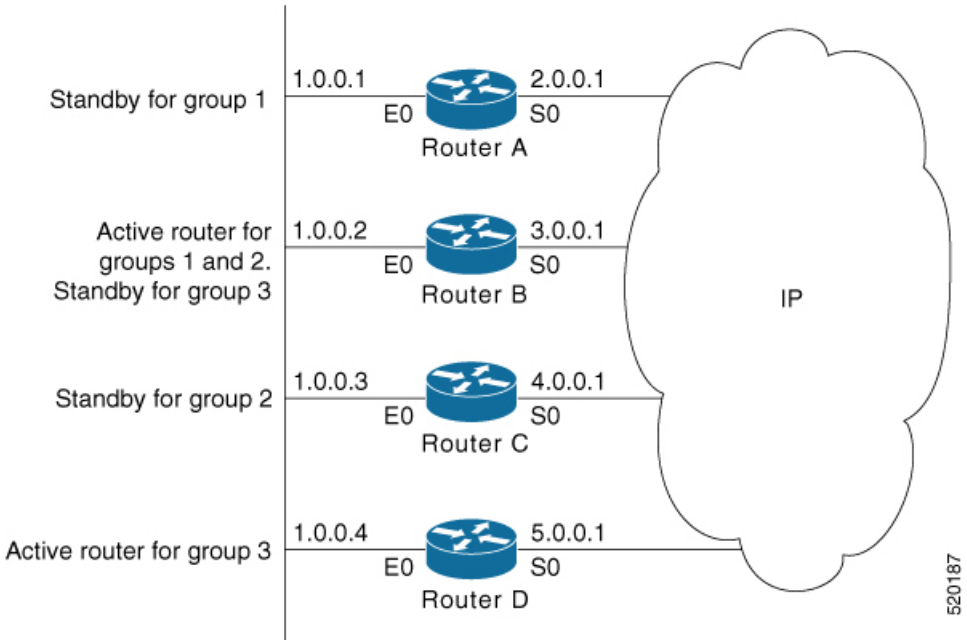
Figure 16: Routers configured as an HSRP group



All hosts on the network are configured to use the IP address of the virtual router (in this case, 1.0.0.3) as the default gateway.

A single router interface can also be configured to belong to more than one HSRP group. The following figure shows routers configured as members of multiple HSRP groups.

Figure 17: Routers configured as members of multiple HSRP groups



In the figure above, the Ethernet interface 0 of Router A belongs to group 1. Ethernet interface 0 of Router B belongs to groups 1, 2, and 3. The Ethernet interface 0 of Router C belongs to group 2, and the Ethernet interface 0 of Router D belongs to group 3. When you establish groups, you might want to align them along departmental organizations. In this case, group 1 might support the Engineering Department, group 2 might support the Manufacturing Department, and group 3 might support the Finance Department.

Router B is configured as the active router for groups 1 and 2 and as the standby router for group 3. Router D is configured as the active router for group 3. If Router D fails for any reason, Router B assumes the packet-transfer functions of Router D and maintains the ability of users in the Finance Department to access data on other subnets.

Note

A different virtual MAC address (VMAC) is required for each sub interface. VMAC is determined from the group ID. Therefore, a unique group ID is required for each sub interface configured, unless the VMAC is configured explicitly.

Note

We recommend that you disable Spanning Tree Protocol (STP) on switch ports to which the virtual routers are connected. Enable RSTP or rapid-PVST on the switch interfaces if the switch supports these protocols.

HSRP and ARP

This topic describes how HSRP routers use ARP responses to advertise the virtual IP and virtual MAC addresses when a router in an HSRP group becomes active.

When a router in an HSRP group goes active, it sends a number of ARP responses containing its virtual IP address and the virtual MAC address. These ARP responses help switches and learning bridges update their port-to-MAC maps. These ARP responses also provide routers configured to use the burned-in address of the interface as its virtual MAC address (instead of the preassigned MAC address or the functional address) with a means to update the ARP entries for

the virtual IP address. Unlike the gratuitous ARP responses sent to identify the interface IP address when an interface comes up, the HSRP router ARP response packet carries the virtual MAC address in the packet header. The ARP data fields for IP address and media address contain the virtual IP and virtual MAC addresses.

Preemption

This topic describes the HSRP preemption feature, which enables the router with the highest priority to immediately become the active router.

The HSRP preemption is an HSRP configuration feature that

- allows the router with the highest priority to immediately become the active router
- determines priority first by the priority value that you configure, and then by the IP address, and
- manages state transitions through specific messaging.

How state transition process for HSRP preemption works

This topic describes how HSRP evaluates router priority and transitions the active role when a higher-priority router joins the network.

The network architecture identifies the active router based on priority values, where the device with the higher value holds the primary position. If a higher-priority router joins the network, it assumes the active role by sending a coup message. The lower-priority active router receives this notification and relinquishes its position to ensure the most capable device manages traffic.

Summary

Workflow

These stages describe the state transition process for HSRP preemption:

1. The system evaluates router priority based on the configured priority value and the IP address.
2. A higher-priority router sends a coup message to assert its status as the active device.
3. The lower-priority active router changes to the speak state and sends a resign message upon receiving a coup or hello message from a higher-priority router.

ICMP redirect messages

This topic describes how ICMP redirect messages are filtered through HSRP so that the next-hop IP address is changed to an HSRP virtual IP address, preventing hosts from discovering the real MAC addresses of routers in the HSRP group.

The ICMP redirect for HSRP is a network function that

- facilitates error reporting through IP processing
- prevents host discovery of individual router MAC addresses, and
- maps real IP addresses to virtual IP addresses for traffic redirection.

ICMP provides diagnostic support within the network layer, but it poses risks to network reliability if hosts learn the real MAC addresses of physical routers. If a host relies on a specific physical interface that subsequently fails, the host loses its connection to the network. HSRP addresses this vulnerability by filtering outgoing ICMP redirect messages to ensure that the next-hop address points to an HSRP virtual IP address rather than a physical one.

The router performs these tasks to ensure traffic continuity:

- The protocol filters outgoing ICMP redirect messages to replace the next-hop address with the HSRP virtual IP address.
- The router maintains a mapping between real IP addresses and virtual IP addresses to manage filtering effectively.
- HSRP tracks individual router advertisements to maintain an accurate understanding of the network topology.

HSRP over BVI

This topic describes how HSRP runs over Bridge-Group Virtual Interfaces (BVIs) to provide default gateway redundancy for a LAN, including supported scale, topology, and systems.

