IP Addresses and Services Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.11.x

First Published: 2023-11-30

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2023 Cisco Systems, Inc. All rights reserved.

CONTENTS

PREFACE

Preface xi

Changes to This Document xi

Communications, Services, and Additional Information xi

CHAPTER 1

New and Changed IP Addresses and Services Features 1

IP Addresses and Services Features Added or Modified in IOS XR Release 7.11.x 1

CHAPTER 2

YANG Data Models for IP Addressing Features 3

Using YANG Data Models 3

CHAPTER 3

Implementing Network Stack IPv4 and IPv6 5

Implementing Network Stack IPv4 and IPv6 5

Network Stack IPv4 and IPv6 Exceptions 5

IPv4 and IPv6 Functionality 6

IPv6 for Cisco IOS XR Software 6

Prerequisites for Implementing Network Stack IPv4 and IPv6 6

How to Implement Network Stack IPv4 and IPv6 6

Configuring IPv4 Addressing 6

IPv4 Virtual Addresses 8

IPv4 ICMP Rate Limiting 9

Enabling IPv4 Processing on an Unnumbered Interface 10

Assigning Multiple IP Addresses to Network Interfaces 11

Configuring IPv6 Addressing 12

Larger IPv6 Address Space 13

IPv6 Address Formats 13

IPv6 Address Type: Unicast 14

Simplified IPv6 Packet Header 17
IPv6 Multicast Groups 20
IPv6 Virtual Addresses 24
IPv6 ICMP Rate Limiting 25
Configuring IPv4 and IPv6 Protocol Stacks 26
Selecting Flexible Source IP 27
Path MTU Discovery for IPv6 27
IPv6 Neighbor Discovery 28
IPv6 Neighbor Solicitation Message 28
IPv6 Router Advertisement Message 30
IPv6 Neighbor Redirect Message 31
Configuring IPARM Conflict Resolution 32
Static Policy Resolution 32
Longest Prefix Address Conflict Resolution 33
Highest IP Address Conflict Resolution 33
Route-Tag Support for Connected Routes 34
Address Repository Manager 35
Address Conflict Resolution 35
VRF Fallback and Redirect 35
Configure VRF Fallback and Redirect 37
Local Station MAC Address for Router 38

CHAPTER 4 **Configuring ARP** 41
Configuring ARP 41
Address Resolution Overview 41
ARP Cache Entries 43
Defining a Static ARP Cache Entry 43
Proxy ARP and Local Proxy ARP 44
Enabling Proxy ARP 44
Enabling Local Proxy ARP 45
Configuring ARP purge-delay 46
Configuring ARP Timeout 47
Limit ARP Cache Entries per Interface 48
Policing Duplicate ARP Packets 50

Direct Attached Gateway Redundancy 50
Configuring DAGR 51

CHAPTER 5 **Implementing DHCP** 53
Introduction to DHCP Relay 53
Prerequisites for Configuring DHCP Relay Agent 54
Limitations for DHCP Relay Feature 54
How to Configure and Enable DHCP Relay Agent 54
Configuring and Enabling the DHCP Relay Agent 54
Enabling a DHCP Relay Agent on an Interface 55
Disabling DHCP Relay on an Interface 56
Configuring and Enabling DHCP Relay Agent with DHCP MAC Address Verification 56
Configuring the DHCPv6 (Stateless) Relay Agent 57
Enabling DHCP Relay on a VRF 58
Configure a DHCP Relay Profile with Multiple Helper Addresses 58
DHCP Relay Agent Notification for Prefix Delegation 59
DHCPv6 Relay Over BVI for IANA Address Allocation 59
Configuring the Relay Agent Information Feature 62
Configuring Relay Agent Giaddr Policy 64
Implementing DHCP Snooping 64
Prerequisites for Configuring DHCP Snooping 64
Information about DHCP Snooping 65
Trusted and Untrusted Ports 66
DHCP Snooping in a Bridge Domain 66
Assigning Profiles to a Bridge Domain 66
Relay Information Options 66
How to Configure DHCP Snooping 66
Enabling DHCP Snooping in a Bridge Domain 66
Disabling DHCP Snooping on a Specific Bridge Port 69
Using the Relay Information Option 70

CHAPTER 6 **Implementing Host Services and Applications** 73
Implementing Host Services and Applications 73
Network Connectivity Tools 73

Ping 73
Checking Network Connectivity 74
Checking Network Connectivity for Multiple Destinations 75
Traceroute 76
Checking Packet Routes 77
Domain Services 78
Configuring Domain Services 78
File Transfer Services 79
FTP 79
Configuring a Router to Use FTP Connections 79
TFTP 80
Configuring a Router to Use TFTP Connections 80
SCP 81
Transferring Files Using SCP 81
Cisco inetd 82
Telnet 82
Syslog source-interface 83
HTTP Client Application 83

CHAPTER 7 **Implementing Access Lists** 85
Understanding Access Lists 85
IP Access List Entry Sequence Numbering 90
Sequence Numbering Behavior 90
Adding Entries with Sequence Numbers: Example 91
Adding Entries Without Sequence Numbers: Example 91
Applying Access Lists 92
Hybrid ACLs 92
Restrictions for egress hybrid ACLs 94
Configuring an Object-Group ACL 94
Configuring a Network Object-Group ACL 95
Configuring a Port Object-Group ACL 96
Verifying Object-Group ACL Compression 97
Configuring IPv4 ACLs 99
Configuring IPv6 ACLs 102

TCP Flags in ACLs 110
Configuring Extended Access Lists 113
IPv4 and IPv6 ACL in Class Map 114
Configuring IPv6 ACL QoS - An Example 114
User-Defined TCAM Keys for IPv4 and IPv6 115
IPv4 and IPv6 Key Formats 115
Modifying ACLs 117
ACLs on Bridge Virtual Interfaces 119
Configuring ACL-based Forwarding 121
Access Control List Counters 127
ACL Statistics Counter 127
Configuring ACLs with Fragment Control 129
Configuring an IPv4 ACL to Match on Fragment Type 131
Matching by Fragment Offset in ACLs 132
Configuring ACL Matching by Fragment Offset 132
Configuring ACL Filtering by IP Packet Length 133
Configuring Scaled IPv4 ACLs to Filter by Packet Length 134
Configuring Scaled IPv6 ACLs to Filter by Packet Length 135
Configuring TTL Matching 136
Understanding IP Access List Logging Messages 136
Enable Interface Logging on ACLs 137
Per Interface Statistics 140

CHAPTER 8 **Implementing Cisco Express Forwarding 143**
Implementing Cisco Express Forwarding 143
Prerequisites for Implementing Cisco Express Forwarding 144
Verifying CEF 144
Configuration Status of Cisco Express Forwarding Hardware Modules 146
Configuring Static Route 146
BGP Attributes Download 147
Proactive Address Resolution Protocol and Neighbor Discovery 148
Route Scale Improvements 149

CHAPTER 9 **Implementing LPTS 151**

LPTS Overview 151
LPTS Policers 151
LPTS and NPU Traps 154
Defining Dynamic LPTS Flow Type 156
User-managed control and management plane ACLs 158
Best practices for control and management plane ACLs 160
Restrictions for user-managed control and management plane ACLs 160
Configuring Control Plane and Management Plane Traffic 161
Configuring ACLs for AH and ESP Headers 162

CHAPTER 10 **Implement VRRP** 167
Implement VRRP 167
Generic Restrictions for VRRP Configuration 167
Understand VRRP 168
Unicast VRRP 172
VRRP over BVI 173
Configure VRRP over BVI 175
View VRRP statistics in the Router 179

CHAPTER 11 **Implement HSRP** 181
Implement HSRP 181
General Restrictions for HSRP Configuration 181
Information About Implementing HSRP 181
HSRP Overview 182
HSRP Groups 182
HSRP and ARP 184
Preemption 185
ICMP Redirect Messages 185
HSRP over BVI 185
Configure HSRP over BVI 188

CHAPTER 12 **Configuring Transports** 193
Information About Configuring NSR, TCP, UDP Transports 193
NSR Overview 193

Restrictions for NSR 193
TCP Overview 194
UDP Overview 194
Prerequisites for Configuring NSR, TCP, UDP, Transports 194
Configuring Failover as a Recovery Action for NSR 194

Preface

This preface contains these sections:

- Changes to This Document, on page xi
- Communications, Services, and Additional Information, on page xi

Changes to This Document

Table 1: Changes to This Document

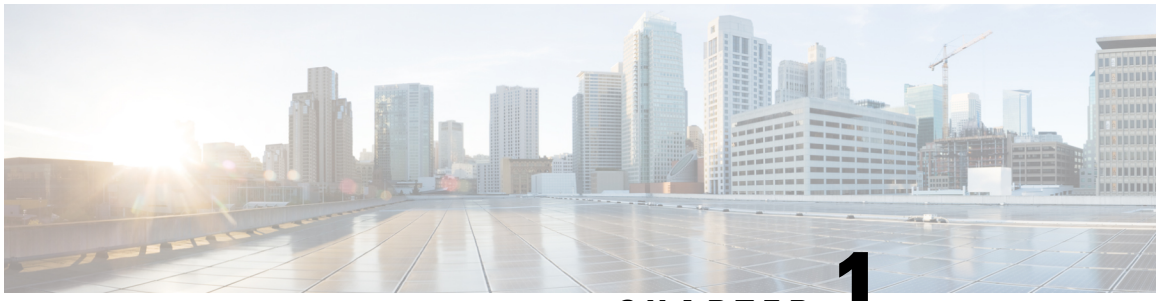
Date	Change Summary
December 2023	Initial release of this document

Communications, Services, and Additional Information

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

Cisco Bug Search Tool

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



CHAPTER 1

New and Changed IP Addresses and Services Features

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

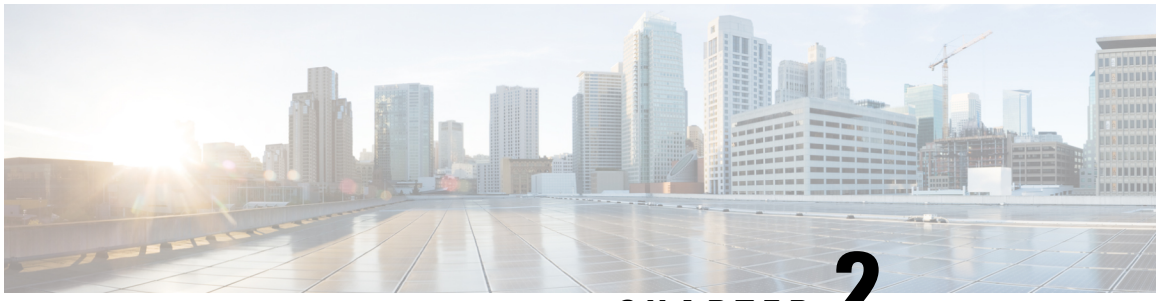
- IP Addresses and Services Features Added or Modified in IOS XR Release 7.11.x, on page 1

IP Addresses and Services Features Added or Modified in IOS XR Release 7.11.x

IP Addresses Features Added or Modified in IOS XR Release 7.11.x

Table 2: New and Changed Features

Feature	Description	Changed in Release	Where Documented
Unicast VRRP	This feature was introduced.	Release 7.11.1	Unicast VRRP, on page 172



CHAPTER 2

YANG Data Models for IP Addressing Features

This chapter provides information about the YANG data models for IP Addressing features.

• Using YANG Data Models, on page 3

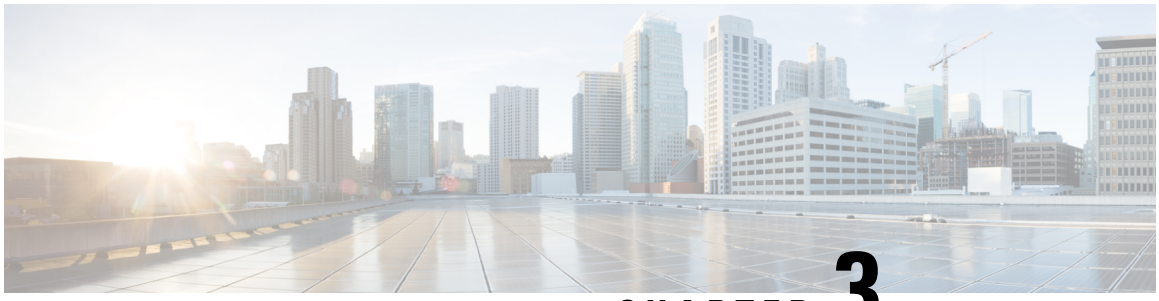
Using YANG Data Models

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

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

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

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



CHAPTER 3

Implementing Network Stack IPv4 and IPv6

- Implementing Network Stack IPv4 and IPv6, on page 5
- Network Stack IPv4 and IPv6 Exceptions, on page 5
- IPv4 and IPv6 Functionality, on page 6
- IPv6 for Cisco IOS XR Software, on page 6
- Prerequisites for Implementing Network Stack IPv4 and IPv6, on page 6
- How to Implement Network Stack IPv4 and IPv6, on page 6

Implementing Network Stack IPv4 and IPv6

The Network Stack IPv4 and IPv6 features are used to configure and monitor Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6).

Restrictions

In any Cisco IOS XR software release with IPv6 support, multiple IPv6 global addresses can be configured on an interface. However, multiple IPv6 link-local addresses on an interface are not supported.

Network Stack IPv4 and IPv6 Exceptions

The Network Stack feature in the Cisco IOS XR software has the following exceptions:

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

IPv4 and IPv6 Functionality

When Cisco IOS XR software is configured with both an IPv4 and an IPv6 address, the interface can send and receive data on both IPv4 and IPv6 networks.

The architecture of IPv6 has been designed to allow existing IPv4 users to make the transition easily to IPv6 while providing services such as end-to-end security, quality of service (QoS), and globally unique addresses. The larger IPv6 address space allows networks to scale and provide global reachability. The simplified IPv6 packet header format handles packets more efficiently. IPv6 prefix aggregation, simplified network renumbering, and IPv6 site multihoming capabilities provide an IPv6 addressing hierarchy that allows for more efficient routing. IPv6 supports widely deployed routing protocols such as Open Shortest Path First (OSPF), and multiprotocol Border Gateway Protocol (BGP).

The IPv6 neighbor discovery (nd) 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 network (local link), verify the reachability of a neighbor, and keep track of neighboring routers.

IPv6 for Cisco IOS XR Software

IPv6, formerly named IPng (next generation) is the latest version of the Internet Protocol (IP). IP is a packet-based protocol used to exchange data, voice, and video traffic over digital networks. IPv6 was proposed when it became clear that the 32-bit addressing scheme of IP version 4 (IPv4) was inadequate to meet the demands of Internet growth. After extensive discussion, it was decided to base IPng on IP but add a much larger address space and improvements such as a simplified main header and extension headers. IPv6 is described initially in RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification* issued by the Internet Engineering Task Force (IETF). Further RFCs describe the architecture and services supported by IPv6.

Prerequisites for Implementing Network Stack IPv4 and IPv6

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

How to Implement Network Stack IPv4 and IPv6

This section contains the following procedures:

Configuring IPv4 Addressing

A basic and required task for configuring IP is to assign IPv4 addresses to network interfaces. Doing so enables the interfaces and allows communication with hosts on those interfaces using IPv4. An IP address identifies a location to which IP datagrams can be sent. An interface can have one primary IP address and multiple secondary addresses. Packets generated by the software always use the primary IPv4 address. Therefore, all networking devices on a segment should share the same primary network number.

Note Starting with Cisco IOS XR Release 7.10.2, the CLI no longer accepts IPv4 addresses with leading zeros. Use canonical dotted-decimal IPv4 addresses such as 198.51.100.1.

If your existing configuration, scripts, or Route Policy Language (RPL) expressions contain IPv4 addresses with leading zeros, NETCONF and YANG reject the address. This can cause configuration retrieval, synchronization, or BGP-related configuration to fail. Therefore, configure all IPv4 addresses in the correct format, ensuring that no part of the address includes leading zeros.

Associated with this task are decisions about subnetting and masking the IP addresses. A mask identifies the bits that denote the network number in an IP address. When you use the mask to subnet a network, the mask is then referred to as a *subnet mask*.

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

Configuration Example

An IPv4 address of 192.0.2.27 and a network mask of "/24" is assigned to the interface HundredGigE 0/0/0/24.

Note The network mask can be a four-part dotted decimal address. For example, 255.255.255.0 indicates that each bit equal to 1 means the corresponding address bit belongs to the network address. The network mask can be indicated as a slash (/) and a number- a prefix length. The prefix length is a decimal value that indicates how many of the high-order contiguous bits of the address comprise the prefix (the network portion of the address). A slash must precede the decimal value, and there is no space between the IP address and the slash.

```
Router#configure
Router(config)#interface HundredGigE 0/0/0/24
Router(config-if)#ipv4 address 192.0.2.27/24
Router(config-if)#no shutdown
Router(config-if)#commit
```

Running Configuration

```
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
!
```

Verification

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

```
Router# show ipv4 interface HundredGigE 0/0/0/24
Fri Sep 27 09:36:36.244 UTC
HundredGigE0/0/0/24 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.0.2.27/24
  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
```

IPv4 Virtual Addresses

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

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

The **use-as-src-addr** keyword eliminates the need for configuring a loopback interface as the source interface (that is, update source) for management applications. When an update source is not configured, management applications allow the transport processes (TCP, UDP, raw_ip) to select a suitable source address. The transport processes, in turn, consult the FIB for selecting a suitable source address. If a Management Ethernet's IP address is selected as the source address and if the **use-as-src-addr** keyword is configured, then the transport substitutes the Management Ethernet's IP address with a relevant virtual IP address. This functionality works across RP switchovers. If the **use-as-src-addr** is not configured, then the source-address selected by transports can change after a failover and the NMS software may not be able to manage this situation.

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

Configuration Example

The following example shows how to define an IPv4 virtual address:

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

The following example shows how to define an IPv4 virtual address on a management interface:

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

Router#show running-config interface mgmtEth 0/RP0/CPU0/0
Fri Sep 27 09:55:08.173 UTC
interface MgmtEth0/RP0/CPU0/0
 ipv4 address 192.0.2.28 255.255.255.0
!
```

The following example show how to configure the virtual IP addresses for management interfaces on a per VRF basis:

```
Router#configure terminal
Router(config)#interface mgmtEth 0/RP0/CPU0/0
Router(config)#VRF test
Router(config-if)#ipv4 address 192.0.2.29 2555.255.255.0
Router(config-if)#no shutdown
Router(config-if)#commit
```

```
Router#show running-config interface mgmtEth 0/RP0/CPU0/0
Fri Sep 27 10:32:45.932 UTC
interface MgmtEth0/RP0/CPU0/0
 vrf test
 ipv4 address 192.0.2.29 255.255.255.0
!
```

IPv4 ICMP Rate Limiting

The IPv4 ICMP rate limiting feature limits the rate that IPv4 ICMP destination unreachable messages are generated. The Cisco IOS XR software maintains two timers: one for general destination unreachable messages and one for DF destination unreachable messages. Both share the same time limits and defaults. If the DF keyword is not configured, the icmp ipv4 rate-limit unreachable command sets the time values for DF destination unreachable messages. If the DF keyword is configured, its time values remain independent from those of general destination unreachable messages.

Configuration Example

Limits the rate that IPv4 ICMP destination unreachable messages are generated every 1000 millisecond.

The **DF** keyword, which is optional limits the rate at which ICMP destination unreachable messages are sent when code 4 fragmentation is needed and Don't Fragment (DF) is set, as specified in the IP header of the ICMP destination unreachable message.

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

Running Configuration

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

Verification

```
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 192.85.1.1/24
  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
```

The number of ICMP unreachable messages that were we sent or received can be identified using the **show ipv4 traffic** command.

```
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
```

Enabling IPv4 Processing on an Unnumbered Interface

This section describes the process of enabling an IPv4 point-to-point interface without assigning an explicit IP address to the interface. Whenever the unnumbered interface generates a packet (for example, for a routing update), it uses the address of the interface you specified as the source address of the IP packet. It also uses the specified interface address in determining which routing processes are sending updates over the unnumbered interface. Restrictions are as follows:

- You cannot use the **ping** EXEC command to determine whether the interface is up, because the interface has no IP address. The Simple Network Management Protocol (SNMP) can be used to remotely monitor interface status.
- You cannot support IP security options on an unnumbered interface.

Configuration Example

Enables an IPv4 point-to-point interface without assigning an explicit IP address to the interface.