Bridge Group Virtual Interface (BVI) is a virtual interface which provides Layer 3 or routed functionality to a bridge group. Layer 2 functionality is applicable to the interfaces which are part of a bridge group and BVI is the routed interface for that bridge group.

The HSRP and BVI configuration is a redundancy mechanism that

- provides resilient gateway services for a local area network
- allows multiple routers to share a single virtual IP address, and
- enables Layer 3 functionality for bridge groups.

Table 77: Feature History Table

Feature Name	Release Information	Feature Description
HSRP over BVI	Release 26.2.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) *This feature is supported on: • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
HSRP over BVI	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.
HSRP over BVI	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) * This feature is supported on: • 8712-MOD-M • 8711-48Z-M

Feature Name	Release Information	Feature Description
HSRP over BVI	Release 25.2.1	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH
HSRP over BVI	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
HSRP over BVI	Release 7.5.2	Hot Standby Router Protocol (HSRP) runs on top of interfaces of multiple routers in the same home network that has only Cisco routers. It allows a group of routers to behave as a single virtual default gateway router, thereby providing default gateway redundancy and minimizing traffic loss. HSRP now supports Bridge-Group Virtual Interface (BVI) on Cisco Silicon One Q100 systems, which means that HSRP sessions can run between BVI interfaces of multiple routers.

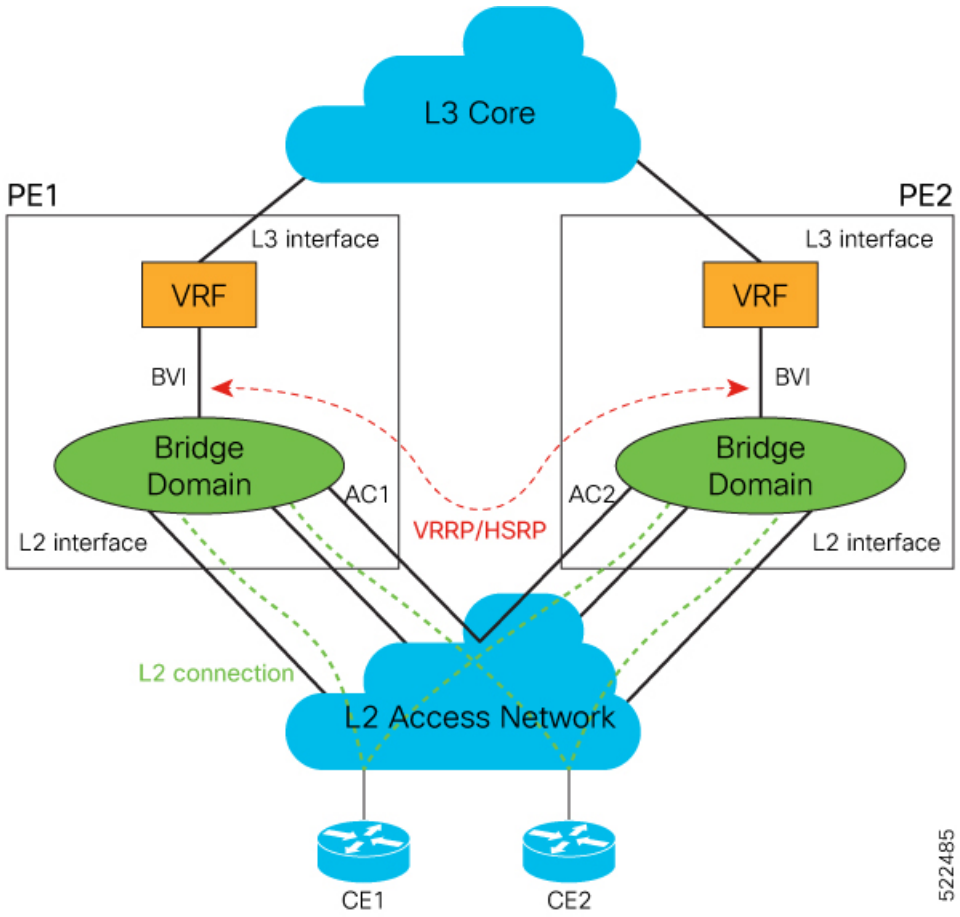
Participating routers function as a single virtual entity, which maintains network availability even if an individual device fails. These routers share a virtual IP address and a corresponding virtual MAC address, which hosts identify as the default first-hop gateway. While the active router handles packet forwarding, the standby router monitors the active device and assumes its duties if a failure occurs. The Bridge Group Virtual Interface (BVI) extends this capability by acting as a routed interface that provides Layer 3 functionality to members of a bridge group.

The following components define the operation of this redundancy configuration:

- Participating routers share a virtual IP address and virtual MAC address to act as a single gateway for connected hosts.
- The active router forwards all traffic while the standby router maintains readiness to take over in the event of a failure.
- The BVI provides a routed interface that allows Layer 3 operations within a bridge group.

Topology

This topology showcases how HSRP functions over BVI.



In this topology, PE1 and PE2 are paired in a redundant group. This group provides Layer 3 gateway service to CE1 and CE2. HSRP is configured over BVI interfaces on PE1 and PE2. HSRP ensures one BVI is the active gateway. The other is the standby gateway.

You can configure one of the BVIs to be active and the other BVI as standby by setting the HSRP priority value. The active BVI is programmed with the virtual MAC address chosen by HSRP. Hosts, CE1 and CE2 send the traffic to the virtual destination MAC address and the active BVI forwards the traffic.

During failover, the standby BVI becomes active and is programmed with the virtual MAC address. The traffic from the hosts is forwarded through this active BVI.

Supported scale and systems

HSRP over Bridge Virtual Interfaces (BVIs) is supported:

- You can configure upto 512 HSRP groups (IPv4 and IPv6 combined) over BVIs on:
  - Cisco Silicon One Q100 ASIC-based systems
  - Cisco Silicon One Q200 ASIC-based systems
  - Cisco Silicon One P100 ASIC-based systems
- Where the underlay IRB bridge domains consist of bridge members on L2 main or subinterfaces. Only physical and bundle interfaces are supported for L2 bridging in IRB.
- For IPv4 and IPv6 configurations, in both the default and VRF tables.
- On both the fixed and distributed systems.
- HSRP on BVI is supported on the 88-LC1-12TH24FH-E line card from Cisco IOS XR Release 25.1.1.