```
Router#configure
Router(config)#interface HundredGigE 0/0/0/25
Router(config-if)#ipv4 point-to-point
Router(config-if)#commit
Router(config-if)#ipv4 unnumbered loopback 0
Router(config-if)#commit
```

Running Configuration

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

Verification

```
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
  output flow control is off, input flow control is off
  Carrier delay (up) is 10 msec
  loopback not set,
  Last link flapped 01:38:49
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters 02:34:16
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 7647051000 bits/sec, 12254894 packets/sec
     1061401410 packets input, 82789675614 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 5 broadcast packets, 19429 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     76895885948 packets output, 6192569128048 bytes, 0 total output drops
     Output 7 broadcast packets, 18916 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     2 carrier transitions

Router #show runnning-config interface loopback 0
interface Loopback0
 ipv4 address 192.0.2.1 255.255.255.255
```

Assigning Multiple IP Addresses to Network Interfaces

The Cisco IOS XR software supports multiple IP addresses (secondary addresses) per interface. You can specify an unlimited number of secondary addresses. Secondary IP addresses can be used in a variety of situations. The following are the most common applications:

- There might not be enough host addresses for a particular network segment. For example, suppose your subnetting allows up to 254 hosts per logical subnet, but on one physical subnet you must have 300 host addresses. Using secondary IP addresses on the routers or access servers allows you to have two logical subnets using one physical subnet.
- Many older networks were built using Level 2 bridges, and were not subnetted. The judicious use of secondary addresses can aid in the transition to a subnetted, router-based network. Routers on an older, bridged segment can easily be made aware that many subnets are on that segment.

- Two subnets of a single network might otherwise be separated by another network. You can create a single network from subnets that are physically separated by another network by using a secondary address. In these instances, the first network is *extended*, or layered on top of the second network. Note that a subnet cannot appear on more than one active interface of the router at a time.

Note 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.

Caution Inconsistent use of secondary addresses on a network segment can quickly cause routing loops.

Configuration Example

A secondary IPv4 address of 192.0.2.1 is assigned to the Hundredgige interface-0/0/0/24.

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

```
Router#configure
Router(config)#interface HundredGigE 0/0/0/24
Router(config-if)#ipv4 address 192.0.2.1 255.255.255.0 secondary
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE 0/0/0/24
Mon Sep 30 09:37:12.096 UTC
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
!
```

Verification

```
Router#show ipv4 interface HundredGigE 0/0/0/24
Mon Sep 30 09:38:18.262 UTC
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
```

Configuring IPv6 Addressing

IPv6 addresses are configured to individual router interfaces in order to enable the forwarding of IPv6 traffic globally on the router. By default, IPv6 addresses are not configured.

Note The *ipv6-prefix* argument in the **ipv6 address** command must be in the form documented in RFC 2373 in which the address is specified in hexadecimal using 16-bit values between colons.

The **/prefix-length** argument in the **ipv6 address** command is a decimal value that indicates how many of the high-order contiguous bits of the address comprise the prefix (the network portion of the address) A slash must precede the decimal value.

The *ipv6-address* argument in the **ipv6 address link-local** command must be in the form documented in RFC 2373 where the address is specified in hexadecimal using 16-bit values between colons.

Larger IPv6 Address Space

The primary motivation for IPv6 is the need to meet the anticipated future demand for globally unique IP addresses. Applications such as mobile Internet-enabled devices (such as personal digital assistants [PDAs], telephones, and cars), home-area networks (HANs), and wireless data services are driving the demand for globally unique IP addresses. IPv6 quadruples the number of network address bits from 32 bits (in IPv4) to 128 bits, which provides more than enough globally unique IP addresses for every networked device on the planet. By being globally unique, IPv6 addresses inherently enable global reachability and end-to-end security for networked devices, functionality that is crucial to the applications and services that are driving the demand for the addresses. Additionally, the flexibility of the IPv6 address space reduces the need for private addresses and the use of Network Address Translation (NAT); therefore, IPv6 enables new application protocols that do not require special processing by border routers at the edge of networks.

IPv6 Address Formats

IPv6 addresses are represented as a series of 16-bit hexadecimal fields separated by colons (:) in the format: x:x:x:x:x:x:x:x. Following are two examples of IPv6 addresses:

2001:0DB8:7654:3210:FEDC:BA98:7654:3210

2001:0DB8:0:0:8:800:200C:417A

It is common for IPv6 addresses to contain successive hexadecimal fields of zeros. To make IPv6 addresses less cumbersome, two colons (::) can be used to compress successive hexadecimal fields of zeros at the beginning, middle, or end of an IPv6 address. (The colons represent successive hexadecimal fields of zeros.) Table 3: Compressed IPv6 Address Formats, on page 14 lists compressed IPv6 address formats.

A double colon may be used as part of the *ipv6-address* argument when consecutive 16-bit values are denoted as zero. You can configure multiple IPv6 addresses per interfaces, but only one link-local address.

Note Two colons (::) can be used only once in an IPv6 address to represent the longest successive hexadecimal fields of zeros.

The hexadecimal letters in IPv6 addresses are not case-sensitive.

Table 3: Compressed IPv6 Address Formats

IPv6 Address Type	Preferred Format	Compressed Format
Unicast	2001:0:0:0:0DB8:800:200C:417A	1080::0DB8:800:200C: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 loopback address listed in Table 3: Compressed IPv6 Address Formats, on page 14 may be used by a node to send an IPv6 packet to itself. The loopback address in IPv6 functions the same as the loopback address in IPv4 (127.0.0.1).

Note The IPv6 loopback address cannot be assigned to a physical interface. A packet that has the IPv6 loopback address as its source or destination address must remain within the node that created the packet. IPv6 routers do not forward packets that have the IPv6 loopback address as their source or destination address.

The unspecified address listed in Table 3: Compressed IPv6 Address Formats, on page 14 indicates the absence of an IPv6 address. For example, a newly initialized node on an IPv6 network may use the unspecified address as the source address in its packets until it receives its IPv6 address.

Note The IPv6 unspecified address cannot be assigned to an interface. The unspecified IPv6 addresses must not be used as destination addresses in IPv6 packets or the IPv6 routing header.

An IPv6 address prefix, in the format *ipv6-prefix/prefix-length* , can be used to represent bit-wise contiguous blocks of the entire address space. The *ipv6-prefix* argument must be in the form documented in RFC 2373, in which the address is specified in hexadecimal using 16-bit values between colons. The prefix length is a decimal value that indicates how many of the high-order contiguous bits of the address compose the prefix (the network portion of the address). For example, 2001:0DB8:8086:6502::/32 is a valid IPv6 prefix.

IPv6 Address Type: Unicast

An IPv6 unicast address is an identifier for a single interface, on a single node. A packet that is sent to a unicast address is delivered to the interface identified by that address. Cisco IOS XR software supports the following IPv6 unicast address types:

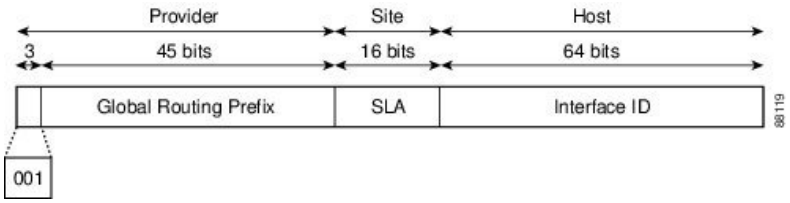
- Global aggregatable address
- Site-local address (proposal to remove by IETF)
- Link-local address
- IPv4-compatible IPv6 address

Aggregatable Global Address

An aggregatable global address is an IPv6 address from the aggregatable global unicast prefix. The structure of aggregatable global unicast addresses enables strict aggregation of routing prefixes that 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 the Internet service providers (ISPs).

Aggregatable global IPv6 addresses are defined by a global routing prefix, a subnet ID, and an 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 range of addresses that start with binary value 001 (2000::/3). This figure below shows the structure of an aggregatable global address.

Figure 1: Aggregatable Global Address Format



Addresses with a prefix of 2000::/3 (001) through E000::/3 (111) are required to have 64-bit interface identifiers in the extended universal identifier (EUI)-64 format. The Internet Assigned Numbers Authority (IANA) allocates the IPv6 address space in the range of 2000::/16 to regional registries.

The aggregatable global address typically consists of a 48-bit global routing prefix and a 16-bit subnet ID or Site-Level Aggregator (SLA). In the IPv6 aggregatable global unicast address format document (RFC 2374), the global routing prefix included two other hierarchically structured fields named Top-Level Aggregator (TLA) and Next-Level Aggregator (NLA).The IETF decided to remove the TLS and NLA fields from the RFCs, because these fields are policy-based. Some existing IPv6 networks deployed before the change might still be using networks based on the older architecture.

A 16-bit subnet field called the subnet ID could be used by individual organizations to create their own local addressing hierarchy and to 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.

An interface ID is used to identify interfaces on a link. The interface ID must be unique to the link. It may 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.

Interface IDs are constructed in the modified EUI-64 format in one of the following ways:

- For all IEEE 802 interface types (for example, Ethernet interfaces and FDDI interfaces), the first three octets (24 bits) are taken from the Organizationally Unique Identifier (OUI) of the 48-bit link-layer address (MAC address) of the interface, the fourth and fifth octets (16 bits) are a fixed hexadecimal value of FFFE, and the last three octets (24 bits) are taken from the last three octets of the MAC address. The construction of the interface ID is completed by setting the Universal/Local (U/L) bit—the seventh bit of the first octet—to a value of 0 or 1. A 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.

Note For interfaces using Point-to-Point Protocol (PPP), given that the interfaces at both ends of the connection might have the same MAC address, the interface identifiers used at both ends of the connection are negotiated (picked randomly and, if necessary, reconstructed) until both identifiers are unique. The first MAC address in the router is used to construct the identifier for interfaces using PPP.

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

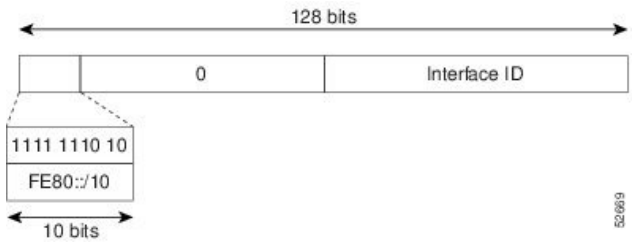
1. The router is queried for MAC addresses (from the pool of MAC addresses in the router).
2. If no MAC address is available, the serial number of the Route Processor (RP) or line card (LC) is used to form the link-local address.

Link-Local Address

A link-local address is an IPv6 unicast address that can be automatically configured on any interface using the link-local prefix FE80::/10 (1111 1110 10) and the interface identifier in the modified EUI-64 format. Link-local addresses are used in the neighbor discovery protocol and the stateless autoconfiguration process. Nodes on a local link can use link-local addresses to communicate; the nodes do not need site-local or globally unique addresses to communicate. This figure below shows the structure of a link-local address.

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

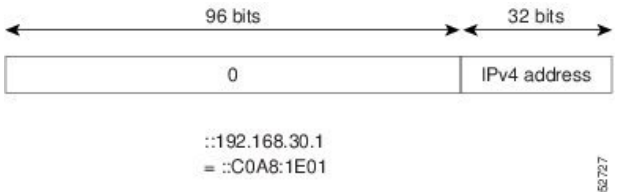
Figure 2: Link-Local Address Format



IPv4-Compatible IPv6 Address

An IPv4-compatible IPv6 address is an IPv6 unicast address that has zeros in the high-order 96 bits of the address and an IPv4 address in the low-order 32 bits of the address. 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 and the IPv4 address embedded in the low-order 32 bits is used as the IPv4 address of the node. IPv4-compatible IPv6 addresses are assigned to nodes that support both the IPv4 and IPv6 protocol stacks and are used in automatic tunnels. This figure below shows the structure of an IPv4-compatible IPv6 address and a few acceptable formats for the address.

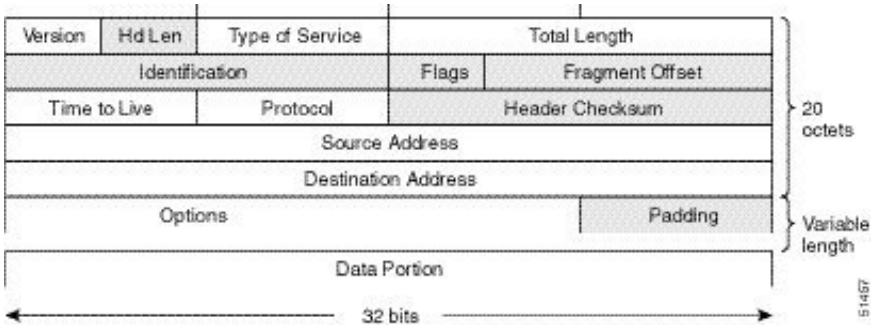
Figure 3: IPv4-Compatible IPv6 Address Format



Simplified IPv6 Packet Header

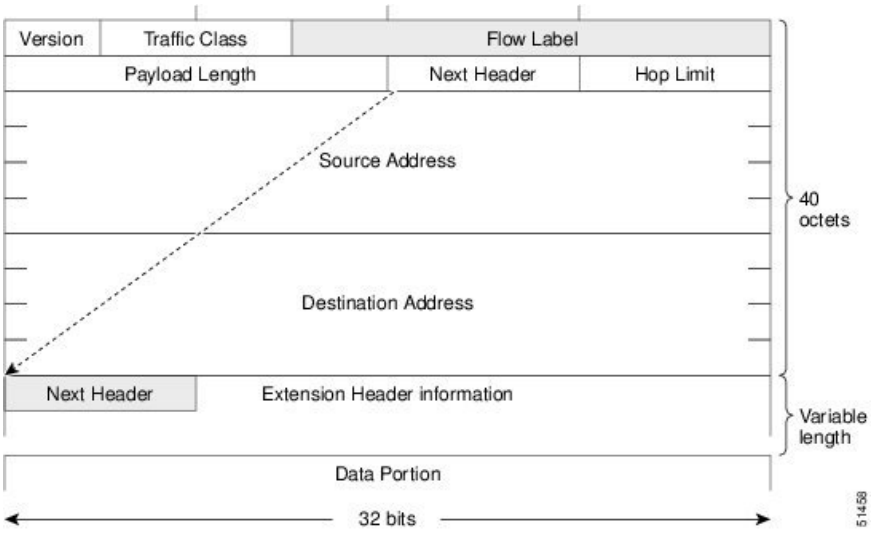
The basic IPv4 packet header has 12 fields with a total size of 20 octets (160 bits). The 12 fields may be followed by an Options field, which is followed by a data portion that is usually the transport-layer packet. The variable length of the Options field adds to the total size of the IPv4 packet header. The shaded fields of the IPv4 packet header are not included in the IPv6 packet header.

Figure 4: IPv4 Packet Header Format



The basic IPv6 packet header has 8 fields with a total size of 40 octets (320 bits). Fields were removed from the IPv6 header because, in IPv6, fragmentation is not handled by routers and checksums at the network layer are not used. Instead, fragmentation in IPv6 is handled by the source of a packet and checksums at the data link layer and transport layer are used. (In IPv4, the User Datagram Protocol (UDP) transport layer uses an optional checksum. In IPv6, use of the UDP checksum is required to check the integrity of the inner packet.) Additionally, the basic IPv6 packet header and Options field are aligned to 64 bits, which can facilitate the processing of IPv6 packets.

Figure 5: IPv6 Packet Header Format



This table lists the fields in the basic IPv6 packet header.

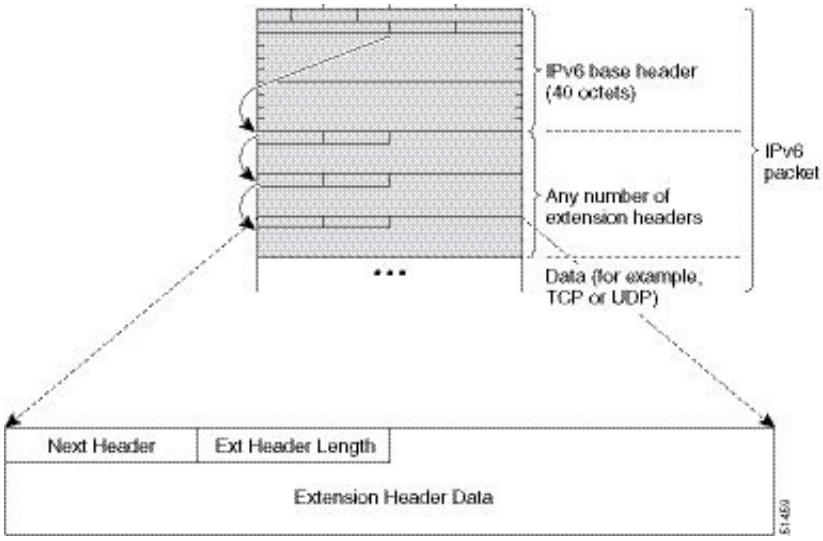
Table 4: Basic IPv6 Packet Header Fields

Field	Description
Version	Similar to the Version field in the IPv4 packet header, except that the field lists number 6 for IPv6 instead of number 4 for IPv4.
Traffic Class	Similar to the Type of Service field in the IPv4 packet header. The Traffic Class field tags packets with a traffic class that is used in differentiated services.
Flow Label	A new field in the IPv6 packet header. The Flow Label field tags packets with a specific flow that differentiates the packets at the network layer.
Payload Length	Similar to the Total Length field in the IPv4 packet header. The Payload Length field indicates the total length of the data portion of the packet.
Next Header	Similar to the Protocol field in the IPv4 packet header. The value of the Next Header field determines the type of information following the basic IPv6 header. The type of information following the basic IPv6 header can be a transport-layer packet, for example, a TCP or UDP packet, or an Extension Header.
Hop Limit	Similar to the Time to Live field in the IPv4 packet header. The value of the Hop Limit field specifies the maximum number of routers that an IPv6 packet can pass through before the packet is considered invalid. Each router decrements the value by one. Because no checksum is in the IPv6 header, the router can decrement the value without needing to recalculate the checksum, which saves processing resources.
Source Address	Similar to the Source Address field in the IPv4 packet header, except that the field contains a 128-bit source address for IPv6 instead of a 32-bit source address for IPv4.

Field	Description
Destination Address	Similar to the Destination Address field in the IPv4 packet header, except that the field contains a 128-bit destination address for IPv6 instead of a 32-bit destination address for IPv4.

Following the eight fields of the basic IPv6 packet header are optional extension headers and the data portion of the packet. If present, each extension header is aligned to 64 bits. There is no fixed number of extension headers in an IPv6 packet. Together, the extension headers form a chain of headers. Each extension header is identified by the Next Header field of the previous header. Typically, the final extension header has a Next Header field of a transport-layer protocol, such as TCP or UDP. This figure below shows the IPv6 extension header format.

Figure 6: IPv6 Extension Header Format



This table lists the extension header types and their Next Header field values.

Table 5: IPv6 Extension Header Types

Header Type	Next Header Value	Description
Hop-by-hop options header	0	This header is processed by all hops in the path of a packet. When present, the hop-by-hop options header always follows immediately after the basic IPv6 packet header.
Destination options header	60	The destination options header can follow any hop-by-hop options header, in which case the destination options header is processed at the final destination and also at each visited address specified by a routing header. Alternatively, the destination options header can follow any Encapsulating Security Payload (ESP) header, in which case the destination options header is processed only at the final destination.
Routing header	43	The routing header is used for source routing.

Header Type	Next Header Value	Description
Fragment header	44	The fragment header is used when a source must fragment a packet that is larger than the maximum transmission unit (MTU) for the path between itself and a destination. The Fragment header is used in each fragmented packet.
Authentication header and ESP header	51 50	The Authentication header and the ESP header are used within IP Security Protocol (IPSec) to provide authentication, integrity, and confidentiality of a packet. These headers are identical for both IPv4 and IPv6.
Upper-layer header	6 (TCP) 17 (UDP)	The upper-layer (transport) headers are the typical headers used inside a packet to transport the data. The two main transport protocols are TCP and UDP.
Mobility header	To be done by IANA	Extension headers used by mobile nodes, correspondent nodes, and home agents in all messaging related to the creation and management of bindings.

IPv6 Multicast Groups

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

Additionally, the configured interface automatically joins the following required multicast groups for that 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

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

Configuration Example

An IPv6 address of 2001:0DB8:0:1::1/64 is assigned to the HundredGigE interface 0/0/0/25:

```
Router# configure
Router(config)# interface HundredGigE 0/0/0/25
Router(config-if)# ipv6 address 2001:0DB8:0:1::1/64
Router(config-if)# no shutdown
Router(config-if)#commit
```

Running Configuration

```
Router# show running-config interface HundredGigE 0/0/0/25
 interface HundredGigE0/0/0/25
```

```
 ipv6 address 2001:db8:0:1::1/64
!
```

Verification

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

```
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
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 160 to 240 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Configuration Example Using eui-64Keyword

An IPv6 address of 2001:0DB8:0:1::/64 is assigned to the interface HundredGigE 0/0/0/35. The **eui-64** keyword configures site-local and global IPv6 addresses with an interface identifier (ID) in the low-order 64 bits of the IPv6 address. Only the 64-bit network prefix for the address needs to be specified; the last 64 bits are automatically computed from the interface ID.

```
Router# configure
Router(config)# interface HundredGigE 0/0/0/35
Router(config-if)# ipv6 address 2001:0DB8:0:1::/64 eui-64
Router(config-if)#no shutdown
Router(config-if)#commit
```

Running Configuration

```
Router# show running-config interface HundredGigE 0/0/0/35
Mon Sep 30 09:05:07.715 UTC
interface HundredGigE0/0/0/35
 ipv6 address 2001:db8:0:1::/64 eui-64
!
```

Verification

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

```
Router#show ipv6 interface HundredGigE 0/0/0/35
```

```
Mon Sep 30 09:06:12.638 UTC
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
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 160 to 240 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Configuration Example Using link-local Address

An IPv6 address of FE80::260:3EFF:FE11:6770 is assigned to the interface HundredGigE 0/0/0/34. The link-local keyword configures a link-local address on the interface that is used instead of the link-local address that is automatically configured when IPv6 is enabled on the interface.

```
Router#configure
Router(config)#interface HundredGigE 0/0/0/34
Router(config-if)#ipv6 address FE80::260:3EFF:FE11:6770 link-local
Router(config-if)#no shutdown
Router(config-if)#commit
```

Running Configuration

```
Router# show running-config interface HundredGigE 0/0/0/34
Mon Sep 30 09:11:09.742 UTC
interface HundredGigE0/0/0/34
 ipv6 address fe80::260:3eff:fe11:6770 link-local
!
```

Verification

Verify that the HundredGigE interface is active and IPv6 is enabled with link-local address.

```
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
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
```

```
  ND router advertisements are sent every 160 to 240 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Enable IPv6

Enable IPv6 processing on the interface HundredGigE 0/0/0/33; that has not been configured with an explicit IPv6 address.

```
Router#configure
Router(config)#interface HundredGigE 0/0/0/33
Router(config-if)#ipv6 enable
Router(config-if)#no shutdown
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE 0/0/0/33
Mon Sep 30 09:14:03.623 UTC
interface HundredGigE0/0/0/33
 ipv6 enable
!
```

Verification

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

```
Router#show ipv6 interface HundredGigE 0/0/0/33
Mon Sep 30 09:14:48.430 UTC
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
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 160 to 240 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

IPv6 Virtual Addresses

Configuring an IPv6 virtual address enables you to access the router from a single virtual address with a management network, without the prior knowledge of which route processor (RP) is active. An IPv6 virtual address persists across RP failover situations. For this to happen, the virtual IPv6 address must share a common IPv6 subnet with a Management Ethernet interface on both RPs.

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

The **use-as-src-addr** keyword eliminates the need for configuring a loopback interface as the source interface (that is, update source) for management applications. When an update source is not configured, management applications allow the transport processes (TCP, UDP, raw_ip) to select a suitable source address. The transport processes, in turn, consult the FIB for selecting a suitable source address. If a Management Ethernet's IP address is selected as the source address and if the **use-as-src-addr** keyword is configured, then the transport substitutes the Management Ethernet's IP address with a relevant virtual IP address. This functionality works across RP switchovers. If the **use-as-src-addr** is not configured, then the source-address selected by transports can change after a failover and the NMS software may not be able to manage this situation.

Note Protocol configuration such as tacacs source-interface, snmp-server trap-source, ntp source, logging source-interface do not use the virtual management IP address as their source by default. Use the **ipv6 virtual address use-as-src-addr** command to ensure that the protocol uses the virtual IPv6 address as its source address. Alternatively, you can also configure a loopback address with the designated or desired IPv6 address and set that as the source for protocols such as TACACS+ via the **tacacs source-interface** command.

Configuration Example

The following example shows how to define an IPv6 virtual address:

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

The following example shows how to define an IPv6 virtual address on a management interface:

```
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

Router#show running-config interface mgmtEth 0/RP0/CPU0/0
Mon Sep 30 09:18:50.468 UTC
interface MgmtEth0/RP0/CPU0/0
  ipv6 address 2001:db8:0:1::1/64
!
```

The following example show how to configure the virtual IP addresses for management interfaces on a per VRF basis:

```
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
```

```
Router#show running-config interface mgmtEth 0/RP0/CPU0/0
Mon Sep 30 09:24:19.111 UTC
interface MgmtEth0/RP0/CPU0/0
 ipv6 address 2001:db8:0:1::1/64
!
```

IPv6 ICMP Rate Limiting

The IPv6 ICMP rate limiting feature implements a token bucket algorithm for limiting the rate at which IPv6 ICMP error messages are sent out on the network. The initial implementation of IPv6 ICMP rate limiting defined a fixed interval between error messages, but some applications, such as traceroute, often require replies to a group of requests sent in rapid succession. The fixed interval between error messages is not flexible enough to work with applications such as traceroute and can cause the application to fail. Implementing a token bucket scheme allows a number of tokens-representing the ability to send one error message each-to be stored in a virtual bucket. The maximum number of tokens allowed in the bucket can be specified, and for every error message to be sent, one token is removed from the bucket. If a series of error messages is generated, error messages can be sent until the bucket is empty. When the bucket is empty of tokens, IPv6 ICMP error messages are not sent until a new token is placed in the bucket. The token bucket algorithm does not increase the average rate limiting time interval, and it is more flexible than the fixed time interval scheme.

Configuration Example

Configure the interval for 50 milliseconds and the bucket size for 20 tokens, for IPv6 ICMP error messages.

- 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.

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

Running Configuration

```
Router#show running-config
Building configuration...
!! IOS XR Configuration version = 6.0.0.26I
!! Last configuration change at Mon Dec 14 22:07:35 2015 by root
!
hostname test-83
logging console debugging
username root
 group root-lr
 group cisco-support
 secret 5 $1$d2NC$RbAdqdU7kw/kEJoMP/IJG1
!
cdp
ipv6 icmp error-interval 50 20
icmp ipv4 rate-limit unreachable DF 1000
icmp ipv4 rate-limit unreachable 1000
ipv4 conflict-policy static
```

Associated Commands

- ipv6 icmp error-interval

Configuring IPv4 and IPv6 Protocol Stacks

This task configures an interface in a Cisco networking device to support both the IPv4 and IPv6 protocol stacks.

When an interface in a Cisco networking device is configured with both an IPv4 and an IPv6 address, the interface forwards both IPv4 and IPv6 traffic—the interface can send and receive data on both IPv4 and IPv6 networks.

Configuration Example

An IPv4 address of 192.0.2.1 and an IPv6 address of 2001:0DB8:c18:1::3/64 is configured on the interface HundredGigE 0/0/0/31.

```
Router#configure
Router(config)#interface HundredGigE 0/0/0/31
Router(config-if)#ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)#ipv6 address 2001:0DB8:c18:1::3/64\
Router(config-if)#no shutdown
Router(config-if)#commit
```

Running Configuration

```
Router# show running-config interface HundredGigE 0/0/0/31
Mon Sep 30 09:45:48.379 UTC
interface HundredGigE0/0/0/31
 ipv4 address 192.0.2.1 255.255.255.0
 ipv6 address 2001:db8:c18:1::3/64
!
```

Verification

Verify that the HundredGigE interface is active and IPv4 and IPv6 are enabled.

```
Router#show ipv4 interface HundredGigE 0/0/0/31
Mon Sep 30 09:49:10.383 UTC
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
  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

Router#show ipv6 interface HundredGigE 0/0/0/31
 Mon Sep 30 09:51:01.472 UTC
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
ND reachable time is 0 milliseconds
ND cache entry limit is 1000000000
ND advertised retransmit interval is 0 milliseconds
ND router advertisements are sent every 160 to 240 seconds
ND router advertisements live for 1800 seconds
Hosts use stateless autoconfig for addresses.
Outgoing access list is not set
Inbound  common access list is not set, access list is not set
Table Id is 0xe0800000
Complete protocol adjacency: 0
Complete glean adjacency: 0
Incomplete protocol adjacency: 0
Incomplete glean adjacency: 0
Dropped protocol request: 0
Dropped glean request: 0
```

Selecting Flexible Source IP

You can select flexible source IP address in the Internet Control Message Protocol (ICMP) response packet to respond to a failure.

Configuration Example

Enables RFC compliance for source address selection.

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

Running Configuration

```
Router#show running-config | in source rfc
Building configuration...
icmp ipv4 source rfc
```

Associated Commands

- icmp ipv4 source vrf

Path MTU Discovery for IPv6

As in IPv4, path MTU discovery in IPv6 allows a host to dynamically discover and adjust to differences in the MTU size of every link along a given data path. In IPv6, however, fragmentation is handled by the source of a packet when the path MTU of one link along a given data path is not large enough to accommodate the size of the packets. Having IPv6 hosts handle packet fragmentation saves IPv6 router processing resources and helps IPv6 networks run more efficiently.

In IPv4, the minimum link MTU is 68 octets, which means that the MTU size of every link along a given data path must support an MTU size of at least 68 octets. In IPv6, the minimum link MTU is 1280 octets. We recommend using an MTU value of 1500 octets for IPv6 links.

Note Path MTU discovery is supported only for applications using TCP.

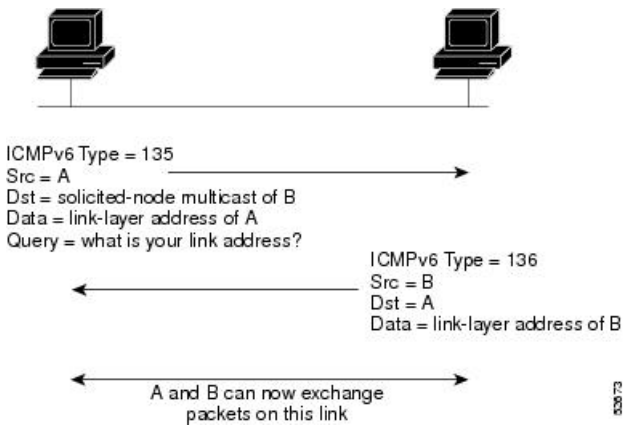
IPv6 Neighbor Discovery

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

IPv6 Neighbor Solicitation Message

A value of 135 in the Type field of the ICMP packet header identifies a neighbor solicitation message. Neighbor solicitation messages are sent on the local link when a node wants to determine the link-layer address of another node on the same local link. When a node wants to determine the link-layer address of another node, the source address in a neighbor solicitation message is the IPv6 address of the node sending the neighbor solicitation message. The destination address in the neighbor solicitation message 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.

Figure 7: IPv6 Neighbor Discovery—Neighbor Solicitation Message



After receiving the neighbor solicitation message, the destination node replies by sending a neighbor advertisement message, which has a value of 136 in the Type field of the ICMP packet header, on the local link. The source address in the neighbor advertisement message is the IPv6 address of the node (more specifically, the IPv6 address of the node interface) sending the neighbor advertisement message. The destination address in the neighbor advertisement message 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 sending the neighbor advertisement message.

After the source node receives the neighbor advertisement, the source node and destination node can communicate.

Neighbor solicitation messages are also used to verify the reachability of a neighbor after the link-layer address of a neighbor is identified. When a node wants to verifying the reachability of a neighbor, the destination address in a neighbor solicitation message is the unicast address of the neighbor.

Neighbor advertisement messages are also sent when there is a change in the link-layer address of a node on a local link. When there is such a change, the destination address for the neighbor advertisement is the all-nodes multicast address.

Neighbor solicitation messages are also used to verify the reachability of a neighbor after the link-layer address of a neighbor is identified. Neighbor unreachability detection identifies the failure of a neighbor or the failure of the forward path to the neighbor, and is used for all paths between hosts and neighboring nodes (hosts or routers). Neighbor unreachability detection is performed for neighbors to which only unicast packets are being sent and is not performed for neighbors to which multicast packets are being sent.

A neighbor is considered reachable when a positive acknowledgment is returned from the neighbor (indicating that packets previously sent to the neighbor have been received and processed). A positive acknowledgment—from an upper-layer protocol (such as TCP)—indicates that a connection is making forward progress (reaching its destination) or that a neighbor advertisement message in response to a neighbor solicitation message has been received. If packets are reaching the peer, they are also reaching the next-hop neighbor of the source. Therefore, forward progress is also a confirmation that the next-hop neighbor is reachable.

For destinations that are not on the local link, forward progress implies that the first-hop router is reachable. When acknowledgments from an upper-layer protocol are not available, a node probes the neighbor using unicast neighbor solicitation messages to verify that the forward path is still working. The return of a solicited neighbor advertisement message from the neighbor is a positive acknowledgment that the forward path is still working. (Neighbor advertisement messages that have the solicited flag set to a value of 1 are sent only in response to a neighbor solicitation message.) Unsolicited messages confirm only the one-way path from the source to the destination node; solicited neighbor advertisement messages indicate that a path is working in both directions.

Note A neighbor advertisement message that has the solicited flag set to a value of 0 must not be considered as a positive acknowledgment that the forward path is still working.

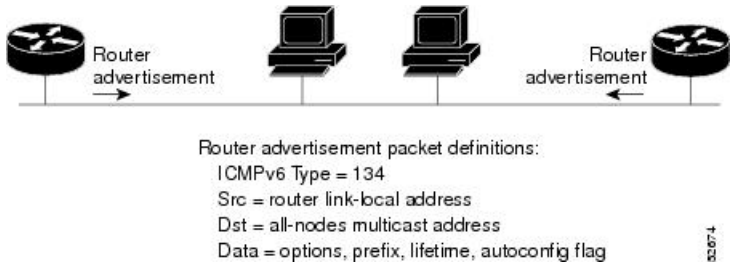
Neighbor solicitation messages are also used in the stateless autoconfiguration process to verify the uniqueness of unicast IPv6 addresses before the addresses are assigned to an interface. Duplicate address detection is performed first on a new, link-local IPv6 address before the address is assigned to an interface. (The new address remains in a tentative state while duplicate address detection is performed.) Specifically, a node sends a neighbor solicitation message with an unspecified source address and a tentative link-local address in the body of the message. If another node is already using that address, the node returns a neighbor advertisement message that contains the tentative link-local address. If another node is simultaneously verifying the uniqueness of the same address, that node also returns a neighbor solicitation message. 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 attempting to verify the same tentative address, the node that sent the original neighbor solicitation message considers the tentative link-local address to be unique and assigns the address to the interface.

Every IPv6 unicast address (global or link-local) must be checked for uniqueness on the link; however, until the uniqueness of the link-local address is verified, duplicate address detection is not performed on any other IPv6 addresses associated with the link-local address. The Cisco implementation of duplicate address detection in the Cisco IOS XR software does not check the uniqueness of anycast or global addresses that are generated from 64-bit interface identifiers.

IPv6 Router Advertisement Message

Router advertisement (RA) messages, which have a value of 134 in the Type field of the ICMP packet header, are periodically sent out each configured interface of an IPv6 router. The router advertisement messages are sent to the all-nodes multicast address.

Figure 8: IPv6 Neighbor Discovery—Router Advertisement Message



Router advertisement messages typically include the following information:

- One or more onlink IPv6 prefixes that nodes on the local link can use to automatically configure their IPv6 addresses
- Lifetime information for each prefix included in the advertisement
- Sets of flags that indicate the type of autoconfiguration (stateless or statefull) that can be completed
- Default router information (whether the router sending the advertisement should be used as a default router and, if so, the amount of time, in seconds, that the router should be used as a default router)
- Additional information for hosts, such as the hop limit and MTU a host should use in packets that it originates

Router advertisements are also sent in response to router solicitation messages. Router solicitation messages, which have a value of 133 in the Type field of the ICMP packet header, are sent by hosts at system startup so that the host can immediately autoconfigure without needing to wait for the next scheduled router advertisement message. Given that router solicitation messages are usually sent by hosts at system startup (the host does not have a configured unicast address), the source address in router solicitation messages is usually the unspecified IPv6 address (0:0:0:0:0:0:0:0). If the host has a configured unicast address, the unicast address of the interface sending the router solicitation message is used as the source address in the message. The destination address in router solicitation messages is the all-routers multicast address with a scope of the link. When a router advertisement is sent in response to a router solicitation, the destination address in the router advertisement message is the unicast address of the source of the router solicitation message.

The following router advertisement message parameters can be configured:

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

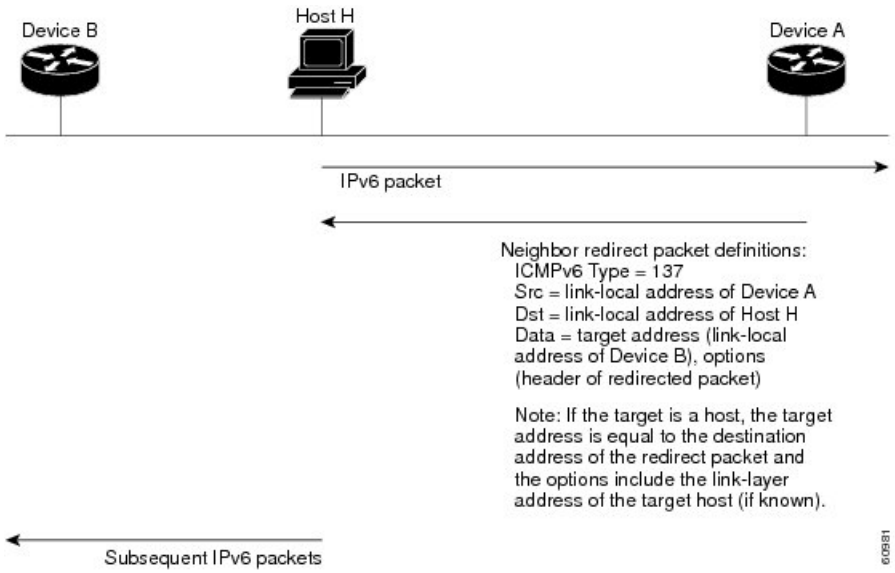
The configured parameters are specific to an interface. The sending of router advertisement messages (with default values) is automatically enabled on Ethernet and FDDI interfaces. For other interface types, the sending of router advertisement messages must be manually configured by using the **no ipv6 nd suppress-ra** command in interface configuration mode. The sending of router advertisement messages can be disabled on individual interfaces by using the **ipv6 nd suppress-ra** command in interface configuration mode.

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

IPv6 Neighbor Redirect Message

A value of 137 in the Type field of the ICMP packet header identifies an IPv6 neighbor redirect message. Routers send neighbor redirect messages to inform hosts of better first-hop nodes on the path to a destination.

Figure 9: IPv6 Neighbor Discovery—Neighbor Redirect Message



Note A router must be able to determine the link-local address for each of its neighboring routers to ensure that the target address (the final destination) in a redirect message identifies the neighbor router by its link-local address. For static routing, the address of the next-hop router should be specified using the link-local address of the router; for dynamic routing, all IPv6 routing protocols must exchange the link-local addresses of neighboring routers.

After forwarding a packet, a router should send a redirect message to the source of the packet under the following circumstances:

- 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.

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

Note A router must not update its routing tables after receiving a neighbor redirect message, and hosts must not originate neighbor redirect messages.

Configuring IPARM Conflict Resolution

This task sets the IP Address Repository Manager (IPARM) address conflict resolution parameters:

- Static Policy Resolution
- Longest Prefix Address Conflict Resolution
- Highest IP Address Conflict Resolution
- Route-Tag Support for Connected Routes

Static Policy Resolution

The static policy resolution configuration prevents new address configurations from affecting interfaces that are currently running.

Configuration Example

Sets the conflict policy to static, that is, prevents new interface addresses from affecting the currently running interface.

```
Router# configure
Router(config)#ipv4 conflict-policy static
*/For IPv6, use the ipv6 conflict-policy static command/*
Router(config)#commit
```

Running Configuration

```
Router#show running-config | in ipv4 config
Building configuration...
!! IOS XR Configuration version = 6.0.0.26I
!! Last configuration change at Mon Dec 14 21:57:27 2015 by root
!
hostname sample-83
logging console debugging
username root
 group root-lr
 group test
 secret 5 $1$d2NC$RbAdqdU7kw/eKJpMo/GJI1
!
```

```
cdp
ipv4 conflict-policy static
interface Loopback0
 ipv4 address 192.0.2.1 255.255.255.0
!
…..
```

Verification

```
Router#show arm ipv4 conflicts
F Forced down
| Down interface & addr             Up interface & addr VRF
F Hu0/0/0/1 192.0.2.1/24    Hu0/0/0/1 192.0.2.27/24 default
```

Longest Prefix Address Conflict Resolution

This conflict resolution policy attempts to give highest precedence to the IP address that has the longest prefix length, that is, all addresses within the conflict-set that do not conflict with the longest prefix address of the currently running interface are allowed to run as well.

Configuration Example

Configures longest prefix address conflict resolution.