Restrictions for HSRP over BVI

This topic lists the restrictions to consider before you configure HSRP over Bridge-Group Virtual Interfaces (BVIs).

Consider these restrictions before you configure HSRP over BVI:

- The minimum supported HSRP Hello timer is 100 ms. At the minimum timer, a total of 50 sessions are supported. Above 100 ms timers, the sessions scale goes up proportionately. A maximum of 1024 HSRP groups and 1024 HSRP sessions are supported.
- HSRP on BVI is not supported on the 88-LC1-12TH24FH-E line card in Cisco IOS XR Release 24.3.1.

Configure HSRP over BVI

This topic describes how to configure and verify HSRP sessions over Bridge-Group Virtual Interfaces (BVIs) on PE1 and PE2 routers for IPv4 and IPv6 addresses.

To configure HSRP sessions over BVI, complete the following configurations on PE1 and PE2 in the specified order:

1. Configure a set of interfaces as Layer 2 interfaces and a set of VLAN sub-interfaces.
2. Configure a bridge group.
3. Configure a BVI.
4. Configure HSRP over BVI.

Procedure

1. Enter the global configuration mode and configure a set of interfaces as Layer 2 interfaces and a set of VLAN sub-interfaces.

   Example

```
Router# configure
Router(config)# interface HundredGigE0/0/1/0.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# interface HundredGigE0/0/1/1.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
```

2. Enter the Layer 2 VPN configuration mode and configure a bridge group.

   Example

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 5
Router(config-l2vpn-bg)# bridge-domain 5
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/1/0.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/1/1.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# routed interface BVI 10
```

```
Router(config-l2vpn-bg-bd-bvi)# commit
Router(config-l2vpn-bg-bd-bvi)# exit
```

3. Configure a BVI in the global configuration mode.

   Example

```
Router(config)# interface BVI 10
Router(config-if)# ipv4 address 209.165.200.225 255.255.255.0
Router(config-if)# ipv6 address 2001:DB8:A:B::1/64
Router(config-if)# commit
```

4. Configure HSRP over BVI in the global configuration mode for the IPv4 address.

   Example

```
Router(config)# router HSRP
Router(config-hsrp)# interface BVI 10
Router(config-hsrp-if)# address-family ipv4
Router(config-hsrp-ipv4)# HSRP 10
Router(config-hsrp-gp)# priority 101
Router(config-hsrp-gp)# address 209.165.200.226
Router(config-hsrp-gp)# commit
```

5. Configure HSRP over BVI in the global configuration mode for the IPv6 address.

   Example

```
Router(config)# router HSRP
Router(config-hsrp)# interface BVI 10
Router(config-hsrp-if)# address-family ipv6
Router(config-hsrp-ipv6)# HSRP 11
Router(config-hsrp-gp)# address global 2001:DB8:A:B::2
Router(config-hsrp-gp)# address linklocal autoconfig
Router(config-hsrp-gp)# commit
```

6. Verify the bridge domain details by using the **show l2vpn bridge-domain detail** command.

   Example

```
Router# show l2vpn bridge-domain detail

Legend: pp = Partially Programmed.
Bridge group: 5, bridge-domain: 5, id: 1, state: up, ShgId: 0, MSTi: 0
Coupled state: disabled
VINE state: BVI Resolved
MAC learning: enabled
MAC withdraw: enabled
MAC withdraw for Access PW: enabled
MAC withdraw sent on: bridge port up
MAC withdraw relaying (access to access): disabled
Flooding:
Broadcast & Multicast: enabled
Unknown unicast: enabled
MAC aging time: 300 s, Type: inactivity
MAC limit: 32768, Action: none, Notification: syslog
MAC limit reached: no, threshold: 75%
```

```
MAC port down flush: enabled
MAC Secure: disabled, Logging: disabled
Split Horizon Group: none
Dynamic ARP Inspection: disabled, Logging: disabled
IP Source Guard: disabled, Logging: disabled
DHCPv4 Snooping: disabled
DHCPv4 Snooping profile: none
IGMP Snooping: disabled
IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control: disabled
Bridge MTU: 1500
MIB cvplsConfigIndex: 2
Filter MAC addresses:
P2MP PW: disabled
Multicast Source: Not Set
Create time: 26/05/2020 17:08:54 (00:11:30 ago)
No status change since creation
ACs: 3 (3 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
List of ACs:
AC: BVI10, state is up
Type Routed-Interface
MTU 1514; XC ID 0x80000001; interworking none
BVI MAC address:
c472.95a6.8b90
Virtual MAC addresses:
0000.5e00.010a
0000.5e00.020b
Split Horizon Group: Access
AC: HundredGigE0/0/1/0.1, state is up
Type VLAN; Num Ranges: 1
Rewrite Tags: []
VLAN ranges: [1, 1]
MTU 1500; XC ID 0x1; interworking none
MAC learning: enabled
```

7. Verify HSRP details for IPv4 by using the **show hsrp ipv4 detail** command.

Example

```
Router# show hsrp ipv4 detail

BVI10 - IPv4 vrID 10
State is Master
2 state changes, last state change 00:11:57
State change history:
May 26 17:08:59.470 UTC Init -> Backup Delay timer expired
May 26 17:09:03.075 UTC Backup -> Master Master down timer expired
Last resign sent: Never
Last resign received: Never
Virtual IP address is 209.165.200.226
Virtual MAC address is 0000.5E00.010a, state is active
Master router is local
Version is 2
Advertise time 1 secs
Master Down Timer 3.605 (3 x 1 + (155 x 1/256))
Minimum delay 1 sec, reload delay 5 sec
Current priority 101
Configured priority 101, may preempt
minimum delay 0 secs
```

8. Verify HSRP details for IPv6 by using the **show hsrp ipv6 detail** command.

Example