```
Router# configure
Router(config)# ipv4 conflict-policy longest-prefix
*/For IPv6, use the ipv6 conflict-policy command*/
Router(config)# commit
```

Running Configuration

```
Router# show running-config | in longest-prefix
Building configuration...
ipv4 conflict-policy longest-prefix
```

Verification

```
Router#show arm ipv4 conflicts
F Forced down
| Down interface & addr             Up interface & addr VRF

F Hun0/0/0/1 192.0.2.2/24   HundredGigE0/0/0/1  192.0.2.1/24 default
```

Highest IP Address Conflict Resolution

This conflict resolution policy attempts to give highest precedence to the IP address that has the highest value, that is, the IP address with the highest value gets precedence.

Configuration

Configures highest IP address conflict resolution.

```
Router# configure
Router(config)#ipv4 conflict-policy highest-ip
*/For IPv6, use the ipv6 conflict-policy highest-ip command/*
Router(config)#commit
```

Running Configuration

```
Router#show running-config | in highest-ip
Building configuration...
ipv4 conflict-policy highest-ip
```

Verification

```
Router#show arm ipv4 conflicts
F Forced down
| Down interface & addr             Up interface & addr VRF

F Hun0/0/0/1 192.0.2.2/24   HundredGigE0/0/0/1 192.0.2.1/24 default

Forced down interface         Up interface                   VRF
```

Route-Tag Support for Connected Routes

The Route-Tag Support for Connected Routes feature attaches a tag with all IPv4 and IPv6 addresses of an interface. The tag is propagated from the IPv4 and IPv6 management agents (MA) to the IPv4 and IPv6 address repository managers (ARM) to routing protocols, thus enabling the user to control the redistribution of connected routes by looking at the route tags, by using routing policy language (RPL) scripts. This prevents the redistribution of some interfaces, by checking for route tags in a route policy. The route tag feature is already available for static routes and connected routes (interfaces) wherein the route tags are matched to policies and redistribution can be prevented.

Configuration Example

Specifies an IPv4 address 192.0.2.27 that has a route tag of 20 to the interface HundredGigE 0/0/0/24.

```
Router# configure
Router(config)#interface HundredGigE 0/0/0/24
Router(config-if)#ipv4 address 192.0.2.27/24 route-tag 20
Router(config)#commit
```

Running Configuration

```
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 route-tag 20
 shutdown
!
```

Verification

Verify the parameters of the route.

```
Router#show route 192.0.2.27
Routing entry for 192.0.2.27/24
  Known via "local", distance 0, metric 0 (connected)
  Tag 20
Routing Descriptor Blocks
    directly connected, via HundredGigE0/0/0/24
      Route metric is 0
  No advertising protos.
```

Address Repository Manager

IPv4 and IPv6 Address Repository Manager (IPARM) enforces the uniqueness of global IP addresses configured in the system, and provides global IP address information dissemination to processes on route processors (RPs) and line cards (LCs) using the IP address consumer application program interfaces (APIs), which includes unnumbered interface information.

Address Conflict Resolution

There are two parts to conflict resolution; the conflict database and the conflict set definition.

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 are said to be part of a conflict set if at least one prefix in the set conflicts with every other IP address belonging to the same set. For example, the following four addresses are part of a single conflict set.

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

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

This conflict policy algorithm is deterministic, that is, the user can tell which addresses on the interface are enabled or disabled. The address on the interface that is enabled is declared as the highest precedence ip address for that conflict set.

The conflict policy algorithm determines the highest precedence ip address within the set.

VRF Fallback and Redirect

Table 6: 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. 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.

Virtual Routing and Forwarding (VRF) is an IP technology that allows multiple instances of a routing table to coexist simultaneously on the same router. Because the routing instances are independent, the same IP addresses can be used without conflict.

If the destination prefix of a data packet does not match any route in the configured VRF, a default route is identified from the global routing table. However, using a default route needs an explicit next hop and that may not be efficient. A better option is to configure a fallback VRF route. If the destination does not have a match in the VRF table, the fallback VRF table is used. The fallback VRF can either be the global routing table or a non-global VRF table.

Restrictions

The following restrictions apply if you configure a fallback VRF route:

- You can configure only one fallback VRF route for each address family of each primary VRF.
- Ping, traceroute, or any slow path application is not supported on fallback VRF because there is no support for LPTS receive trap.
- Only 512 VRFs and 1 global table are supported ion the router.
- If you configure a static default route to a VRF, the static default route takes precedence over the fallback VRF. If you configure the default route for a VRF, the global routing table is used for a route lookup. The default route is always directed to the configured next hop.
- If a route lookup for a packet fails in the primary VRF, the route lookup is retried in the fallback VRF.
- If you configure both ACL-based forwarding (ABF) VRF redirect and VRF fallback for a packet, there will be up to 25% PPS impact in forwarding performance as both ABF VRF redirect and VRF fallback are processed at full throughput rate of the system.

- If a route for a packet is found in the fallback VRF, only the Glean IPv4 and Glean IPv6 adjacency packets are punted successfully.
- In a looped configuration, if the route for a packet is not found in both the primary and fallback VRF, the packet loops in the recycle path. Eventually, the packet is dropped in the recycle egress queue. The recycle queue is of highest priority. Therefore, if there is a high rate of looped traffic, other good recycled packets may be dropped.

Configure VRF Fallback and Redirect

- Configure VRF Fallback
- Configure VRF Redircct

Configure VRF Fallback

Configure the **fallback-vrf** command for a destination that does not match any routes in the configured VRF.

```
Router# configure
Router(config)# vrf vrf1
Router(config-vrf)# fallback-vrf vrf2
```

Verification

```
Router# show access-list 8075-BYPASS-V4 hardware ingress detail location 0/0/CPU0
Hit Packet Count: 1123

show cef vrf vrf100 192.0.2.1 hardware egress location 0/1/CPU0
0.0.0.0/0, version 0, proxy default, internal 0x1200011 0x0 (ptr 0x8983f534) [1], 0x0
(0x894fa728), 0x0 (0x0)
Updated Mar 21 14:01:43.765
Prefix Len 0, traffic index 0, precedence n/a, priority 15
   via 0.0.0.0/32, 0 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8871b168 0x0]
    next hop VRF - 'vrf200', table - 0xe0000008
    next hop 0.0.0.0/32
LEAF - HAL pd context :
sub-type : IPV4, ecd_marked:0, has_collapsed_ldi:0
collapse_bwalk_required:0, ecdv2_marked:0,
HW Walk:
```

Configure VRF Redirect

You can configure the VRF R edirect using an ACL-based forwarding (ABF).

```
Router(config)# ipv6 access-list IPV6-BYPASS-ACL
Router(config)# 5 permit ipv6 any any nexthop1 vrf tms-bypass-l3vrf
Router(config)# ipv4 access-list 8075-BYPASS-V4
Router(config)# 10  permit ipv4 any any nexthop1 vrf tms-bypass-l3vrf
/* You can configure either a HundredGig or a bundle interface. */
Router(config)# interface HundredGigE0/0/0/1
Router(config-if)# ipv4 address 103.4.96.1 255.255.255.254
Router(config-if)# ipv6 address 2620:0:1cff:dead:beee::1/127
Router(config-if)# ipv4 access-group 8075-BYPASS-V4 ingress
Router(config-if)# ipv6 access-group IPV6-BYPASS-ACL ingress
```

Running Configuration

```
ipv6 access-list IPV6-BYPASS-ACL
 5 permit ipv6 any any nexthop1 vrf tms-bypass-l3vrf
!
ipv4 access-list 8075-BYPASS-V4
 10  permit ipv4 any any nexthop1 vrf tms-bypass-l3vrf
!
interface HundredGigE0/0/0/1  (or bundle interface)
ipv4 address 103.4.96.1 255.255.255.254
 ipv6 address 2620:0:1cff:dead:beee::1/127
 ipv4 access-group 8075-BYPASS-V4 ingress
 ipv6 access-group IPV6-BYPASS-ACL ingress
```

Verification

Verify VRF redirect.

```
show route vrf tms-bypass-l3vrf
S*   0.0.0.0/0 [1/0] via 0.0.0.0 (nexthop in vrf tms-l3vrf), 00:15:48
B    99.99.99.0/24 [20/2219] via 29.55.0.1 (nexthop in vrf default), 00:08:30

show access-list 8075-BYPASS-V4 hardware ingress detail  location 0/0/CPU0
Hit Packet Count: 1123
```

Local Station MAC Address for Router

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
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)

When the router receives a Layer 3 traffic packet, the destination MAC address in it would be the MAC address of the interface in the router that accepted the traffic packet. In dynamic ARP, the router learns such MAC address while exchanging ARP response packets. But, when using static ARP, the router needs to support a system-wide MAC address to identify the router. This system-wide MAC address should be available in the static ARP table of the router. Previously, the router had the MAC address of each interface in the static ARP table. Here, the static ARP entries utilize higher resources on the router as it needs to maintain a pool of static ARP entries for all the interfaces against the current implementation with one single entry per system.

Restrictions

- The Cisco Silicon One Q100 ASICs don't support Local Station MAC Address for the Router. For a list of Cisco 8000 Series hardware using Cisco Silicon One Q100 ASIC, refer Cisco 8000 Series Routers Data Sheet.

Configuration Example

The following section details on configuring the Local Station MAC address for a router:

Configuration

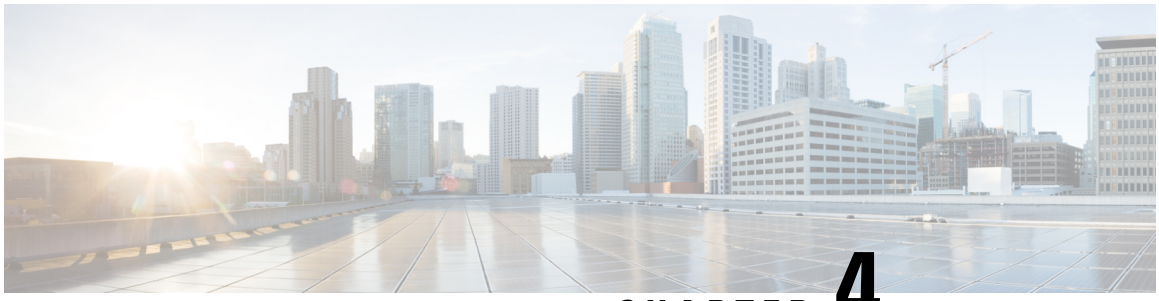
```
Router# config
Router(config)# hw-module local-station-mac B03F.C98C.B948
Router(config)# commit
```

Running Configuration

```
Router# show running-config hw-module local-station-mac
hw-module local-station-mac B03F.C98C.B948
!
```

Verification

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



CHAPTER 4

Configuring ARP

- Configuring ARP, on page 41
- Direct Attached Gateway Redundancy, on page 50

Configuring ARP

Address resolution is the process of mapping network addresses to Media Access Control (MAC) addresses, which is typically done dynamically by the system using the ARP protocol, but can also be done by Static ARP entry configuration. This process is accomplished using the Address Resolution Protocol (ARP).

ARP is used to associate IP addresses with media or MAC addresses. Taking an IP address as input, ARP determines the associated media address. After a media or MAC address is determined, the IP address or media address association is stored in an ARP cache for rapid retrieval. Then the IP datagram is encapsulated in a link-layer frame and sent over the network.

ARP and Proxy ARP

Two forms of address resolution are supported by Cisco IOS XR software: Address Resolution Protocol (ARP) and proxy ARP, as defined in RFC 826 and RFC 1027, respectively. Cisco IOS XR software also supports a form of ARP called local proxy ARP.

For more details on Proxy ARP and Local Proxy ARP, see Proxy ARP and Local Proxy ARP, on page 44

Restrictions

The following restrictions apply to configuring ARP :

- Reverse Address Resolution Protocol (RARP) is not supported.
- ARP throttling, which is the rate limiting of ARP packets in Forwarding Information Base (FIB), is not supported.

Address Resolution Overview

A device in the IP can have both a local address (which uniquely identifies the device on its local segment or LAN) and a network address (which identifies the network to which the device belongs). The local address is more properly known as a *data link address*, because it is contained in the data link layer (Layer 2 of the OSI model) part of the packet header and is read by data-link devices (bridges and all device interfaces, for

example). The more technically inclined person will refer to local addresses as *MAC addresses*, because the MAC sublayer within the data link layer processes addresses for the layer.

To communicate with a device on Ethernet, for example, Cisco IOS XR software first must determine the 48-bit MAC or local data-link address of that device. The process of determining the local data-link address from an IP address is called address resolution.

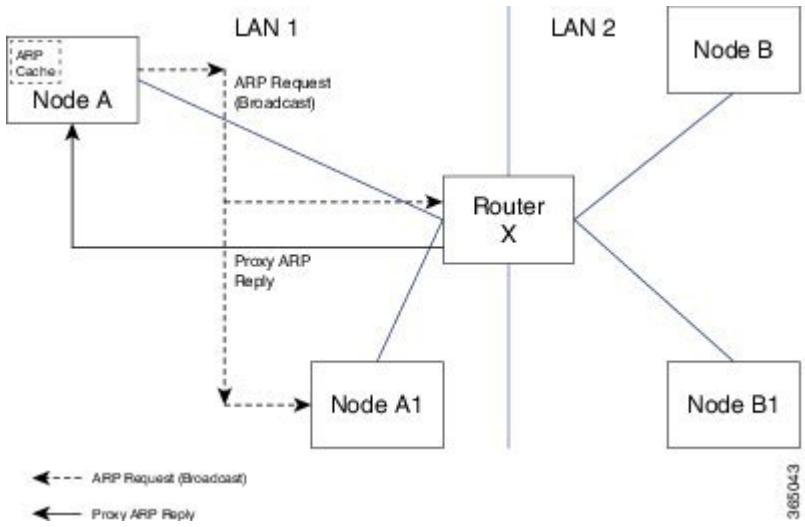
Address Resolution on a Single LAN

1. End System A (Node A) broadcasts an ARP request onto the LAN, attempting to learn the MAC address of End System B (Node B).
2. The broadcast is received and processed by all devices on the LAN, including End System B.
3. Only End System B replies to the ARP request. It sends an ARP reply containing its MAC address to End System A (Node A).
4. End System A (Node A) receives the reply and saves the MAC address of End System B in its ARP cache. (The ARP cache is where network addresses are associated with MAC addresses.)

   On learning the entry from End System B (Node B) adjacency is programmed in the hardware of system A (Node A). Further, packet forwarding for Node B is taken care by the hardware of system A (Node A).

Address Resolution When Interconnected by a Router

The following process describes address resolution when the source and destination devices are attached to different LANs that are interconnected by a router (only if proxy-arp is turned on):



1. End System Y (Node A) broadcasts an ARP request onto the LAN, attempting to learn the MAC address of End System Z (Node B).
2. The broadcast is received and processed by all devices on the LAN, including Router X.
3. Router X checks its routing table and finds that End System Z (Node B) is located on a different LAN.
4. Router X therefore acts as a proxy for End System Z (Node B). It replies to the ARP request from End System Y (Node A), sending an ARP reply containing its own MAC address as if it belonged to End System Z (Node B).

5. End System Y (Node A) receives the ARP reply and saves the MAC address of Router X in its ARP cache, in the entry for End System Z (Node B).
6. When End System Y (Node A) needs to communicate with End System Z (Node B), it checks the ARP cache, finds the MAC address of Router X, and sends the frame directly, without using ARP requests.
7. Router X receives the traffic from End System Y (Node A) and forwards it to End System Z (Node B) on the other LAN.

ARP Cache Entries

ARP establishes correspondences between network addresses (an IP address, for example) and Ethernet hardware addresses. A record of each correspondence is kept in a cache for a predetermined amount of time and then discarded.

You can also add a static (permanent) entry to the ARP cache that persists until explicitly removed.

Defining a Static ARP Cache Entry

ARP and other address resolution protocols provide a dynamic mapping between IP addresses and media addresses. Because most hosts support dynamic address resolution, generally you need not specify static ARP entries. If you must define them, you can do so globally. Performing this task installs a permanent entry in the ARP cache. Cisco IOS XR software uses this entry to translate 32-bit IP addresses into 48-bit hardware addresses.

Optionally, you can specify that the software responds to ARP requests as if the software was identified by the specified IP address, by making an alias entry in the ARP cache.

Configuration Example

A cache entry is created to establish connection between an IP address **203.0.113.2** and the MAC address **0010.9400.000c**. Additionally, the cache entry is created as an alias entry such that the interface to which the entry is attached will respond to ARP request packets for this network layer address with the data link layer address in the entry.

```
Router#config
Router(config)#arp 203.0.113.2 0010.9400.000c arPA
Router(config)#commit
```

Running Configuration

```
Router#show run arp 203.0.113.2 0010.9400.000c arpA
arp vrf default 203.0.113.2 0010.9400.000c ARPA
```

Verification

Verify that the State is static for proper functioning:

```
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

When proxy ARP is disabled, the networking device responds to ARP requests received on an interface only if one of the following conditions is met:

- The target IP address in the ARP request is the same as the interface IP address on which the request is received.
- The target IP address in the ARP request has a statically configured ARP alias.

When proxy ARP is enabled, the networking device also responds to ARP requests that meet all the following conditions:

- The target IP address is not on the same physical network (LAN) on which the request is received.
- The networking device has one or more routes to the target IP address.
- All of the routes to the target IP address go through interfaces other than the one on which the request is received.

When local proxy ARP is enabled, the networking device responds to ARP requests that meet all the following conditions:

- The target IP address in the ARP request, the IP address of the ARP source, and the IP address of the interface on which the ARP request is received are on the same Layer 3 network.
- The next hop for the target IP address is through the same interface as the request is received.

Typically, local proxy ARP is used to resolve MAC addresses to IP addresses in the same Layer 3 network. Local proxy ARP supports all types of interfaces supported by ARP and unnumbered interfaces.

Enabling Proxy ARP

Cisco IOS XR software uses proxy ARP (as defined in RFC 1027) to help hosts with no knowledge of routing determine the media addresses of hosts on other networks or subnets. For example, if the router receives an ARP request for a host that is not on the same interface as the ARP request sender, and if the router has all of its routes to that host through other interfaces, then it generates a proxy ARP reply packet giving its own local data-link address. The host that sent the ARP request then sends its packets to the router, which forwards them to the intended host. Proxy ARP is disabled by default; this task describes how to enable proxy ARP if it has been disabled.

Configuration Example

Proxy ARP is enabled on the HundredGigE interface-0/0/0/0:

```
Router#configure
Router(config)#interface HundredGigE 0/0/0/0
Router(config-if)#proxy-arp
Router(config-if)#commit
```

Running Configuration

```
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
```

```
!
!
```

Verification

Verify that proxy ARP is configured and enabled:

```
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
  IPv4 caps added (state up)
  MPLS caps not added
  Interface not virtual, not client fwd ref,
  Proxy arp is configured, is enabled
  Local Proxy arp not configured
  Packet IO layer is NetIO
  Srg Role : DEFAULT
  Idb Flag : 262332
  IDB is Complete
```

Enabling Local Proxy ARP

Local proxy ARP is used to resolve MAC addresses to IP addresses in the same Layer 3 network such as, private VLANs that are Layer 2-separated. Local proxy ARP supports all types of interfaces supported by ARP and unnumbered interfaces.

Configuration Example

Local proxy ARP is enabled on the HundredGigE interface-0/0/0/0

```
Router#configure
Router(config)#interface HundredGigE 0/0/0/0
Router(config-if)#local-proxy-arp
Router(config-if)#commit
```

Running Configuration

```
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
!
```

Verification

Verify that local proxy ARP is configured:

```
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
  IPv4 caps added (state up)
  MPLS caps not added
  Interface not virtual, not client fwd ref,
```

```
  Proxy arp not configured, not enabled
  Local Proxy arp is configured
  Packet IO layer is NetIO
  Srg Role : DEFAULT
  Idb Flag : 264332
  IDB is Complete
```

Associated Commands

- local-proxy-arp
- show arp idb

Configuring ARP purge-delay

With Equal Cost Multi Path (ECMP), traffic is load balanced across multiple paths with equal cost. This should provide resiliency against interface flaps. If an interface goes down, the traffic is then routed via the other interface without traffic loss. However, if the first interface comes up, traffic is routed back over it but forwarding will only resume once ARP has been (re)resolved and the adjcency (re)installed. Here a short unexpected interface flap causes this traffic loss and is particularly undesirable.