```
Router# show hsrp ipv6 detail

BVI10 - IPv6 vrID 11
State is Master
2 state changes, last state change 00:04:29
State change history:
May 26 17:16:43.476 UTC Init -> Backup Virtual IP configured
May 26 17:16:47.085 UTC Backup -> Master Master down timer expired
Last resign sent: Never
Last resign received: Never
Virtual IP address is fe80::200:5eff:fe00:20b
Secondary Virtual IP address is 2001:db8:a:b::2
Virtual MAC address is 0000.5E00.020b, state is active
Master router is local
Version is 3
Advertise time 1 secs
Master Down Timer 3.609 (3 x 1 + (156 x 1/256))
Minimum delay 1 sec, reload delay 5 sec
Current priority 100
Configured priority 100, may preempt
minimum delay 0 secs
```

You have successfully configured and verified HSRP over BVI for IPv4 and IPv6 addresses.

HSRP over physical and bundle interfaces

This topic describes HSRP support for IPv4 and IPv6 networks on physical interfaces, physical sub-interfaces, bundle interfaces, and bundle sub-interfaces, including supported scale and topology.

Starting Cisco IOS XR Release 24.2.11, HSRP is supported on Physical interfaces, Physical sub-interfaces, Bundle interfaces, and Bundle sub-interfaces on the Cisco Silicon One P100 and Cisco Silicon One Q200 ASIC-based systems.

You can configure up to:

- 510 HSRP groups (IPv4 and IPv6 combined) on the Cisco Silicon One Q200 ASIC-based and Cisco Silicon One P100 ASIC-based systems.

HSRP sessions run on top of interfaces of the multiple routers which are in the same home network.

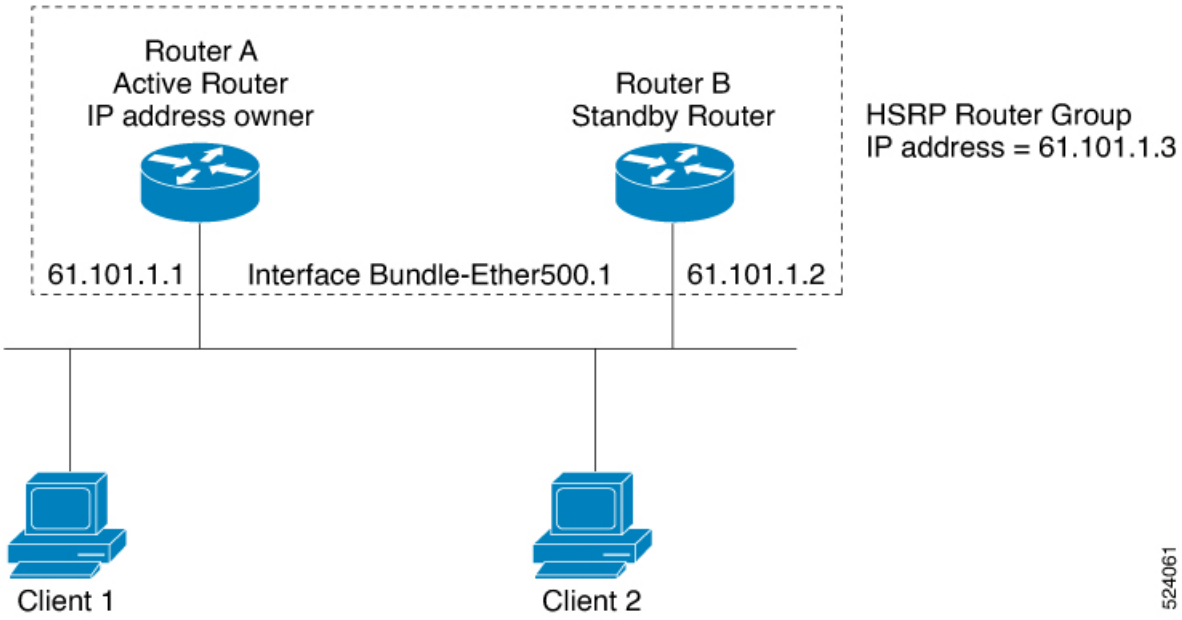
Table 78: Feature History Table

Feature Name	Release Information	Feature Description
HSRP over Physical interfaces and Bundle 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

Feature Name	Release Information	Feature Description
HSRP over Physical interfaces and 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
HSRP over Physical interfaces and Bundle interfaces	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
HSRP over Physical interfaces and Bundle interfaces	Release 24.2.11	This feature provides first-hop redundancy and enables failover to a standby interface within a group of physical or bundle interfaces or sub-interfaces in a network in the event of any failure in the active interface or sub-interface in that group. The feature allows you to configure HSRP for IPv4 and IPv6 networks on the physical and bundle interfaces and sub-interfaces.

Topology

This topology provides the HSRP implementation over a Bundle sub-interface on a network segment.



HSRP Topology and underlying network elements:

- Interface Bundle-Ether500.1: This is the Bundle sub-interface which is configured on the active router (IP Address: 61.101.1.1) and the standby router (IP Address: 61.101.1.2).
- HSRP Router Group: The HSRP router group consists of two or more routers configured with HSRP. This group provides backup services to the standby router (Router B) in the event of the active router (Router A) failure. The virtual IP address of the HSRP router group in this topology is 61.101.1.3.
- Router A: Router A is the active router on which the Bundle sub-interface Bundle-Ether500.1 is configured. This router receives and routes packets destined for the MAC address of the group. This router is known as the IP address owner of the HSRP router group.
- Router B: When the designated active router fails, HSRP detects it and selects another router with the highest priority in the HSRP router group as the standby router. This standby router with the Bundle sub-interface Bundle-Ether500.1 configuration takes over the job of Router A and assumes control of the MAC and IP Addresses of the HSRP router group.
- Client 1 and Client 2: These are source hosts from which traffic is forwarded through the active router or standby router to the destination hosts.

Guidelines for HSRP configuration

This topic describes the HSRP group ID range and the LPTS flow parameter adjustments required when you configure HSRP for higher scale of sessions or with aggressive timer intervals.

Consider these usage guidelines when you configure HSRP:

- Group ID value must be within the 0-255 range. However, RFC indicates 0-4095 as the id range.
- For a higher scale of HSRP IPv6 group sessions, set the LPTS default UDP entry policer rate to 3000 or higher if you see LPTS drops when running the **show lpts pifib hardware police location 0/0/CPU0** command. For information about this configuration, see the *Usage Guidelines* section in the **lpts pifib hardware police** command.
- When setting up HSRP for both IPv4 and IPv6 groups with more than 510 sessions, or when using aggressive timer intervals below 1 second, it is crucial to adjust the LPTS flow parameters accordingly. Specifically, increase the LPTS policer rate to a minimum of 3000 pps. This ensures optimal performance and avoids any LPTS drops for HSRP traffic to maintain network stability and efficiency. For detailed instructions on configuring these settings, refer to the *Usage Guidelines* section within the **lpts pifib hardware police** command documentation.

Configure HSRP for IPv4 networks on physical and bundle interfaces

This topic describes how to configure and verify HSRP for IPv4 networks on physical interfaces, physical sub-interfaces, bundle interfaces, and bundle sub-interfaces.

The **hsrp ipv4** command activates HSRP on the configured interface. If an IP address is specified, that address is used as the designated address for the Hot Standby group. If no IP address is specified, the virtual address is learned from the active router. For HSRP to elect a designated router, at least one router in the Hot Standby group must have been configured with, or learned, the designated address. Configuring the designated address on the active router always overrides a designated address that is currently in use.

Perform the following configuration on both the active router and the standby router. On each router, first configure the main bundle interface (*Bundle-Ether500*) and add the member links to *Bundle-Ether500*.

Procedure

1. On the active router, enter the interface configuration mode and configure an IPv4 address for the interface.

   Example

   ```
   Router(config)# interface Bundle-Ether500.1
   Router(config-if)# ipv4 address 61.101.1.1/24
   Router(config-if)# encapsulation dot1q 1
   Router(config-if)# no shut
   Router(config-if)# commit
   Router(config-if)# exit
   ```

2. Enter the HSRP configuration mode, add the configured interface, and configure HSRP for IPv4.

   Example

   ```
   Router(config)# router hsrp
   Router(config-hsrp)# interface Bundle-Ether500.1
   Router(config-hsrp-if)# address-family ipv4 hsrp version 2
   Router(config-hsrp-virtual-router)# address 61.101.1.3
   ```

3. Set a priority for the virtual router.

   Example

   ```
   Router(config-hsrp-virtual-Router)# priority 100
   ```

4. Configure a preempt delay value that controls the selection of the IP address owner virtual router, and commit the configuration.

   Example

   ```
   Router(config-hsrp-virtual-Router)# preempt delay 30
   Router(config-hsrp-virtual-Router)# commit
   ```

5. On the standby router, repeat the preceding steps by using an IPv4 address that is unique to the standby router but sharing the same HSRP virtual address.

Example