The purge-delay feature allows existing dynamic entries to persist rather than immediately delete entries which could cause traffic loss following an interface flap.

The purge delay feature works by caching existing dynamic ARP entries when an interface goes down and starting a purge delay timer. When the interface is brought back and the purge delay timer not yet fired, the entries are reinstalled as before. The normal entry timeout is reduced in order to re-ARP for the entries after any interface state change related churn has died down; should the purge delay timer fire before the interface comes back up, the entries are deleted from the cache.

Configuring Example

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

Running Configuration

```
Router#show running-config interface HundredGigE 0/0/0/34
Wed Jul 24 09:14:28.200 UTC
interface HundredGigE0/0/0/34
 arp purge-delay 100
 shutdown
!
```

Verification

```
Router#show arp idb HundredGigE 0/0/0/34 location 0/RP0/CPU0
Wed Jul 24 09:16:16.593 UTC

HundredGigE0/0/0/34 (0x0f000208):
  IDB Client: default
  IPv4 address 19.0.2.1, Vrf ID 0x00000000
  VRF Name unknown
  Dynamic learning: Enable
  Dynamic entry timeout: 14400 secs
  Drop adjacency timeout: Disable
  Purge delay: 100 seconds
```

```
  IPv4 caps not added
  MPLS caps not added
  Interface not virtual, not client fwd ref,
  Proxy arp not configured, not enabled
  Local Proxy arp not configured
  Packet IO layer is NetIO
  Srg Role : DEFAULT
  Idb Flag : 0
  IDB is not Complete
, No Media, No HW Addr, No IPv4 Caps, No IPv4 State, No IPv4 Addr, No VRF

 Total entries : 1
| Event Name            |     Time Stamp       | S, M
| idb-update               | Jul 24 09:13:42.912  | 1, 0
```

Configuring ARP Timeout

Dynamic ARP entries which are learnt by ARP address resolution (when valid ARP replies are received) are timed out every 4 hours by default in order to remove stale entries.

ARP entries that correspond to the local interface or that are statically configured by the user never time out.

Configuring Example

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

Running Configuration

```
Router#show running-config interface HundredGigE 0/0/0/35
Wed Jul 24 08:56:56.428 UTC
interface HundredGigE0/0/0/35
 arp timeout 100
 shutdown
!
```

Verification

```
Router#show arp idb HundredGigE 0/0/0/35 location 0/RP0/CPU0
Wed Jul 24 09:04:55.127 UTC

HundredGigE0/0/0/35 (0x0f000200):
  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
  IPv4 caps not added
  MPLS caps not added
  Interface not virtual, not client fwd ref,
  Proxy arp not configured, not enabled
  Local Proxy arp not configured
  Packet IO layer is NetIO
  Srg Role : DEFAULT
  Idb Flag : 0
  IDB is not Complete
```

```
, No Media, No HW Addr, No IPv4 Caps, No IPv4 State, No IPv4 Addr, No VRF

 Total entries : 1
| Event Name            |     Time Stamp       | S, M
| idb-update              | Jul 24 08:56:13.440  | 1, 0
```

Limit ARP Cache Entries per Interface

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
Limit Address Resolution Protocol (ARP) Cache Entries per Interface	Release 7.9.1	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.

The ARP cache overflow occurs when the number of entries in the cache exceeds the maximum limit value of 127999. Such instances make the router vulnerable to threats like DOS attacks. With this feature, you can configure the maximum limit of dynamic ARP entries learned per interface. The router won't accept any cache entries unless cleared after the number entries exceeds the maximum limit in the configuration. You can configure the maximum limit range of 0–127999 per interfaces in the router.

Note The arp cache resources vary depending on the hardware resources available in a router. Ensure the cache-limit configured such that the available resources in the router are able to accomodate the entries.

Feature highlights

This section details the good to know information for using ARP overflow protection:

- The router drops new ARP requests when the number of entries are more than or equal to the applied cache limit value.
- The router won't learn from ARP packets received after exceeding the applied cache limit value.
- The ARP cache limit isn't applicable to static ARP entries.
- The router doesn't enforce the ARP cache limit on ARP client triggered entries.
- The router issues a syslog message when it reaches the cache limit. For every 1000 entries after the cache limit, the router issues a new syslog message. The syslog message includes the interface name and cache

entries drop counters. For example, `RP/0/RP0/CPU0:Jul 1 10:10:25.781 IST: grid_svr[211]: %L2-GRID-4-BANK_FULL : GRID POOL:GLIF(2), BANK 0 FULL. Max size 4091, Curr RIDs 4091.`

- You can view the ARP entries statistics using the **show arp idb** command.
- The ARP Cache limit doesn't drops the already learned dynamic ARP entries. That is, if the number of dynamic ARP entries in the cache is higher or equal to the newer cache limit set in the router, then the router will neither take any new entries or drop the preexisting entries in the cache, but it will start issuing the syslog message the cache limit.

Configuration Example

The following example shows how to set the ARP cache limit for an interface:

Configuration

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

Running Configuration

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

Verification

```
Router#show arp idb HundredGigE 0/0/0/0 location RP0
HundredGigE (0x00000090):
  IDB Client: default
  IPv4 address 1.1.1.1, 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
  IPv4 caps added (state up)
  MPLS caps not added
  Interface is virtual, not client fwd ref,
  Proxy arp not configured, not enabled
  Local Proxy arp not configured
  Packet IO layer is SPIO
  Srg Role : DEFAULT
  Idb Flag : 49294
  IDB is Complete
  IDB Flag Description:
  [VIRTUAL | CAPS | COMPLETE | IPV4_CAPS_CREATED |
   SPIO_ATTACHED | SPIO_SUPPORTED]
  Idb Flag Ext : 0x0
  Idb Oper Progress : NONE
  Client Resync Time : N/A
```

Policing Duplicate ARP Packets

This example sets the police interval within which the duplicate ARP packets from the same sender protocol address (IP) or same source MAC address are policed:

```
configure
arp police-interval 34
```

ARP Policer Behaviour

- When the **arp police-interval** command is configured, ARP requests coming from the same IP address or MAC address within the configured interval are dropped.
- If there was an ARP response existing for the ARP request from the same IP address within the configured interval, the new request will be dropped.
- If the ARP response was not existing already for the ARP request from the same IP address, then the MAC policer will be checked.

Direct Attached Gateway Redundancy

Direct Attached Gateway Redundancy (DAGR) allows third-party redundancy schemes on connected devices to use gratuitous ARP as a failover signal, enabling the ARP process to advertise a new type of route in the Routing Information Base (RIB). These routes are distributed by Open Shortest Path First (OSPF).

Sometimes part of an IP network requires redundancy without routing protocols. A prime example is in the mobile environment, where devices such as base station controllers and multimedia gateways are deployed in redundant pairs, with aggressive failover requirements (subsecond or less), but typically do not have the capability to use native Layer 3 protocols such as OSPF or Intermediate System-to-Intermediate System (IS-IS) protocol to manage this redundancy. Instead, these devices assume they are connected to adjacent IP devices over an Ethernet switch, and manage their redundancy at Layer 2, using proprietary mechanisms similar to Virtual Router Redundancy Protocol (VRRP). This requires a resilient Ethernet switching capability, and depends on mechanisms such as MAC learning and MAC flooding.

DAGR is a feature that enables many of these devices to connect directly to without an intervening Ethernet switch. DAGR enables the subsecond failover requirements to be met using a Layer 3 solution. No MAC learning, flooding, or switching is required.

Note Since mobile devices' 1:1 Layer 2 redundancy mechanisms are proprietary, they do not necessarily conform to any standard. So although most IP mobile equipment is compatible with DAGR, interoperability does require qualification, due to the possibly proprietary nature of the Layer 2 mechanisms with which DAGR interfaces.

Restrictions

The following additional restrictions apply when configuring the Direct Attached Gateway Redundancy (DAGR) feature:

- IPv6 is not supported.
- Ethernet bundles are not supported.

- Non-Ethernet interfaces are not supported.
- Hitless ARP Process Restart is not supported.
- Hitless RSP Failover is not supported.

Configuring DAGR

Configuration Example

```
Router# configure terminal
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# arp dagr
Router(config-if-dagr)# peer ipv4 192.0.2.1
Router(config-if-dagr-peer)# route distance normal 140 priority 3
Router(config-if-dagr-peer)# route metric normal 84 priority 80
Router(config-if-dagr-peer)# timers query 2 standby 19
Router(config-if-dagr-peer)# priority-timeout 25
Router(config-if-dagr)# commit
```

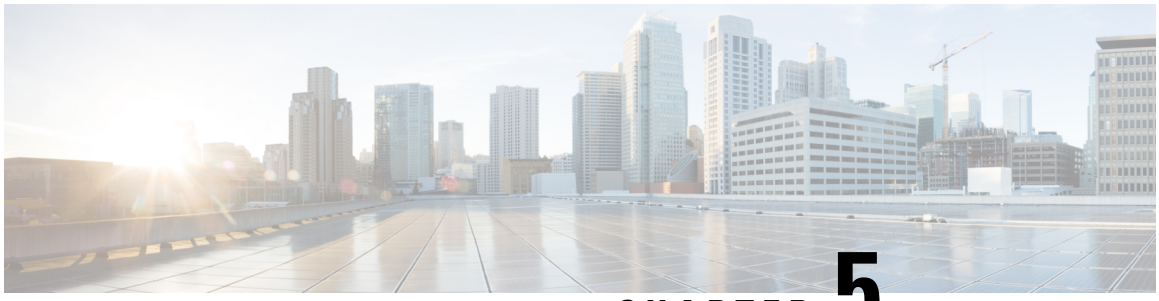
Running Configuration

```
configure
 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
    commit
```

Verification

The following example shows how to display the current operational state of the DAGR groups:

```
Router# show arp dagr
------------------------------------------------------------------------------
0/1/CPU0
------------------------------------------------------------------------------
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
```



CHAPTER 5

Implementing DHCP

This module describes the concepts and tasks you will use to configure Dynamic Host Configuration Protocol (DHCP).

- Introduction to DHCP Relay, on page 53
- Prerequisites for Configuring DHCP Relay Agent, on page 54
- Limitations for DHCP Relay Feature , on page 54
- How to Configure and Enable DHCP Relay Agent, on page 54
- Implementing DHCP Snooping, on page 64

Introduction to DHCP Relay

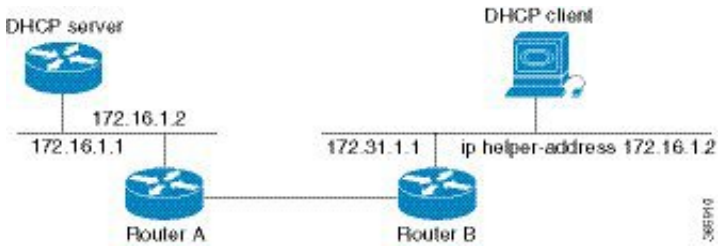
A DHCP relay agent is a host that forwards DHCP packets between clients and servers that do not reside on a shared physical subnet. Relay agent forwarding is distinct from the normal forwarding of an IP router where the IP datagrams switch between networks transparently.

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. You can configure a DHCP relay agent to forward DHCP packets to a remote server. You can configure a DHCP relay profile on the DHCP relay agent and configure one or more helper addresses in it. You can assign the profile to an interface or a VRF.

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

Figure 10: Forwarding UDP Broadcasts to a DHCP Server Using a Helper Address



Prerequisites for Configuring DHCP Relay Agent

The following are the prerequisites to 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 Feature

The limitations for implementing DHCP relay feature are as follows:

- 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.

Note DHCP relay profile submode does not support the configuration of DHCP option code.

How to Configure and Enable DHCP Relay Agent

This section contains the following tasks:

Configuring and Enabling the DHCP Relay Agent

Configuration Example

```
Router# configure
/* Enters the global configuration mode */
```

```
Router(config)# dhcp ipv4
/* Configures DHCP for IPv4 and enters the DHCPv4 configuration submode. */

Router(config-dhcpv4)# profile r1 relay
/* Enables DHCP relay profile */

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
/* Configures VRF addresses for forwarding UDP broadcasts, including DHCP. */

Router(config-dhcpv4-relay-profile)# relay information option vpn
Router(config-dhcpv4-relay-profile)# relay information option vpn-mode rfc
/* Inserts the DHCP relay agent information option (option-82 field) in forwarded BOOTREQUEST
 messages to a DHCP server. */


Router(config-dhcpv4-relay-profile)# relay information option allow-untrusted
/* (Optional) Configures the DHCP IPv4 Relay not to discard BOOTREQUEST packets
that have an existing relay information option and the giaddr set to zero. */

Router(config-dhcpv4-relay-profile)# exit
Router(config-dhcpv4)# interface BVI 1 relay profile r1
Router(config-dhcpv4)# commit
/* Configures DHCP relay on a BVI interface and commits the configuration */
```

Running Configuration

```
Router#show running-config
Tue May 23 10:56:14.463 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Tue May 23 10:56:08 2017 by annseque
!
dhcp ipv4
 vrf vrf1 relay profile client
 profile r1 relay
  helper-address vrf A 10.10.7.1 giaddr 10.10.7.2
  broadcast-flag policy check
  relay information option vpn
  relay information option vpn-mode rfc
  relay information option allow-untrusted
 !
```

Enabling a DHCP Relay Agent on an Interface

This section describes how to enable the Cisco IOS XR DHCP relay agent on an interface.

Configuration Example

The DHCP relay agent is disabled by default.

```
Router#configure

Router(config)#dhcp ipv4
/* Configures DHCP for IPv4 and enters the DHCPv4 configuration submode. */

Router(config-dhcpv4)#interface HundredGigE 0/2/0/2 relay profile client
/* Attaches 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. */

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

Running Configuration

```
Router#show running-config dhcp ipv4
dhcp ipv4
interface HundredGigE 0/2/0/2 relay profile client
!
```

Disabling DHCP Relay on an Interface

This task describes how to disable the DHCP relay on an interface by using the **no** keyword on the interface.

```
Router# configure terminal
Router(config)# dhcp ipv6
Router(config-dhcpv6)# no interface type name relay profile profile-nameRouter(config-dhcpv6)#
 no interface type name none
Router(config-dhcpv6-if)# commit
```

Configuring and Enabling DHCP Relay Agent with DHCP MAC Address Verification

This section discusses how to configure and enable DHCP Relay Agent with DHCP MAC address verification.

Configuration Example

```
Router# configure

Router(config)# dhcp ipv4
/* Configures DHCP for IPv4 and enters the DHCPv4 configuration submode. */

Router(config-dhcpv4)# profile client relay
/* Enables DHCP relay profile */

Router(config-dhcpv4)# client-mac-mismatch action drop
/* Enables MAC address verification. If 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  */

Router(config-dhcpv4-relay-profile)# relay information option
/* Inserts the DHCP relay agent information option (option-82 field) in forwarded
BOOTREQUEST messages to a DHCP server. */

Router(config-dhcpv4-relay-profile)# relay information check
/* (Optional) Configures DHCP to check the validity of the relay agent information
option in forwarded BOOTREPLY messages. */

Router(config-dhcpv4-relay-profile)# relay information policy drop
/* (Optional) Configures the reforwarding policy for a DHCP relay agent;
that is, whether the relay agent will drop or keep (using the 'keep' keyword)
 the relay information. */

Router(config-dhcpv4-relay-profile)# relay information option allow-untrusted
/* (Optional) Configures the DHCP IPv4 Relay not to discard BOOTREQUEST packets that have
an existing
```

```
relay information option and the giaddr set to zero. */

Router(config-dhcpv4-relay-profile)# giaddr policy drop
/* Drops the packet that has an existing nonzero giaddr value. Use the 'replace' keyword
 to replace the existing giaddr value with a value that it generates (the default behavior).
  */

Router(config-dhcpv4-relay-profile)# helper-address vrf vrf1 10.1.1.1
/* Forwards UDP broadcasts, including DHCP. */

Router(config-dhcpv4-relay-profile)# commit

Router(config-dhcpv4-relay-profile)# exit
Router(config-dhcpv4)# vrf vrf1 relay profile client
Router(config-dhcpv4)# commit
/* Configures DHCP Relay on a VRF and commits the entire configuration. */
```

Running Configuration

Confirm your configuration.

```
Router# show run
Thu May 11 09:00:57.839 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Thu May 11 09:00:54 2017 by annseque
!
dhcp ipv4
vrf vrf1 relay profile client
profile client relay
client-mac-match action drop
helper-address vrf vrf1 10.1.1.1
giaddr policy drop
relay information check
relay information option
relay information policy drop
relay information option allow-untrusted
!
!
```

DHCP MAC Address Verification

Use the following show command to check if DHCP MAC addresses are verified on the router.

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

The output validates the verified DHCP MAC address of the packets is verified.

Configuring the DHCPv6 (Stateless) Relay Agent

Use the following steps:

- To specify a destination address for client messages.
- To enable DHCP IPv6 relay service on the interface.

Configuration Example

To configure the DHCPv6 (stateless) relay agent, you must complete the following configurations:

1. Enable the DHCP IPv6 configuration mode.
2. Configure the DHCPv6 relay profile.
3. Configure helper addresses.
4. Specify the interface for the relay profile.

Configuration

```
/* Enter the global configuration mode, and then enter the DHCP IPv6 configuration mode */
Router# configure terminal
Router(config)# dhcp ipv6
Router(config-dhcpv6)# profile test relay
Router(config-dhcpv6-relay-profile)# helper-address vrf default 2001:1::1
Router(config-dhcpv6-relay-profile)# !
Router(config-dhcpv6-relay-profile)# interface TenGigE0/0/0/0 relay profile test
Router(config-dhcpv6)# !
```

Enabling DHCP Relay on a VRF

This task describes how to enable DHCP relay on a VRF.

```
/CPU0:router# configure terminal
Router(config)# dhcp ipv4
Router(config-dhcpv6)# vrf vrf-name relay profile profile-name
Router(config-dhcpv6-if)# commit
```

Configure a DHCP Relay Profile with Multiple Helper Addresses

You can configure up to 16 helper IPv4 and IPv6 addresses for a DHCPv4 or DHCPv6 relay profile.

1. Enter the DHCPv4 or DHCPv6 configuration mode.

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

2. Configure the DHCPv4 or DHCPv6 relay profile.

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

3. Configure helper addresses.

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

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

4. Confirm your configuration.

   ```
   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.

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

6. Exit the configuration mode.

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

You have successfully configured the DHCPv6 relay helper address.

DHCP Relay Agent Notification for Prefix Delegation

DHCP relay agent notification for prefix delegation allows the router working as a DHCPv6 relay agent to find prefix delegation options. The notifications allow you to review the contents of a DHCP RELAY-REPLY packet that is sent to the client. When the relay agent finds the prefix delegation option, it extracts the information about the delegated prefix. After extracting the information, the relay agent inserts an IPv4 or IPv6 subscriber route matching the prefix delegation information onto the relay agent. A relay agent forwards future packets that are destined to that prefix based on the information that is 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 packet forwarding (uRPF) to work by allowing the router doing 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.

DHCPv6 Relay Over BVI for IANA Address Allocation

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 Bridge Virtual Interface (BVI) through Internet Assigned Numbers Authority (IANA) address allocation. Therefore, the DHCPv6 relay agent acts as a stateful relay agents and maintains DHCPv6 binding and respective route entry for the allocated IPv6 addresses.

Restrictions

- 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.

Configuration Example

To configure DHCPv6 Relay Over BVI for IANA Address Allocation, use the following steps.

1. Enter the interface configuration mode and configure a BVI interface.
2. Assign an IPv6 address to the BVI interface.
3. Route the L2 access interface to the L3 BVI interface of the relay agent.
4. Enter the DHCP IPv6 configuration mode and then create a DHCP IPv6 Stateful relay profile.
5. Attach the relay profile to a server address.
6. Configure a stateful relay agent by enabling route allocation through IANA.
7. Attach the BVI Interface to the DHCPv6 relay profile.

Configuration

```
/* Enter the interface configuration mode and configure a BVI interface. */
Router# configure
Router(config)# interface BVI1

Assign an IPv6 address to the BVI interface.
Router(config-if)# ipv6 address 2001:db8::2/64
Router(config-if)# commit
Router(config-if)# exit

/* Route the L2 access interface to the L3 BVI interface of the relay agent. */
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-bg)# exit
Router(config-l2vpn)# exit
Router(config)#

/* Enter the DHCP IPv6 configuration mode and then create a DHCP IPv6 Stateful relay profile.
 */
Router(config)# dhcp ipv6
Router(config-dhcpv6)# profile RELAY1 relay

/* Attach the relay profile to a server address. */
Router(config-dhcpv6-relay-profile)# helper-address vrf default 2001:DB8::1

/* Configure a stateful relay agent by enabling route allocation through IANA. */
Router(config-dhcpv6-relay-profile)# iana-route-add
Router(config-dhcpv6-relay-profile)# exit


/* Attach the BVI Interface to the DHCPv6 relay profile. */
Router(config-dhcpv6-relay-profile)# interface BVI1 relay profile RELAY1
Router(config-dhcpv6-relay-profile)# commit
```

Running Configuration

```
Router# show running configuration
interface BVI1
 ipv6 address 2001:db8::2/64
!
l2vpn
 bridge group 1
  bridge-domain 1
   interface HundredGigE0/0/0/1.100
   !
   routed interface BVI1
   !
  !
 !
!
dhcp ipv6
 profile RELAY1 relay
  helper-address vrf default 2001:db8::1
  iana-route-add
  !
 interface BVI1 relay profile RELAY1
!
```

Verification

Use the following command to verify that more than one DHCP client is bridged over BVI:

```
Router# show dhcp ipv6 relay binding
Thu Nov 21 05:48:38.463 UTC

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
```

Use the following command to verify that a unique IPv6 address is assigned to a client due to IANA allocation:

```
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
```

Configuring the Relay Agent Information Feature

You can configure the DHCP relay agent information option. 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).

Configuration

Use the following steps to configure the DHCP relay agent information option:

1. Enter DHCP IPv4 configuration submode.

2. Enter DHCP IPv4 profile relay submode.
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
  - 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.

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.

5. (Optional) Configure the reforwarding policy for a DHCP relay agent; that is, whether the relay agent will drop or keep the relay information.

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

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

Configuration Example

```
/* Enter DHCP IPv4 configuration submode. */
Router# Configure
Router(config)# dhcp ipv4

/* Enter DHCP IPv4 profile relay submode. */
Router(config-dhcpv4)# profile RELAY relay
```

```
/* Configure the system to insert the DHCP relay agent information option. */
Router(config-dhcpv4-relay-profile)# relay information option

/* (Optional) Configure DHCP to check the validity of the relay agent information option.
*/
Router(config-dhcpv4-relay-profile)# relay information check

/* (Optional) Configure the reforwarding policy for a DHCP relay agent. */
Router(config)# dhcp ipv4 profile TEST relay relay information policy drop
Router(config)# commit
```

Configuring Relay Agent Giaddr Policy

You can configure the DHCP relay agent's processing capabilities for the BOOTREQUEST packets that already contain a nonzero giaddr attribute. Use the `giaddr policy replace` command to replace the existing giaddr value with a value that it generates. Use the `giaddr policy drop` command to drop the packet that has an existing nonzero giaddr value.

Configuration

To configure a relay agent giaddr policy, use the following steps:

- Enter the DHCP IPv4 configuration submode.
- Enter the relay profile submode.
- Configure the giaddr policy.

Configuration Example

```
/* Enter the DHCP IPv4 configuration submode. */
Router# configure
Router(config)# dhcp ipv4

/* Enter the relay profile submode. */
Router(config-dhcpv4)# profile client relay

/* Configure the giaddr policy. */
Router(config-dhcpv4-relay-profile)# giaddr policy drop
Router(config-dhcpv4-relay-profile)# commit
```

Implementing DHCP Snooping

Table 9: Feature History Table

Prerequisites for Configuring DHCP Snooping

The following prerequisites are required example shows how to configure DHCP IPv4 snooping relay agent broadcast flag policy:

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

- A Cisco 8000 Series Router running Cisco IOS XR software.
- A configured and running DHCP client and DHCP server.

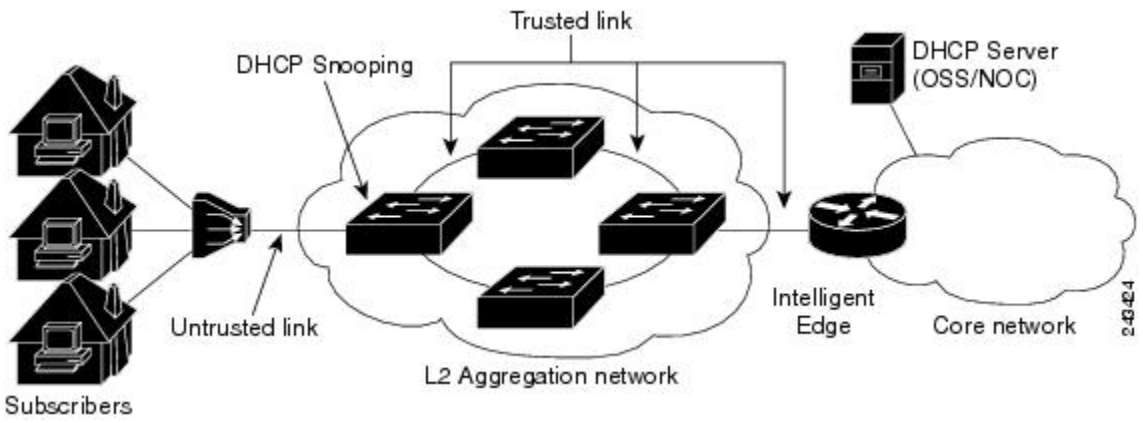
Information about DHCP Snooping

DHCP Snooping features are focused on the edge of the aggregation network. Security features are applied at the first point of entry for subscribers. Relay agent information option information 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 is one providing 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. Figure 11: DHCP Snooping in an Aggregation Network, on page 65 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 connecting the switches in the aggregation network and the link from the aggregation network to the intelligent edge is 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



Configuration guidelines and restrictions for DHCP snooping

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 EVPN (Ethernet VPN) configurations.

Trusted and Untrusted Ports

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 clients'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

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.

Assigning Profiles to a Bridge Domain

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 to 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.

Relay Information Options

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 already in the client packet when it is received. This is a different condition than a DHCP snooping trusted/untrusted port. The **relay information option allow-untrusted** command determines how the DHCP snooping application handles untrusted relay information options.

How to Configure DHCP Snooping

This section contains the following tasks:

Enabling DHCP Snooping in a Bridge Domain

The following configuration creates two ports, a client-facing port and a server-facing port. In Step 1 through Step 8, an untrusted DHCP snooping profile is assigned to the client bridge port and trusted DHCP snooping profile is assigned to the server bridge port. In Step 9 through Step 18, an untrusted DHCP snooping profile is assigned to the bridge domain and trusted DHCP snooping profiles are assigned to server bridge ports.

Procedure

Step 1 Enter the DHCP IPv4 profile configuration submode using the **dhcp ipv4** command.

Example:

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

Step 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
```

Step 3 Exit the DHCP IPv4 profile configuration mode using the **exit** command.

Example:

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

Step 4 Enable DHCP for IPv4 and entersDHCP IPv4 profile configuration mode using the **dhcp ipv4** command.

Example:

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

Step 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
```

Step 6 Configure a DHCP snoop profile to be trusted using the **trusted** command.

Example:

```
Router(config-dhcv4)# trusted
```

Step 7 Exit the DHCP IPv4 profile configuration mode using the **exit** command.

Example:

```
Router(config-dhcv4)# exit
```

Step 8 Enter the l2vpn configuration mode using the **l2vpn** command.

Example:

```
Router(config)# l2vpn
```

Step 9 Create a bridge group to contain bridge domains and enter l2vpn bridge group configuration submode using the **bridge group** *group-name* command.

Example:

```
Router(config-l2vpn)# bridge group ccc
```

Step 10 Establish a bridge domain using the **bridge-domain** *bridge-domain-name* command.

Example:

```
Router(config-l2vpn-bg)# bridge-domain ddd
```

Step 11 Identify the interface using the **interface** *type interface-path-id* command.

Example:

```
Router(config-l2vpn-bg-bd)# interface gigabitethernet 0/1/0/0
```

Step 12 Attach an untrusted DHCP snoop profile to the bridge port using the **dhcp ipv4 snoop profile** *untrusted-profile-name* command.

Example:

```
Router(config-l2vpn-bg-bd-ac)# dhcp ipv4 snoop profile untrustedClientProfile
```

Step 13 Identify the interface using the **interface** *type interface-path-id* command.

Example:

```
Router(config-l2vpn-bg-bd-ac)# gigabitethernet 0/1/0/1
```

Step 14 Attache a trusted DHCP snoop profile to the bridge port using the **dhcp ipv4 snoop profile** *trusted-profile-name* command.

Example:

```
Router(config-l2vpn-bg-bd-ac)# dhcp ipv4 snoop profile trustedServerProfile
```

Step 15 Exit the l2vpn bridge group bridge-domain interface configuration submode using the **exit** command.

Example:

```
Router(config-l2vpn-bg-bd-ac)# exit
```

Step 16 Exit the l2vpn bridge group bridge-domain configuration submode using the **exit** command.

Example:

```
Router(config-l2vpn-bg-bd)# exit
```

Step 17 Commit the configuration changes on the router.

Example:

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

Disabling DHCP Snooping on a Specific Bridge Port

The following configuration enables DHCP to snoop packets on all bridge ports in the bridge domain ISP1 except for bridge port 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

Step 1 Enter l2vpn configuration submode using the **l2vpn** command.

Example:

```
Router(config)# l2vpn
```

Step 2 Create a bridge group to contain bridge domains and enter l2vpn bridge group configuration submode using the **bridge group** *group-name* command.

Example:

```
Router(config-l2vpn)# bridge group GRP1
```

Step 3 Establish a bridge domain and enter l2vpn bridge group bridge-domain configuration submode. **bridge-domain** *bridge-domain-name*

Example:

```
Router(config-l2vpn-bg)# bridge-domain ISP1
```

Step 4 Attach the untrusted DHCP snooping profile to the bridge domain using the **dhcp ipv4 snoop profile** *profile-name* command.

Example:

```
Router(config-l2vpn-bg-bd)# dhcp ipv4 snoop profile untrustedClientProfile
```

Step 5 Identify an interface and enter l2vpn bridge group bridge-domain interface configuration submode using the **interface** *type interface-path-id* command.

Example:

```
Router(config-l2vpn-bg-bd)# interface gigabitethernet 0/1/0/1
```

Step 6 Disable DHCP snooping on the port using the **dhcp ipv4 none** command.

Example:

```
Router(config-l2vpn-bg-bd-if)# dhcp ipv4 none
```

Step 7 Identify an interface and enter l2vpn bridge group bridge-domain interface configuration submode using the **interface** *type interface-path-id* command.

Example:

```
Router(config-l2vpn-bg-bd)# interface gigabitethernet 0/1/0/2
```

Step 8 Attach the trusted DHCP snooping profile to a port using the **dhcp ipv4 snoop profile** *profile-name* command.

Example:

```
Router(config-l2vpn-bg-bd)# dhcp ipv4 snoop profile trustedServerProfile
```

Step 9 Exit l2vpn bridge-domain bridge group interface configuration submode using the **exit** command.

Example:

```
Router(config-l2vpn-bd-bg)# exit
```

Step 10 Exit l2vpn bridge-domain submode using the **exit** command.

Example:

```
Router(config-l2vpn-bg)# exit
```

Step 11 Commit the configuration changes on the router.

Example:

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

Using the Relay Information Option

This task shows 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.

Procedure

Step 1 Enter the DHCP IPv4 profile configuration submode using the **dhcp ipv4** command.

Example:

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

Step 2 Configure an untrusted DHCP snooping profile for the client port using the **profile** *profile-name* **snoop** command.

Example:

```
Router(config-dhcpv4)# profile untrustedClientProfile snoop
```

Step 3 Enable the system to insert the DHCP relay information option field in forwarded BOOTREQUEST messages to a DHCP server using the **relay information option** command.

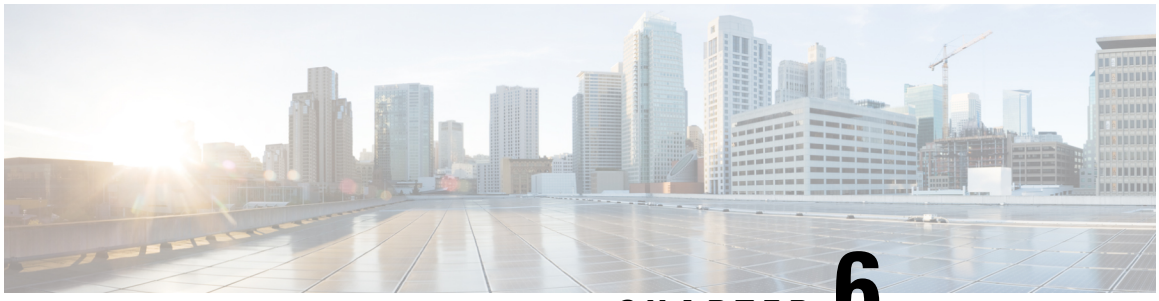
Example:

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

Step 4 Commit the configuration changes on the router.

Example:

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



CHAPTER 6

Implementing Host Services and Applications

- Implementing Host Services and Applications, on page 73
- Network Connectivity Tools, on page 73
- Domain Services, on page 78
- File Transfer Services, on page 79
- Cisco inetd, on page 82
- Telnet, on page 82
- Syslog source-interface, on page 83
- HTTP Client Application, on page 83

Implementing Host Services and Applications

Cisco IOS XR software Host Services and Applications features on the router are used primarily for checking network connectivity and the route a packet follows to reach a destination, mapping a hostname to an IP address or an IP address to a hostname, and transferring files between routers and UNIX workstations.

Prerequisites for implementing Host Services and Applications

Ensure to install the relevant optional RPM package before using the host services or applications. For example, Install Telnet RPM before using Telnet.

Network Connectivity Tools

Network connectivity tools enable you to check device connectivity by running traceroutes and pinging devices on the network:

Ping

The **ping** command is a common method for troubleshooting the accessibility of devices. It uses two Internet Control Message Protocol (ICMP) query messages, ICMP echo requests, and ICMP echo replies to determine whether a remote host is active. The **ping** command also measures the amount of time it takes to receive the echo reply.

The **ping** command first sends an echo request packet to an address, and then it waits for a reply. The ping is successful only if the echo request gets to the destination, and the destination is able to get an echo reply (hostname is alive) back to the source of the ping within a predefined time interval.

The bulk option has been introduced to check reachability to multiple destinations. The destinations are directly input through the CLI. This option is supported for ipv4 destinations only.

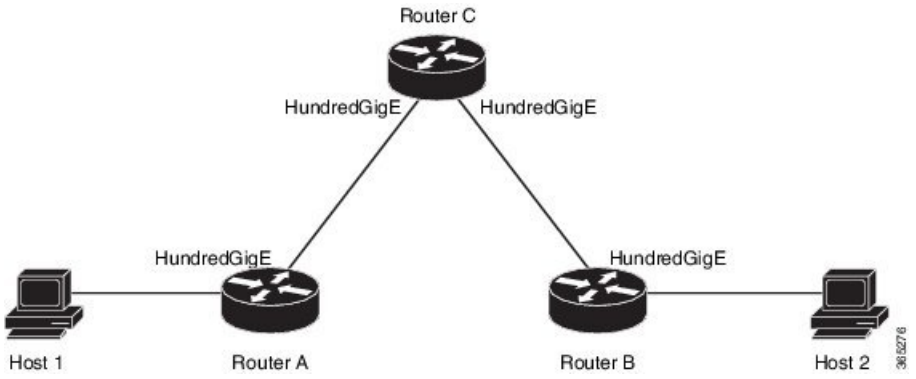
Checking Network Connectivity

As an aid to diagnosing basic network connectivity, many network protocols support an echo protocol. The protocol involves sending a special datagram to the destination host, then waiting for a reply datagram from that host. Results from this echo protocol can help in evaluating the path-to-host reliability, delays over the path, and whether the host can be reached or is functioning.

Configuration for Checking Network Connectivity

The following configuration shows an extended **ping** command sourced from the Router A HundredGigEinterface and destined for the Router B HundredGigE interface. If this ping succeeds, it is an indication that there is no routing problem. Router A knows how to get to the HundredGigEinterface of Router B, and Router B knows how to get to the HundredGigE interface of Router A. Also, both hosts have their default gateways set correctly.

If the extended **ping** command from Router A fails, it means that there is a routing problem. There could be a routing problem on any of the three routers: Router A could be missing a route to the subnet of Router B's interface, or to the subnet between Router C and Router B; Router B could be missing a route to the subnet of Router A's subnet, or to the subnet between Router C and Router A; and Router C could be missing a route to the subnet of Router A's or Router B's Ethernet segments. You should correct any routing problems, and then Host 1 should try to ping Host 2. If Host 1 still cannot ping Host 2, then both hosts' default gateways should be checked. The connectivity between the HundredGigE interface of Router A and the HundredGigE interface of Router B is checked with the extended **ping** command.



With a normal ping from Router A to Router B's HundredGigE interface, the source address of the ping packet would be the address of the outgoing interface; that is the address of the HundredGigE interface, (192.0.2.2). When Router B replies to the ping packet, it replies to the source address (that is, 192.0.2.1). This way, only the connectivity between the HundredGigE interface of Router A (192.0.2.2) and the 10gige interface of Router B (192.0.2.1) is tested.

To test the connectivity between Router A's HundredGigE interface (192.0.2.2) and Router B's interface (192.0.2.1), we use the extended **ping** command. With extended **ping**, we get the option to specify the source address of the **ping** packet.

Configuration Example

In this use case, the extended **ping** command verifies the IP connectivity between the two IP addresses Router A (192.0.2.2) and Router B (192.0.2.1) .

```
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)
Router#!!!!!

*/If you do not enter a hostname or an IP address on the same line as the ping command,
the system prompts you to specify the target IP address and several other command parameters.

After specifying the target IP address, you can specify alternate values for the
remaining parameters or accept the displayed default for each parameter /*

Router# ping
Tue Sep 24 02:41:45.739 UTC
Protocol [ipv4]: ipv4
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]: 36
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 1
Extended commands? [no]: y
Source address or interface: 12.12.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes? [no]:
Type escape sequence to abort.
Sending 5, 36-byte ICMP Echos to 192.0.2.1, timeout is 1 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/7 ms
```

Associated Commands

- ping

Checking Network Connectivity for Multiple Destinations

The bulk option enables you to check reachability to multiple destinations. The destinations are directly input through the CLI. This option is supported for ipv4 destinations only.

Configuration Example

Check reachability and network connectivity to multiple hosts on IP networks with the following IP addresses:

- 1: 192.0.2.1
- 2: 198.51.100.1
- 3: 203.0.113.1

```
Router# ping bulk ipv4 input cli batch
*/You must hit the Enter button and then specify one destination address per line*/
Please enter input via CLI with one destination per line and when done Ctrl-D/(exit) to
initiate pings:
1: 192.0.2.1
2: 198.51.100.1
3: 203.0.113.1
4:
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]: q
% Please answer 'yes' or 'no'.
Sweep range of sizes? [no]: q
% Please answer 'yes' or '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]: q
% A decimal number between 36 and 18024.
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 (4/5),
```

Associated Commands

- ping bulk ipv4

Traceroute

Where 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.

The **traceroute** command records the source of each ICMP "time-exceeded" message to provide a trace of the path that the packet took to reach the destination. You can use the IP **traceroute** command to identify the path that packets take through the network on a hop-by-hop basis. The command output displays all network layer (Layer 3) devices, such as routers, that the traffic passes through on the way to the destination.

The **traceroute** command uses the Time To Live (TTL) field in the IP header to cause routers and servers to generate specific return messages. The **traceroute** command sends a User Datagram Protocol (UDP) datagram to the destination host with the TTL field set to 1. If a router finds a TTL value of 1 or 0, it drops the datagram and sends back an ICMP time-exceeded message to the sender. The traceroute facility determines the address of the first hop by examining the source address field of the ICMP time-exceeded message.

To identify the next hop, the **traceroute** command sends a UDP packet with a TTL value of 2. The first router decrements the TTL field by 1 and sends the datagram to the next router. The second router sees a TTL value of 1, discards the datagram, and returns the time-exceeded message to the source. This process continues until the TTL increments to a value large enough for the datagram to reach the destination host (or until the maximum TTL is reached).

To determine when a datagram reaches its destination, the **traceroute** command sets the UDP destination port in the datagram to a very large value that the destination host is unlikely to be using. When a host receives a datagram with an unrecognized port number, it sends an ICMP port unreachable error to the source. This message indicates to the traceroute facility that it has reached the destination.

Checking Packet Routes

The **traceroute** command allows you to trace the routes that packets actually take when traveling to their destinations.