```
Router(config)# interface Bundle-Ether500.1
Router(config-if)# ipv4 address 61.101.1.2/24
Router(config-if)# encapsulation dot1q 1
Router(config-if)# no shut
Router(config-if)# commit
Router(config-if)# exit
Router(config)# router hsrp
Router(config-hsrp)# interface Bundle-Ether500.1
Router(config-hsrp-if)# address-family ipv4 hsrp version 2
Router(config-hsrp-virtual-router)# address 61.101.1.3
Router(config-hsrp-virtual-Router)# priority 100
Router(config-hsrp-virtual-Router)# preempt delay 30
Router(config-hsrp-virtual-Router)# commit
```

6. Verify the interface configuration on the active router by using the **show run interface** command.

Example

```
Router(config-hsrp-virtual-router)# do show run interface Bundle-Ether500.1
interface Bundle-Ether500.1
 ipv4 address 61.101.1.1/24
 encapsulation dot1q 1
!
```

7. Verify the HSRP configuration by using the **show running-configuration router hsrp** command.

Example

```
Router(config)# show running-configuration router hsrp
 router hsrp
interface Bundle-Ether500.1
  address-family ipv4
   encapsulation dot1q
    hsrp 1 version 2
     preempt
     priority 100
     address 61.101.1.3
   !
  !
```

8. Verify HSRP operation by using the **show hsrp detail** command.

Example

```
Router# show hsrp detail
Bundle-Ether500.1 - IPv4 Group 1 (version 1)
  Local state is Active, priority 100, may preempt
  Hellotime 3000 msec holdtime 10000 msec
  Next hello sent in 2.373
  Minimum delay 1 sec, reload delay 5 sec
  Hot standby IP address is 61.101.1.3 configured
  Active router is local
  Standby router is 61.101.1.2 expires in 00:00:08
  Standby virtual mac address is 0000.0c07.ac01, state is active
    15 state changes, last state change 00:01:17
    State change history:
    Feb 16 04:56:48.724 UTC  Listen   -> Active   Lower priority active
```

```
received
    Feb 16 05:00:29.439 UTC  Active   -> Init     Interface Down update
    Feb 17 05:27:13.516 UTC  Init     -> Listen   Delay timer expired
    Feb 17 05:27:14.643 UTC  Listen   -> Active   Lower priority active
received
    Feb 17 05:53:53.500 UTC  Active   -> Init     Interface Down update
    Feb 17 05:58:49.641 UTC  Init     -> Listen   Delay timer expired
    Feb 17 05:58:51.031 UTC  Listen   -> Active   Lower priority active
received
    Feb 17 06:25:20.589 UTC  Active   -> Init     Interface Down update
    Feb 17 06:31:13.783 UTC  Init     -> Listen   Delay timer expired
    Feb 17 06:31:14.954 UTC  Listen   -> Active   Lower priority active
received
 Last coup sent:       Feb 17 06:31:14.954 UTC
 Last coup received:   Never
```

You have successfully configured and verified HSRP for IPv4 networks on the active and standby routers.

Configure HSRP for IPv6 networks on physical and bundle interfaces

This topic describes how to configure and verify HSRP for IPv6 networks on physical interfaces, physical sub-interfaces, bundle interfaces, and bundle sub-interfaces.

Perform the following configuration on both the active router and the standby router. On each router, first configure the main bundle interface (*Bundle-Ether500*) and add the member links to *Bundle-Ether500*.

Note

- The virtual linklocal address must not match any other virtual linklocal address that is already configured for a different group.
- The virtual linklocal address must not match the interface linklocal IPv6 address.
- If you use the **autoconfig** keyword, the linklocal address is calculated using the EUI-64 format.
- Use the **legacy-compatible** keyword to be compatible with Cisco IOS and other legacy Cisco devices.
- If you configure HSRP for IPv6, you must configure a linklocal IPv6 address or enable it using the **autoconfig** keyword. If you do not configure a linklocal IPv6 address, the router does not accept the configuration when you commit your changes.

Procedure

1. On the active router, enter the interface configuration mode and configure an IPv6 address for the interface.

   Example

```
Router(config)# interface Bundle-Ether500.1
Router(config-if)# ipv6 address 61:101:1:1::1:1/64
Router(config-if)# encapsulation dot1q 1
Router(config-if)# no shut
```

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

2. Enter the HSRP configuration mode, add the configured interface, and configure HSRP for IPv6 by assigning a global IPv6 address and a linklocal address.

   Example

```
Router(config)# router hsrp
Router(config-hsrp)# interface Bundle-Ether500.1
Router(config-hsrp-if)# address-family ipv6 hsrp version 2
Router(config-hsrp-virtual-Router)# address global 61:101:1:1::1:3
Router(config-hsrp-virtual-Router)# address linklocal autoconfig
```

3. Set a priority for the virtual router.

   Example

```
Router(config-hsrp-virtual-Router)# priority 100
```

4. Configure a preempt delay value that controls the selection of the IP address owner virtual router, and commit the configuration.

   Example

```
Router(config-hsrp-virtual-Router)# preempt delay 30
Router(config-hsrp-virtual-Router)# commit
```

5. On the standby router, repeat the preceding steps by using an IPv6 address that is unique to the standby router but sharing the same HSRP virtual address.

   Example

```
Router(config)# interface Bundle-Ether500.1
Router(config-if)# ipv6 address 61:101:1:1::1:2/64
Router(config-if)# encapsulation dot1q 1
Router(config-if)# no shut
Router(config-if)# commit
Router(config-if)# exit
Router(config)# router hsrp
Router(config-hsrp)# interface Bundle-Ether500.1
Router(config-hsrp-if)# address-family ipv6 hsrp version 2
Router(config-hsrp-virtual-Router)# address global 61:101:1:1::1:3
Router(config-hsrp-virtual-Router)# address linklocal autoconfig
Router(config-hsrp-virtual-Router)# priority 100
Router(config-hsrp-virtual-Router)# preempt delay 30
Router(config-hsrp-virtual-Router)# commit
```

6. Verify the interface configuration on the active router by using the **show run interface** command.

   Example

```
Router(config-hsrp-virtual-router)# do show run interface Bundle-Ether500.1
interface Bundle-Ether500.1
 ipv6 address 61:101:1:1::1:1/64
```

```
 encapsulation dot1q 1
!
```

7. Verify the HSRP configuration by using the **show running-configuration router hsrp** command.

   Example

```
Router(config)# show running-configuration router hsrp
 router hsrp
interface Bundle-Ether500.1
  address-family ipv6
   encapsulation dot1q
    hsrp 1 version 2
     preempt
     priority 100
     address 61:101:1:1::1:3
   !
  !
```

8. Verify HSRP operation by using the **show hsrp detail** command.

   Example