Configuration Example

Trace the route from 192.0.2.1 to 198.51.100.1:

```
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

*/If you do not enter a hostname or an IP address on the same line as the traceroute
command, the system prompts you to specify the target IP address and several other
command parameters. After specifying the target IP address, you can specify alternate
values for the remaining parameters or accept the displayed default for each
parameter/*

Router #traceroute
Protocol [ipv4]:
Target IP address: 198.51.100.1
Source address: 192.0.2.1
Numeric display? [no]:
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
```

Associated Commands

- traceroute

Domain Services

Cisco IOS XR software domain services acts as a Berkeley Standard Distribution (BSD) domain resolver. The domain services maintains a local cache of hostname-to-address mappings for use by applications, such as Telnet, and commands, such as **ping** and **traceroute**. The local cache speeds the conversion of host names to addresses. Two types of entries exist in the local cache: static and dynamic. Entries configured using the **domain ipv4 host** or **domain ipv6 host** command are added as static entries, while entries received from the name server are added as dynamic entries.

The name server is used by the World Wide Web (WWW) for translating names of network nodes into addresses. The name server maintains a distributed database that maps hostnames to IP addresses through the DNS protocol from a DNS server. One or more name servers can be specified using the **domain name-server** command.

When an application needs the IP address of a host or the hostname of an IP address, a remote-procedure call (RPC) is made to the domain services. The domain service looks up the IP address or hostname in the cache, and if the entry is not found, the domain service sends a DNS query to the name server.

You can specify a default domain name that Cisco IOS XR software uses to complete domain name requests. You can also specify either a single domain or a list of domain names. Any IP hostname that does not contain a domain name has the domain name you specify appended to it before being added to the host table. To specify a domain name or names, use either the **domain name** or **domain list** command.

Configuring Domain Services

DNS-based hostname-to-address translation is enabled by default. If hostname-to-address translation has been disabled using the **domain lookup disable** command, re-enable the translation using the **no domain lookup disable** command.

Configuration Example

Define a static hostname-to-address mapping. Associate (or map) the IPv4 addresses (192.0.2.1 and 10.2.0.2 198.51.100.1) with two hosts. The host names are host1 and host2.

```
Defining the Domain Host
================================
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

Defining the Domain Name
=========================
*/Define cisco.com as the default domain name/*
Router#configure
Router(config)#domain name cisco.com
Router(config)#commit

Specifying the Addresses of the Name Servers
=============================================
*/Specify host 192.168.1.111 as the primary name server
```

```
and host 192.168.1.2 as the secondary server/*
Router#configure
Router(config)#domain name-server 192.168.1.111
Router(config)#domain name-server 192.168.1.2
Router(config)#commit
```

Verification

```
Router#show hosts
Default domain is cisco.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
```

Associated Commands

- domain name
- domain list
- domain name-server
- domain ipv4 host
- domain ipv6 host

File Transfer Services

File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP), remote copy protocol (rcp) rcp clients, and Secure Copy Protocol (SCP) are implemented as file systems or resource managers. For example, path names beginning with tftp:// are handled by the TFTP resource manager.

The file system interface uses URLs to specify the location of a file. URLs commonly specify files or locations on the WWW. However, on Cisco routers, URLs also specify the location of files on the router or remote file servers.

When a router crashes, it can be useful to obtain a copy of the entire memory contents of the router (called a core dump) for your technical support representative to use to identify the cause of the crash. SCP, FTP, TFTP, rcp can be used to save the core dump to a remote server.

FTP

File Transfer Protocol (FTP) is part of the TCP/IP protocol stack, which is used for transferring files between network nodes. FTP is defined in RFC 959.

Configuring a Router to Use FTP Connections

You can configure the router to use FTP connections for transferring files between systems on the network. You can set the following FTP characteristics:

- Passive-mode FTP

- Password
- IP address

Configuration Example

Enable the router to use FTP connections. Configure the software to use passive FTP connections, a password for anonymous users, and also specify the source IP address for FTP connections.

```
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
```

Running Configuration

```
Router#show running-config ftp client passive
ftp client passive

Router#show running-config ftp client anonymous-password xxxx
ftp client anonymous-password xxxx
Router#show running-config ftp client source-interface HundredGigE 0/0/0/0
```

Associated Commands

- ftp client passive
- ftp client anonymous-password
- ftp client source-interface

TFTP

Trivial File Transfer Protocol (TFTP) is a simplified version of FTP that allows files to be transferred from one computer to another over a network, usually without the use of client authentication (for example, username and password).

Configuring a Router to Use TFTP Connections

Configuration Example

Configure the router to use TFTP connections and set the IP address of the HundredGigE 0/0/0/0 as the source address for TFTP connections:

```
Router#configure
Router(config)#tftp client source-interface HundredGigE 0/0/0/0
Router(config)#commit
```

Running Configuration

```
Router#show running-config tftp client source-interface HundredGigE 0/0/0/0
tftp client source-interface HundredGigE 0/0/0/0
```

Verification

```
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
```

Associated Commands

- tftp client source-interface type
- show cinetd services

SCP

Secure Copy Protocol (SCP) is a file transfer protocol which provides a secure and authenticated method for transferring files. SCP relies on SSHv2 to transfer files from a remote location to a local location or from local location to a remote location.

Cisco IOS XR software supports SCP server and client operations. If a device receives an SCP request, the SSH server process spawns the SCP server process which interacts with the client. For each incoming SCP subsystem request, a new SCP server instance is spawned. If a device sends a file transfer request to a destination device, it acts as the client.

When a device starts an SSH connection to a remote host for file transfer, the remote device can either respond to the request in Source Mode or Sink Mode. In Source Mode, the device is the file source. It reads the file from its local directory and transfers the file to the intended destination. In Sink Mode, the device is the destination for the file to be transferred.

Using SCP, you can copy a file from the local device to a destination device or from a destination device to the local device.

Using SCP, you can only transfer individual files. You cannot transfer a file from a destination device to another destination device.

Transferring Files Using SCP

Secure Copy Protocol (SCP) allows you to transfer files between source and destination devices. You can transfer one file at a time. If the destination is a server, SSH server process must be running.

Configuration Example

Transfers the file "test123.txt" from the local directory to the remote directory.

```
Router#scp /harddisk:/test123.txt xyz@1.75.55.1:/auto/remote/test123.txt
Connecting to 1.75.55.1...
Password:
Router#commit
```

Verification

Verify if the file "test123.txt" is copied:

```
xyz-lnx-v1:/auto/remote> ls -altr test123.txt
-rw-r--r-- 1 xyz eng 0 Nov 23 09:46 test123.txt
```

Associated Commands

- scp

Cisco inetd

Cisco Internet services process daemon (Cinetd) is a multithreaded server process that is started by the system manager after the system has booted. Cinetd listens for Internet services such as Telnet service, TFTP service, and so on. Whether Cinetd listens for a specific service depends on the router configuration. For example, when the **tftp server** command is entered, Cinetd starts listening for the TFTP service. When a request arrives, Cinetd runs the server program associated with the service.

Note You must install Telnet RPM before using the Telnet service and the **show cinetd services** command.

```
RP/0/RP0/CPU0:ios#show cinetd services
Wed Aug 21 16:49:42.609 UTC

Vrf Name   Family  Service  Proto  Port  ACL   max_cnt   curr_cnt  wait   Program Client     Option

default    v4      telnet   tcp    23     10       0     nowait  telnetd   sysdb
```

Telnet

Enabling Telnet allows inbound Telnet connections into a networking device.

Prerequisites

Ensure to install Telnet RPM before using the Telnet service and **show cinetd services** command.

Configuration Example

Enable telnet and limit the number of simultaneous users that can access the router to 10.

```
Router# configure
Router(config)# telnet ipv4 server max-servers 10
Router(config)# commit
```

Verification

```
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
```

Syslog source-interface

You can configure the logging source interface to identify the syslog traffic, originating in a VRF from a particular router, as coming from a single device.

Configuration Example

Enable a source interface for the remote syslog server. Configure interface loopback 2 to be the logging source interface for the default vrf.

```
Router#configure
Router(config)#logging source-interface Loopback2

Router(config)#commit
```

Running Configuration

```
Router#show running-config logging
/*Logging configuration after changing the source into loopback2 interface.
logging console debugging
logging monitor debugging
logging facility local4
logging 123.100.100.189 vrf default severity info port default
logging source-interface Loopback2
```

Associated Commands

- logging source-interface
- show running-configuration logging

HTTP Client Application

HTTP Client allows files to be transferred from http server to another device over a network using HTTP protocol. You can configure http client and various parameters associated with it by using the **http client** command.

Configure HTTP Client

HTTP Client application is available by default. You can configure http client settings or view and modify the existing settings. To configure the settings, use the **http client** command in XR Config mode.

```
Router#configure
Router(config)#http client ?
connection          Configure HTTP Client connection
response            How long HTTP Client waits for a response from the server
                    for a request message before giving up
secure-verify-host  Verify that if server certificate is for the serverit is known as
secure-verify-peer  Verify authenticity of the peer's certificate
source-interface    Specify interface for source address
ssl                 SSL configuration to be used for HTTPS requests
tcp-window-scale    Set tcp window-scale factor for High Latency links
version             HTTP Version to be used in HTTP requests
vrf                 Name of vrf
```

Table 10: Commands used to configure HTTP Client settings

Features	Description
connection	Configure HTTP Client connection by using either retry or timeout options.
response	How long HTTP Client waits for a response from the server for a request message before giving up.
secure-verify-host	Verify host in peer's certificate. To disable verifying this, you can use the command **http client secure-verify-host disable**
secure-verify-peer	Verify authenticity of the peer's certificate.
source-interface	Specifies the interface for source address for all outgoing HTTP connections. You can enter either an ipv4 or ipv6 address or both.
ssl version	SSL version (configuration) to be used for HTTPS requests.
tcp-window-scale scale	Set tcp window-scale factor for high latency links.
version version	HTTP version to be used in HTTP requests. • 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 name	Name of vrf.

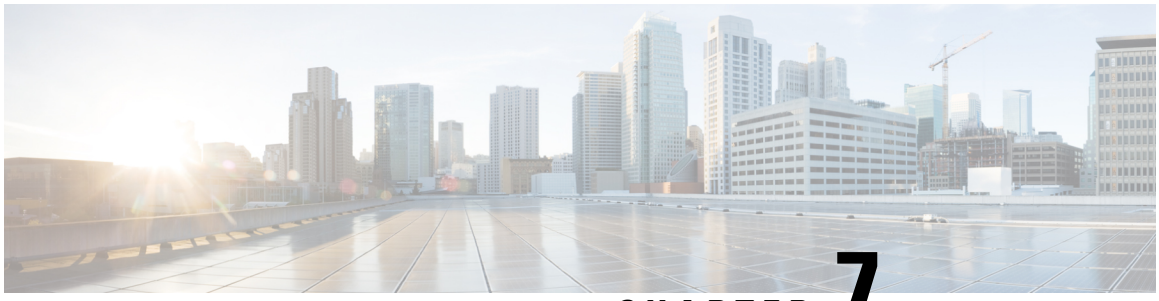
Configuration Examples

Example 1: This example shows how to set the tcp window-scale to 8.

```
Router(config)#http client tcp-window-scale 8
```

Example 2: This example shows how to set the HTTP version to 1.0.

```
Router(config)#http client version 1.0
```



CHAPTER 7

Implementing Access Lists

- Understanding Access Lists, on page 85
- IP Access List Entry Sequence Numbering, on page 90
- Applying Access Lists, on page 92
- Hybrid ACLs, on page 92
- Configuring IPv4 ACLs, on page 99
- Configuring IPv6 ACLs, on page 102
- TCP Flags in ACLs, on page 110
- Configuring Extended Access Lists, on page 113
- IPv4 and IPv6 ACL in Class Map, on page 114
- User-Defined TCAM Keys for IPv4 and IPv6, on page 115
- Modifying ACLs, on page 117
- ACLs on Bridge Virtual Interfaces, on page 119
- Configuring ACL-based Forwarding, on page 121
- Access Control List Counters, on page 127
- ACL Statistics Counter, on page 127
- Configuring ACLs with Fragment Control, on page 129
- Configuring ACL Filtering by IP Packet Length, on page 133
- Configuring TTL Matching, on page 136
- Understanding IP Access List Logging Messages, on page 136
- Per Interface Statistics, on page 140

Understanding Access Lists

Access lists perform packet filtering to control which packets move through the network and where. Such controls help to limit network traffic and restrict the access of users and devices to the network. Access lists have many uses, and therefore many commands accept a reference to an access list in their command syntax. Access lists can be used to do the following:

An access control list (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 2 types of ACLs:

- Standard ACLs- Verifies 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- Verifies 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.

Purpose of IP Access Lists

- 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.

How an IP Access List Works

An access list is a sequential list consisting of permit and deny statements that apply to IP addresses and possibly upper-layer IP protocols. The access list has a name by which it is referenced. Many software commands accept an access list as part of their syntax.

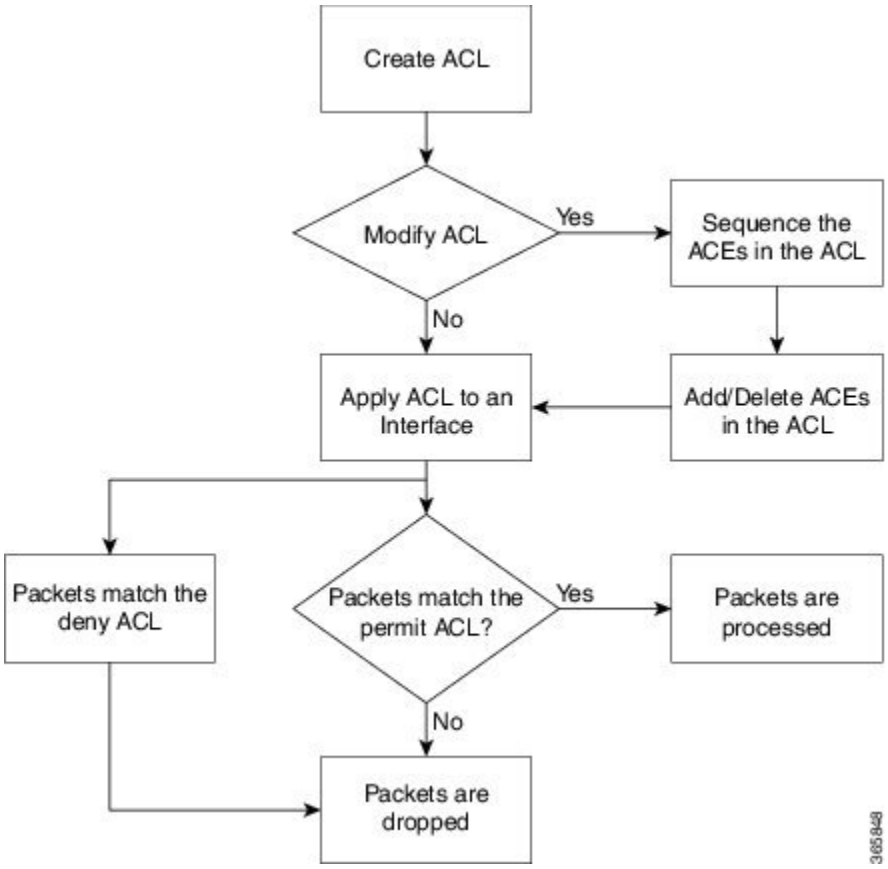
An access list can be configured and named, but it is not in effect until the access list is referenced by a command that accepts an access list. Multiple commands can reference the same access list. An access list can control traffic arriving at the router or leaving the router, but not traffic originating at the router.

Source address and destination addresses are two of the most typical fields in an IP packet on which to base an access list. Specify source addresses to control packets from certain networking devices or hosts. Specify destination addresses to control packets being sent to certain networking devices or hosts.

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.

ACL Workflow

The following image illustrates the workflow of an ACL.



IP Access List Process and Rules

Use the following process and rules when configuring an IP access list:

- 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

Consider the following when creating an IP access list:

- Create the access list before applying it to an interface.
- Organize your access list 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.

Source and Destination Addresses

Source address and destination addresses are two of the most typical fields in an IP packet on which to base an access list. Specify source addresses to control packets from certain networking devices or hosts. Specify destination addresses to control packets being sent to certain networking devices or hosts.

ACL Filtering by Wildcard Mask and Implicit Wildcard Mask

Address filtering uses wildcard masking to indicate 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. By carefully setting wildcard masks, an administrator can select a single or several IP addresses for permit or deny tests.

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.

- A wildcard mask bit 0 means *check* the corresponding bit value.

- A wildcard mask bit 1 means *ignore* that corresponding bit value.

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 also use Classless Inter-Domain Routing (CIDR) format (/x) in place of 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.

Transport Layer Information

You can filter packets on the basis of transport layer information, such as whether the packet is a TCP, UDP, ICMP, or IGMP packet.

Guidelines and Restrictions for Configuring Access Lists

You must be aware of the following restrictions for configuring access lists.

- IPv4 and IPv6 ACLs are not supported for loopback and interflex interfaces.
- 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.
- Filtering of Multiprotocol Label Switching (MPLS) packets through interface ACL is not supported.
- 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.

Including Comments in Access Lists

You can include comments (remarks) about entries in any named IP access list using the remark access list configuration command. The remarks make the access list 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.

IP Access List Entry Sequence Numbering

The ability to apply sequence numbers to IP access-list entries simplifies access list changes. Prior to this feature, there was no way to specify the position of an entry within an access list. If a user wanted to insert an entry (statement) in the middle of an existing list, all the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

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 11: Feature History Table

Sequence Numbering Behavior

The following details 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.

Adding Entries with Sequence Numbers: Example

In the following example, an 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
```

Adding Entries Without Sequence Numbers: Example

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
```

Applying Access Lists

After you create an access list, you must reference the access list to make it work. Access lists 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.

For inbound access lists, after receiving a packet, Cisco IOS XR software checks the source address of the packet against the access list. If the access list permits the address, the software continues to process the packet. If the access list rejects the address, the software discards the packet and returns an ICMP host unreachable message. The ICMP message is configurable.

For outbound access lists, after receiving and routing a packet to a controlled interface, the software checks the source address of the packet against the access list. If the access list permits the address, the software sends the packet. If the access list rejects the address, the software discards the packet and returns an ICMP host unreachable message.

When you apply an access list that has not yet been defined to an interface, the software acts as if the access list has not been applied to the interface and accepts all packets. Note this behavior if you use undefined access lists as a means of security in your network.

Hybrid ACLs

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 12: Feature History Table

Feature Name	Release Information	Feature Description
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.
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. The commands modified are: • ipv4 access-group • ipv6 access-group

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 25.2.1, 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 Object-Group ACLs

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.

Compressing ACLs

Object-group ACLs use compression to accommodate the large number of ACEs. Compression is achieved by compressing the following two fields of an ACE:

- Source IP prefix
- Destination IP prefix

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 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.

Restrictions for egress hybrid ACLs

These are the restrictions for 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.

Configuring an Object-Group ACL

Before You Begin

You must be aware of the following information that apply to object-group ACLs:

- You can configure ACLs that contain both conventional and object-group 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 object-group ACL multiple times with a source group, or a destination group, or both source and destination groups.

Restrictions

Configuring object-group ACLs involves the following restrictions:

- Object-group ACLs can only be configured to an interface. They cannot be used or referenced by applications like SSH, SNMP, NTP.
- To delete an object-group, you must first delete it from all ACLs.
- You cannot configure object-group ACLs along with QoS policies.

- Object-group ACLs are not supported in any policy based configuration.
- Any inline ACE update to an object group ACL clears complete stats of the ACL.

Configuring a Network Object-Group ACL

A network object group can contain a single or multiple network objects.

Starting 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). Thus, improving network traffic security, also allowing improved usage of space and resource to accommodate more ACLs.

Restriction

Compression level 2 only supports prefix masks /n, not arbitrary address masks a.b.c.d.

Configuring a Network Object-Group ACL for Compress Level 2

Use the following set of configuration statements to configure a network object-group ACL for an IPv4 address.

```
/* From the global configuration mode, create a network object group. */
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

/* Create an access list referencing the object group. */
Router(config)# ipv4 access-list network-object-acl permit ipv4 net-group netobj1 any

/* Apply the access list containing the object group to the desired interface and commit
your configuration. */
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
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
```

Use the following set of configuration statements to configure a network object-group ACL for an IPv6 address.

```
/* From the global configuration mode, create a network object group. */
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

/* Create an access list referencing the object group. */
```

```
Router(config)# ipv6 access-list network-object-acl permit ipv6 net-group netobj1 any

/* Apply the access list containing the object group to the desired interface and commit
your configuration. */
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
Tue Mar 28 10:23:34.106 IST

Router(config)# interface HundredGigE 0/0/10/3, changed state to Down
Router(config)# interface HundredGigE 0/0/10/3, changed state to Up

Router(config-if)# exit
```

Running Configuration

Validate your configuration by using the **show run** command.

```
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 object-group ACL for compression level 2.

Configuring a Port Object-Group ACL

A port object-group can contain a single or multiple port objects.

Restriction

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.

Configuration

Use the following set of configuration statements to configure a port object-group ACL.