```
Router# show hsrp detail
Bundle-Ether500.1 - IPv6 Group 1 (version 2)
  Local state is Active, priority 100, may preempt
  Hellotime 3000 msec holdtime 10000 msec
  Next hello sent in 0.132
  Minimum delay 1 sec, reload delay 5 sec
  Hot standby IP address is 61:101:1:1::1:3 configured
  Active router is local
  Standby router is 61:101:1:1::1:2 expires in 00:00:08
  Standby virtual mac address is 0005.73a0.0001, state is active
    15 state changes, last state change 00:01:17
    State change history:
    Feb 16 04:56:50.094 UTC  Listen   -> Active   Lower priority active
received
    Feb 16 05:00:29.464 UTC  Active   -> Init     Interface Down update
    Feb 17 05:27:13.530 UTC  Init     -> Listen   Delay timer expired
    Feb 17 05:27:13.897 UTC  Listen   -> Active   Lower priority active
received
    Feb 17 05:53:53.520 UTC  Active   -> Init     Interface Down update
    Feb 17 05:58:49.616 UTC  Init     -> Listen   Delay timer expired
    Feb 17 05:58:52.112 UTC  Listen   -> Active   Lower priority active
received
    Feb 17 06:25:20.614 UTC  Active   -> Init     Interface Down update
    Feb 17 06:31:13.902 UTC  Init     -> Listen   Delay timer expired
    Feb 17 06:31:14.985 UTC  Listen   -> Active   Lower priority active
received
Last coup sent:       Feb 17 06:31:14.985 UTC
Last coup received:   Never
Last resign sent:     Feb 17 06:25:20.614 UTC
Last resign received: Never
```

You have successfully configured and verified HSRP for IPv6 networks on the active and standby routers.

15 Transport Protocol Operations

Topics:

Summarizes TCP, UDP, and RAW transport behavior, NSR session preservation, failover recovery, and TCP dump file inspection and conversion for troubleshooting.

Use this chapter to understand transport behavior, maintain supported sessions during failures, and examine TCP packet traces.

The chapter covers these transport protocol operations:

- TCP, UDP, and RAW transport behavior for application and protocol traffic.
- Nonstop routing (NSR) session preservation and failover recovery during supported route processor or process failures.
- TCP packet trace storage, display, and conversion for troubleshooting.

Use the transport topics to compare protocol behavior, configure NSR recovery, review NSR restrictions, and convert or display TCP dump files.

TCP, UDP, and RAW transports

Explains the connection, delivery, and application characteristics of TCP, UDP, and RAW transports and identifies when their default settings are sufficient.

TCP, UDP, and RAW transports are Internet Protocol (IP) transport options that

- provide connection-oriented or connectionless data exchange
- carry traffic for multiple applications and protocols, and
- use default settings that meet the requirements of most sites.

Transmission Control Protocol (TCP) is a connection-oriented protocol that defines the data and acknowledgment formats that two systems exchange. It also defines procedures that help the systems deliver data correctly.

TCP allows multiple applications on one system to communicate concurrently by demultiplexing incoming traffic among the applications.

User Datagram Protocol (UDP) is a connectionless transport-layer protocol in the IP family. Application-layer protocols that use UDP include Network File System (NFS), Simple Network Management Protocol (SNMP), Domain Name System (DNS), and TFTP.

A RAW transport carries an IP protocol other than TCP or UDP.

Nonstop routing for transport sessions

Explains how nonstop routing maintains supported routing protocol sessions during route processor failover by synchronizing application and transport state, and describes supported clients, failure handling, restrictions, and recovery behavior.

Nonstop routing (NSR) is a high-availability capability that

- keeps supported routing protocol sessions active during route processor failover
- synchronizes application and transport state between active and standby route processors, and
- reduces session loss and packet loss during supported failures.

NSR supports Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), and Label Distribution Protocol (LDP) during route processor failover. It also supports an OSPF, LDP, or TCP process crash and online insertion and removal.

Table 79: Feature History Table

Feature Name	Release	Description
Nonstop routing	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 with 8404-RSP1-48-EM route processor.

Feature Name	Release	Description
Nonstop routing	Release 24.4.1	Introduced in this release on: Centralized Systems (8600 [ASIC: Q200], (select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) NSR ensures uninterrupted routing by maintaining protocol sessions during process crashes, thus enhancing network reliability. It eliminates session loss and minimizes packet drop by preserving the router's state and synchronizing it with standby processes. This reliability is particularly beneficial for maintaining stable connections in dynamic environments. By automatically synchronizing routing information, NSR provides seamless network operation and reduces the need for manual intervention during failures. This feature is essential for environments demanding continuous network availability and high resilience to faults. *Previously this feature was supported on Q200 and Q100. *This feature is now supported on: • Cisco 8608 router with 8608-RP route processor • Cisco 8800 series routers with 8800-RP or 8800-RP2 router processors

Related routing information

For OSPF-specific NSR information, see the chapter about implementing OSPF in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

How NSR preserves sessions during failures

Describes how nonstop routing synchronizes application and TCP state, migrates active sessions during route processor failover, and restores protection after a standby process failure.

Nonstop routing (NSR) keeps supported routing sessions and transport state synchronized between the active and standby route processors.

Summary

The session-preservation process uses these components:

- Active route processor: Runs the active routing applications and TCP sessions.
- Standby route processor: Maintains synchronized application and transport state.
- NSR-enabled clients: Use the synchronized state to preserve their protocol sessions.

Workflow

These stages describe how NSR preserves sessions during failures:

1. During normal operation, NSR mirrors the application and TCP state from the active route processor to the standby route processor.
2. When a route processor fails over, the synchronized TCP sessions migrate to the standby route processor without notifying their peers.
3. When an active NSR-enabled process crashes and process-failure switchover is configured, the router can use route processor failover as the recovery action.

4. A fault-management policy can also handle an active process failure. The failing process does not generate the logs required to determine the root cause when this policy handles the failure.
5. When a standby NSR-enabled process restarts or crashes, active sessions remain established, but NSR protection is unavailable until the standby instances restart and resynchronize the sessions.

Result

Supported protocol sessions remain established across route processor failover. Without NSR, protocols on the new active route processor must reestablish their sessions. Graceful restart can reduce traffic loss, but it does not provide the same session continuity.

NSR restrictions and supported clients

Lists the process-failure conditions that trigger route processor switchover and identifies the supported NSR clients covered by this recovery feature.

Review the restrictions and client support before you configure process-failure switchover for nonstop routing (NSR).

Process-failure switchover has these restrictions:

- The router does not trigger route processor failover when a process restarts, even when **nsr process-failures switchover** is configured.
- The router triggers route processor failover only when a process crashes.

The process-failure switchover feature supports these NSR clients:

- Layer 2 VPN (L2VPN)
- Virtual Private Dialup Network (VPDN)
- TCP
- Intermediate System-to-Intermediate System (IS-IS)
- Open Shortest Path First (OSPF)
- Multiprotocol Label Switching Label Distribution Protocol (MPLS LDP)
- Border Gateway Protocol (BGP)

Configure failover as a recovery action

Configures route processor failover as the recovery action when active TCP or an active NSR-enabled application crashes, and verifies the committed configuration.

Use this procedure to maintain nonstop routing (NSR) by switching active instances to their standby instances after a supported process crash.

When active TCP or an NSR client of active TCP crashes, its TCP sessions go down. With process-failure switchover configured, the router initiates route processor switchover when active TCP or an active application, such as Label Distribution Protocol (LDP) or Open Shortest Path First (OSPF), crashes.

For information about configuring Multiprotocol Label Switching LDP for NSR, see the *MPLS Configuration Guide for Cisco 8000 Series Routers*. For per-process NSR configuration, see the *Routing Configuration Guide for Cisco 8000 Series Routers*.

Before you begin

You must belong to a user group associated with a task group that includes the required task IDs. If a user group assignment prevents command access, contact your authentication, authorization, and accounting administrator.

Procedure

1. Configure process-failure switchover.

   Example

   ```
   Router# configure
   Router(config)# nsr process-failures switchover
   Router(config)# commit
   ```

2. Verify the process-failure switchover configuration.

   Example

   ```
   Router# show running-configuration nsr process-failures switchover
   nsr process-failures switchover
   ```

The router uses route processor failover as the recovery action when a supported active NSR-enabled process crashes.

TCP dump file conversion

Explains how the TCP dump file converter supports troubleshooting by converting stored binary TCP packet traces into text or pcap format, and describes feature behavior, storage considerations, platform support, and available trace-viewing methods.

The TCP dump file converter is a troubleshooting tool that

- converts binary `tcp_ios-xr` dump files into text or pcap format
- processes all stored packet traces in one operation, and
- supports analysis with command-line, third-party, or open-source tools.

When nonstop routing (NSR) is disabled or a session flaps, TCP stores the 200 most recent packet traces in binary files in a temporary folder. The traces can contain data about configured routing protocols and network traffic that traverses the router.

Table 80: Feature History Table

Feature Name	Release Information	Description
TCP dump file converter	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

Feature Name	Release Information	Description
TCP dump file converter	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
TCP dump file converter	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 supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
TCP dump file converter	Release 24.2.11	You can now convert an entire TCP dump of packet traces in binary files into readable formats such as text or pcap, which makes it easier to analyze them for troubleshooting using third-party or open-source tools. This feature saves time and effort by preventing the need to examine each packet for failure. This feature introduces the **tcp dump-file convert**command.

Packet trace viewing methods

You can inspect stored TCP packet traces in these ways:

- Use **tcp dump-file convert** to convert all stored binary files into pcap, text, or both formats. This method supports efficient analysis of multiple files.
- Use **show tcp dump-file <binary-filename>** to display one binary file at a time in text format.

TCP dump file storage and conversion limitations

Lists packet trace retention, single-file display, and protocol coverage limitations that apply to stored TCP dump files and manual viewing.

Review these limitations before you inspect or convert TCP dump files.

TCP dump file storage and viewing have these limitations:

- The router stores only the 200 most recent message exchanges that occurred immediately before session termination, while NSR was disabled, or during a session flap.
- The **show tcp dump-file <binary-filename>** command displays only one binary file at a time in text format.

- When NSR is disabled, the router stores TCP dump files only for major protocols such as Border Gateway Protocol (BGP), Multicast Source Discovery Protocol (MSDP), and Multiprotocol Label Switching Label Distribution Protocol (MPLS LDP).

Convert TCP dump files to readable formats

Converts stored binary TCP packet traces into text and pcap files, displays converted text in the CLI, and prepares the output for remote analysis.

Use this procedure to convert all stored binary TCP packet traces into text and pcap files for troubleshooting.

By default, the router stores converted files in the `/harddisk:/decoded_dumpfiles` directory.

Procedure

1. Convert all stored TCP dump files into text and pcap formats.

   Example

   ```
   Router# tcp dump-file convert all-formats all
   ascii file is saved at:
   /harddisk:/decoded_dumpfiles/text_tcpdump_peer_all_node0_RP0_CPU0_2024_3_19_10_8_53.462070.txt
   pcap file is saved at:
   /harddisk:/decoded_dumpfiles/pcap_tcpdump_peer_all_node0_RP0_CPU0_2024_3_19_10_8_40.154838.pcap
   [OK]
   ```

   Use the **location node-id** and **file** <**file-path**> keywords to select a node and destination path.

   ```
   Router# tcp dump-file convert all-formats all location 0/RP0/CPU0 file
   /harddisk:/demo2
   ascii file is saved at: /harddisk:/demo2.txt
   pcap file is saved at: /harddisk:/demo2.pcap
   [OK]
   ```

   For complete command syntax, see the **tcp dump-file convert** command in the applicable system management command reference.

   The router creates a text file and a pcap file in the default or specified directory.

2. Display the converted text file in the CLI.

   Example

   ```
   Router# run cat
   /harddisk:/decoded_dumpfiles/text_tcpdump_peer_all_node0_RP0_CPU0_2024_3_19_10_8_53.462070.txt
   Filename: 2024_3_19_10_8_53.462070

   ============================================================
   Connection state is CLOSED, I/O status: 0, socket status: 103
   PCB 0x0000000000f47a80, SO 0xf476d0, TCPCB 0xf6a370, vrfid 0x60000000,
   Pak Prio: Medium, TOS: 192, TTL: 255, Hash index: 563
   Local host: 2001:DB8::1, Local port: 47743 (Local App PID: 19579)
   Foreign host: 2001:DB8::2, Foreign port: 179
   (Local App PID/instance/SPL_APP_ID: 19579/1/0)

   Current send queue size in bytes: 0 (max 0)
   Current receive queue size in bytes: 0 (max 0) mis-ordered: 0 bytes
   Current receive queue size in packets: 0 (max 0)
   ```

```
Timer          Starts   Wakeups        Next (msec)
Retrans            70         2                   0
SendWnd             0         0                   0
TimeWait            2         0                   0
AckHold            66        61                   0
KeepAlive           1         0                   0
PmtuAger            0         0                   0
GiveUp              0         0                   0
Throttle            0         0                   0
FirstSyn            1         1                   0

    iss: 3113104891 snduna: 3113106213 sndnxt: 3113106213
 sndmax: 3113106213 sndwnd: 31523      sndcwnd: 2832
    irs: 4250126727 rcvnxt: 4250128049 rcvwnd: 31448 rcvadv: 4250159497
```

The CLI displays the decoded packet trace. The sample shows only part of the available output.

3. Copy the converted packet traces from the router to a local computer with a remote file copy command such as **scp**.

   Open the copied pcap file in the packet-analysis tool used by your organization.

The stored packet traces are available in readable text and pcap formats for troubleshooting.

View a TCP dump file in text format manually

Displays the stored TCP dump file list and decodes one selected binary packet trace into text directly in the CLI.

Use this procedure to inspect one binary TCP packet trace without using the TCP dump file converter.

Manual viewing displays one binary file at a time. Use the converter when you must process multiple files.

Procedure

---

1. Display the binary packet trace files in the TCP dump directory.

   Example

```
Router# show tcp dump-file list all
total 1176
-rw-r--r-- 1 root root 5927 Nov 22 12:42 31_0_0_126.179.20966.cl.1700656933
-rw-r--r-- 1 root root 5892 Nov 22 12:42 31_0_0_127.179.35234.cl.1700656933
-rw-r--r-- 1 root root 6148 Nov 22 12:42 31_0_0_149.179.54939.cl.1700656933
-rw-r--r-- 1 root root 5894 Nov 22 12:42 31_0_0_155.179.18134.cl.1700656933
-rw-r--r-- 1 root root 6063 Nov 22 12:42 31_0_0_156.179.25445.cl.1700656933
-rw-r--r-- 1 root root 5860 Nov 22 12:42 31_0_0_161.179.30859.cl.1700656933
-rw-r--r-- 1 root root 5832 Nov 22 12:42 31_0_0_173.179.36935.cl.1700656933
-rw-r--r-- 1 root root 5906 Nov 22 12:42 31_0_0_190.179.25642.cl.1700656933
```

2. Display the selected binary file in text format.

   Example

```
Router# show tcp dump-file 10_106_0_73.179.34849.cl.1707424077 location
0/RP0/CPU0
Filename: 10_106_0_73.179.34849.cl.1707424077
```

```
================================================================
Connection state is CLOSED, I/O status: 0, socket status: 103
PCB 0x00007f86bc05e3b8, SO 0x7f86bc05e648, TCPCB 0x7f86bc0c3718, vrfid
0x60000000,
Pak Prio: Medium, TOS: 192, TTL: 1, Hash index: 1593
Local host: 10.106.0.72, Local port: 179 (Local App PID: 11354)
Foreign host: 10.106.0.73, Foreign port: 34849
(Local App PID/instance/SPL_APP_ID: 11354/1/0)

Current send queue size in bytes: 0 (max 0)
Current receive queue size in bytes: 0 (max 0) mis-ordered: 0 bytes
Current receive queue size in packets: 0 (max 0)

Timer          Starts   Wakeups        Next (msec)
Retrans        103448         8                   0
SendWnd             0         0                   0
TimeWait            1         0                   0
AckHold        106815    106545                   0
KeepAlive           1         0                   0
PmtuAger            0         0                   0
GiveUp              0         0                   0
Throttle            0         0                   0
FirstSyn            0         0                   0

    iss: 161240548  snduna: 163206936  sndnxt: 163206936
 sndmax: 163206936  sndwnd: 63104      sndcwnd: 18120
    irs: 3691232436 rcvnxt: 3693473072 rcvwnd: 26099 rcvadv: 3693499171
```

The CLI displays the decoded packet trace. The sample shows only part of the available output.

The selected binary TCP dump file is available as readable text in the CLI.