```
/* From the global configuration mode, create a port object group, and commit your
configuration. */
```

```
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

/* Create an access list referencing the object group. */
Router(config)# ipv4 access-list port-object-acl permit ipv4 net-group portobj1

/* Apply the access list containing the object group to the desired interface and commit
your configuration. */
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)# (config-if)# exit
```

Running Configuration

Confirm your configuration.

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

Verifying Object-Group ACL Compression

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.

Verification

Use the following set of verification commands to verify object-group ACL compression.

```
/* Verify the entries of the ACL in operation. */

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)
...

/* Verify the ACE compression in the ACL. */
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.

Configuring IPv4 ACLs

This section describes the basic configuration of IPv4 ingress and egress ACLs.

Notes and Restrictions for Configuring IPv4 Ingress ACLs

IPv4 ingress ACLs are characterized by the following behavior for Cisco 8000 Series Routers. These restrictions are subject to change with respect to other platforms.

- Ingress IPv4 ACLs are supported on all interfaces except management interfaces.
- In Fixed system, maxmium number of ACLs allowed per NPU is 45, In distributed system, maxmium number of ACLs allowed per NPU is 30.
- Packet Length is not supported.
- ACL logging with input interface (using the **log-input** keyword) is not supported.
- From Release 7.8.1 onwards, ACL logging with input interface (using the **log-input** keyword) is supported.

Notes and Restrictions for Configuring IPv4 Egress ACLs

IPv4 egress ACLs are characterized by the following behavior.

- ACL is not supported on Management interface on egress direction.
- ACL logging is not supported on egress direction.

Configuring an Ingress IPv4 ACL on a HundredGigE Interface

Use the following configuration to configure an ingress IPv4 ACL on a HundredGigE interface.

```
/* Configure a HundredGigE interface with an IPv4 address */
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
Thu Jul 11 08:46:51.930 UTC
Router(config-if)# exit

/* Verify if the interface is up */
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

/* Configure an IPv4 ingress ACL */
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
Thu Jul 11 08:55:12.806 UTC

/* Verify the ingress ACL creation */
Router(config)# do show access-lists ipv4
Thu Jul 11 08:55:44.824 UTC
```

```
...
ipv4 access-list V4-ACL-INGRESS
 10 permit tcp 192.0.2.2 255.255.255.0 any
 20 deny udp any any
 30 permit ipv4 192.0.2.64 255.255.255.0 any


/* 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
Thu Jul 11 09:01:26.744 UTC
Router(config-if)# exit

/* Verify if the ingress ACL has been successfully applied to the interface */
Router(config)# do show ipv4 interface
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
```

You have successfully configured an IPv4 ingress ACL on a HundredGigE interface.

Configuring an Egress IPv4 ACL on a HundredGigE Interface

Use the following configuration to configure an egress IPv4 ACL on a HundredGigE interface.

```
/* Configure a HundredGigE interface with an IPv4 address */
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
Thu Jul 11 08:55:12.806 UTC
Router(config-if)# exit

/* Verify if the interface is up */
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

/* Configure an IPv4 egress ACL */
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
Thu Jul 11 08:59:10.093 UTC

/* Verify the egress ACL creation */
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
...

/* 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
Thu Jul 11 09:19:49.569 UTC
Router(config-if)# exit

/* Verify if the egress ACL has been successfully applied to the interface */
Router(config)# do show ipv4 interface
Thu Jul 11 09:01:50.445 UTC
HundredGigE 0/0/0/1 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 198.51.100/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is V4-ACL-EGRESS
  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
...
```

You have successfully configured an IPv4 egress ACL on a HundredGigE interface. For more information on logging messages, see Understanding IP Access List Logging Messages, on page 136.

Configuring IPv6 ACLs

Table 13: Feature History Table

Feature Name	Release Information	Feature Description
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)

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.

Restrictions and Guidelines

The following restrictions and guidelines apply while configuring IPv6 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.

- 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.

Configuring an Ingress IPv6 ACL on a HundredGigE Interface

Use the following configuration to configure an ingress IPv6 ACL on a HundredGigE interface.

```
/* Configure a HundredGigE interface with an IPv6 address */
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
Thu Jul 11 09:28:07.759 UTC
Router(config-if)# exit

/* Verify if the interface is up */
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
…

/* Configure an IPv6 ingress ACL */
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
Thu Jul 11 09:41:02.625 UTC
Router(config-ipv6-acl)# exit


/* Verify the ingress ACL creation */
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


/* 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
Thu Jul 11 09:43:59.733 UTC
Router(config-if)# exit

/* Verify if the ingress ACL has been successfully applied to the interface */
Router(config)# do show ipv6 interface
Thu Jan 25 11:34:08.028 IST
HundredGigE 0/0/0/0 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::bd:b9ff:fea9:5606
  Global unicast address(es):
    2001::1, subnet is 1001::/64
  Joined group address(es): ff02::1:ff00:1 ff02::1:ffa9:5606 ff02::2
      ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
```

```
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is V6-INGRESS-ACL
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
…
```

You have successfully configured an IPv6 ingress ACL on a HundredGigE interface.

Configuring an Egress IPv6 ACL on a HundredGigE Interface

Use the following configuration steps to configure an egress IPv6 ACL on a HundredGigE interface.

```
/* Configure a HundredGigE interface with an IPv6 address */
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
Thu Jan 25 11:41:25.778 IST
Router(config-if)# exit

/* Verify if the interface is up */
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

/* Configure an IPv6 egress ACL */
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
Thu Jul 11 09:50:40.566 UTC
Router(config-ipv6-acl)# exit

/* Verify the egress ACL creation */
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
…

/* 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
Thu Jul 11 09:52:57.751 UTC
Router(config-if)# exit

/* Verify if the egress ACL has been successfully applied to the interface */
```

```
Router(config)# do show ipv6 interface
Thu Jul 11 09:53:41.365 UTC
…
HundredGigE is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::23:e9ff:fea8:a44e
  Global unicast address(es):
    2001::1, subnet is 2001::/64
  Joined group address(es): ff02::1:ff00:1 ff02::1:ffa8:a44e ff02::2
      ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
   Hosts use stateless autoconfig for addresses.
  Outgoing access list is V6-EGRESS-ACL
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
...
```

You have successfully configured an IPv6 egress ACL on a HundredGigE interface.

Configuring Ingress and Egress IPv6 ACLs on Bundle Interfaces

Use the following configuration to configure ingress and egress IPv6 ACLs on a bundle interface.

```
/* Configure a bundle interface with an IPv6 address */
Router(config)# interface Bundle-Ether 1
Router(config-if)# ipv6 address 2001::1/64
Router(config-if)# no shut
Router(config-if)# commit
Thu Jul 11 09:56:40.603 UTC
Router(config-if)# exit

/* Configure an IPv6 egress ACL */
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
Thu Jul 11 10:02:34.568 UTC
Router(config-ipv6-acl)# exit

/* Configure an IPv6 ingress ACL to deny ingress traffic on the bundle interface */
Router(config)# ipv6 access-list V6-DENY-INGRESS-ACL
Router(config-ipv6-acl)# 10 deny ipv6 any any
Router(config-ipv6-acl)# commit
Thu Jul 11 10:03:43.411 UTC
Router(config-ipv6-acl)# exit

/* Verify the egress and ingress ACL creation */
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
...

/* 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
Thu Jul 11 10:06:06.452 UTC
Router(config-if)# exit

/* Verify if the ACLs have been successfully applied to the interface */
Router(config)# do show ipv6 interface
Thu Jul 11 10:06:49.975 UTC
…
Bundle-Ether1 is Down, ipv6 protocol is Down, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::1:10ff:fe87:8d04 [TENTATIVE]
  Global unicast address(es):
    2001::1, subnet is 2001::/64 [TENTATIVE]
  Joined group address(es): ff02::2 ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 160 to 240 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is V6-EGRESS-ACL-BI
  Inbound  common access list is not set, access list is V6-DENY-INGRESS-ACL
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

You have successfully configured ingress and egress IPv6 ACLs on a bundle interface.

Configuring an Ingress IPv6 ACL with Hop Limit on a Bundle Interface

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.

To configure an ingress IPv6 ACL with hop limit on a bundle interface, use the following configuration steps:

1. Configure a bundle interface with an IPv6 address.

2. In the interface configuration mode, create an object group with a list of hosts.
3. Apply the ACL definition to filter IPv6 packets based on hop limit match.
4. Associate the ACL configuration on to the ingress interface to match based on IPv6 hop limit.

Configuration Example

The following example shows how to set hop limit on an IPv6 ACL. In this 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.

Note Hop limit is referred to as **ttl**.

```
/* Configure a bundle interface with an IPv6 address */
Router(config)# interface Bundle-Ether 1
Router(config-if)# ipv6 address 2001::1/64
Router(config-if)# no shut
Router(config-if)# commit
Thu Jul 11 09:56:40.603 UTC
Router(config-if)# exit

/* Create an object group with the list of host */

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
Thu Jul 11 09:41:02.625 UTC
Router(config-object-group-ipv6)# exit

/* Apply ACL definition to filter packets based on hop-limit match on IPv6.
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
!

/* Associate the ACL configuration on to 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
!
```

Running Configuration

The following example shows the running configuration of an IPv6 ingress ACL with hop limit (ttl) on a bundle interface.

```
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>
```

Verification

Verify that IPv6 ingress ACL with hop limit is set on a bundle interface:

```
/* Verify the ingress ACL creation */
Router(config)# show running-config | i hw
Fri Nov 4 06:26:27.382 UTC
Building configuration...
hw-module profile stats acl-permit

/* Verify the ACL hit count for the ACE */
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)

/* Verify the ACL logging */
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.

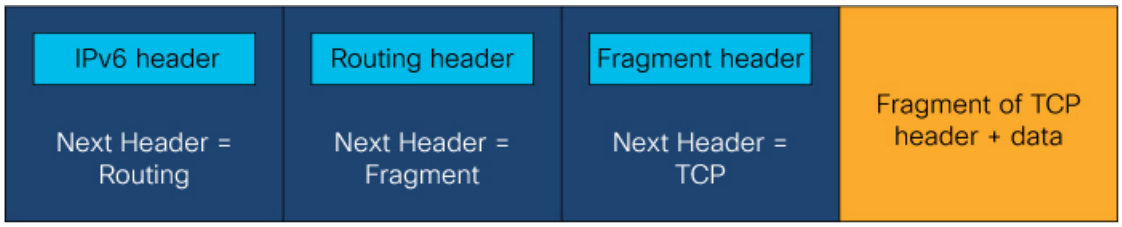
For more information on logging messages, see Understanding IP Access List Logging Messages, on page 136.

Configuring IPv6 Extensions Headers in Ingress IPv6 Hybrid ACLs

Table 14: Feature History Table

Feature Name	Release Information	Description
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.

Figure 12: IPv6 Extension Header in an IPv6 Packet



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.

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.

Restrictions

- 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.

Configuring Ingress IPv6 Hybrid ACLs Based on Extensions Headers

This following example shows you how to create an ingress IPv6 Hybrid ACL with compression level 2 based on IPv6 extensions headers:

```
/* Enter the global configuration mode and create an IPv6 Hybrid ACL with an ACE that has
routing extension header. */
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

/* Enter the interface configuration mode and configure an ingress interface with the Hybrid
 ACL twith compression level 2. */
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
```

TCP Flags in ACLs

Table 15: Feature History Table

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

Feature Name	Release Information	Description
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)

The transmission Control Protocol (TCP) is one of the most widely used protocol 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. The various TCP flags are SYN, ACK, FIN, RST, URG, PSH, and EST. The following table provides information about these flags:

You can create ACEs that permit or deny packets based on TCP flags. By using ACLs:

- You can create ACEs that filter packets based on whether a packet has a TCP flag set or not.
- You can filter packets based on the presence or absence of any one TCP flag or combination of multiple TCP flags.

Therefore, ACLs based on TCP flags provide increased flexibility to filter packets and provides enhanced security. For example, ACLs can permit packets that have a SYN flag to ensure that the packets have a verified source.

TCP Flags

The following TCP flags can be present in a packet:

- SYN: Both the sender and receiver devices use the synchronisation (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 aSYN 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.

Configuring ACLs Based on TCP Flags

You can use the **match-any** keyword in ACLs to permit or deny packets based on whether any of the configured TCP flags is set.

This following example shows you how to create an ACL to permit packets that have either the PSH or URG TCP flags set:

```
/* Enter the global configuration mode and create an ACL with name ACL-TCP.*/
Router# configure
Router(config)# ipv4 access-list ACL-TCP

/* Configure an ACL that permits packets with TCP flag that is either PSH or URG */
Router(config-ipv4-acl)# 10 permit tcp any any match-any + urg + psh
Router(config-ipv4-acl)# commit
```

This following example shows you how to create an ACL to permit packets that have either the SYN or ACK TCP flag set:

```
/* Enter the global configuration mode and create an ACL with name ACL-TCP. */
Router# configure
Router(config)# ipv4 access-list ACL-TCP

/* Configure an ACL that permits packets with both SYN and ACK TCP flags set. */
Router(config-ipv4-acl)# 10 permit tcp any any match-any + syn + ack
Router(config-ipv4-acl)# commit
```

This following example shows you how to create an ACL to permit packets that have the SYN flag set or the ACK flag not set:

```
/* Enter the global configuration mode and create an ACL with name ACL-TCP.*/
Router# configure
Router(config)# ipv4 access-list ACL-TCP

/* Configure an ACL that permits packets with the SYN flag set and the ACK flag not set.
*/
Router(config-ipv4-acl)# 10 permit tcp any any match-any + syn - ack
Router(config-ipv4-acl)# commit
```

You can use the **match-all** keyword in ACLs to permit or deny packets based on whether all the configured TCP flags are set.

This following example shows you how to create an ACL to permit packets that have both the URG flag and FIN flag set:

```
/* Enter the global configuration mode and create an ACL with name ACL-TCP.*/
Router# configure
Router(config)# ipv4 access-list ACL-TCP

/* Configure an ACL that permits packets with the SYN flag set and the ACK flag not set.
*/
Router(config-ipv4-acl)# 10 permit tcp any any match-all + urg + fin
Router(config-ipv4-acl)# commit
```

This following example shows you how to create an ACL to permit packets that have both the SYN flag set and the ACK flag not set:

```
/* Enter the global configuration mode and create an ACL with name ACL-TCP.*/
Router# configure
Router(config)# ipv4 access-list ACL-TCP

/* Configure an ACL that permits packets with the SYN flag set and the ACK flag not set.
*/
Router(config-ipv4-acl)# 10 permit tcp any any match-all + syn - ack
Router(config-ipv4-acl)# commit
```

Configuring Extended Access Lists

Use Extended Access Lists to 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.

Configuration Example

To configure Extended Access Lists, you must completed create an access list and specify the condition to allow or deny the network traffic.

```
/* Enter the global configuration mode and create the access list*/
Router# configure
Router(config)# ipv4 access-list acl_1
Router(config-ipv4-acl)# 10 remark Do not allow user1 to telnet out
/*Specify the condition to allow or deny the network traffic.*/
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
```

Running Configuration

```
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
!
```

Verification

```
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
```

IPv4 and IPv6 ACL in Class Map

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

Configuring IPv6 ACL QoS - An Example

This example shows how to configure IPv6 ACL QoS with IPv4 ACL and other fields :

```
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
!

ipv4 access-list aclv4
10 permit ipv4 host 10.6.10.2 host 10.7.10.2
!

class-map match-any c.aclv6
match access-group ipv6 aclv6
match access-group ipv4 aclv4
match cos 1
```

```
end-class-map
!

policy-map p.aclv6
class c.aclv6
  set precedence 3
!
class class-default
!
end-policy-map
!
```

User-Defined TCAM Keys for IPv4 and IPv6

Access-lists use a TCAM (internal and external) to perform the lookup and action resolution on each packet. The TCAM is a valuable and constrained resource in hardware, which must be shared by multiple features. Therefore, the space (key width) available for these key definitions is also constrained. A key definition specifies which qualifier and action fields are available to the ACL feature when performing the lookup.

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 115.

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 be ignored for the line card.

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] field1field2
```

Note It is recommeded to use global UDK for Cisco 8000 Series Routers Fixed platform.

IPv4 and IPv6 Key Formats

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 16: 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
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.

The following table shows the action fields supported in the IPv4 and IPv6 key formats.

Table 17: 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)

Modifying ACLs

This section describes a sample configuration for modification of ACLs.

```
*/ 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

*/Add a new entry with one with a sequence number "15" to the ACL. Delete an entry with the
 sequence number "30":*/
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*/

*/ Add another entry without a sequence number to the ACL */
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
 will have a sequence
number 10 higher than the last entry in the existing access list*/
```

Note While ACLs allow configuration without sequence numbers, we **recommend** you to 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. See the example below for further clarification.

```
*/Modify an existing entry '10' in the ACL using the sequence number */
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.

ACLs on Bridge Virtual Interfaces

Table 18: Feature History Table

Feature Name	Release Information	Description
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.

Bridge Virtual Interfaces (BVIs) provide a bridge between the routing and bridging domains on a router. You can configure BVI with an IP address that operates as a regular routed interface. You can configure an ACL on a BVI to filter the traffic for the network that uses the interface.

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

You must be aware of the following restrictions before proceeding to configure ACLs on BVIs.

- When you apply a security ACL on a BVI interface, TCAMs are programmed on all NPUs of all linecards. 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 linecard of the L3 interface.

Configuration

The following section describes the procedure for configuring IPv4 ingress and egress ACLs on BVIs.

To configure IPv4 ingress and egress ACLs on a BVI, use the following procedure with a sample configuration.

1. Enter the Global Configuration mode, and configure an IPv4 ingress ACL.

```
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.

```
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 must to the BVI, and enable it for Layer 2 transport.

```
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.

```
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.

```
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.

```
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.

```
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.

Configuring ACL-based Forwarding

Table 19: Feature History Table

Feature Name	Release Information	Description
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**

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. This is achieved by specifying the next-hop address in ACL configurations, so that the configured next-hop address from ACL is used for forwarding packet towards its destination instead of routing through packet-based destination address lookup. This feature of using next-hop in ACL configurations for forwarding is called ACL Based Forwarding (ABF).

ACL-based forwarding enables you to choose service from multiple providers for broadcast TV over IP, IP telephony, data, and so on, which provides a cafeteria-like access to the Internet. Service providers can divert user traffic to various content providers.

Restrictions

- Traffic outages can occur during transitions from an existing nexthop to another nexthop.
- IPv4 and IPv6 ABF nexthops routed over GRE interfaces are not supported.
- ACL Based Forwarding(ABF) is not supported over BVI.

Feature Highlights

- ABF supports nexthop modifications. You can modify a nexthop, remove a nexthop, or make changes between existing nexthops.

Note While defining an ACE rule, you must specify the VRF for all nexthops unless the nexthop is in the default VRF. This process ensures that the packets take the right path towards the nexthop.

- As ABF is ACL-based, packets that do not match an existing rule (ACE) in the ACL are subjected to the default ACL rule (drop all). If the ACL is being used for ABF-redirect only (not for security), then 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 as normal.
- ABF is supported on permit rules only.
- ABF default route is not supported.
- Packets punted in the ingress direction from the NPU to the linecard CPU are not subjected to ABF treatment due to lack of ABF support in the slow path. These packets will be forwarded normally based on destination-address lookup by the software dataplane. Some examples of these types of packets are (but are not limited to) packets with IPv4 options, IPv6 extension headers, and packets destined for glean (unresolved/incomplete) adjacencies.
- Starting with Cisco IOS XR Release 7.3.2, ABF supports logging of permit statistics. This feature keeps track of 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 prior to configuring the ACL based forwarding itself. 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 enabling hardware module statistics for ACL, based on the requirement, you must reboot the line cards or the router.

The following snippet shows you how to configure logging of permit statistics:

```
Router# configure
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
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)#
```

You can use the show access-lists ipv4 or show access-lists ipv6 command to view the ABF statistics.

The following snippet shows the ABF statistics for a IPv4 ACL in a selected interface:

```
Router#sh 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)
```

Configuration Example

To configure ACL-based forwarding for IPv4 packets, use the following steps:

1. Enter IPv4 access list configuration mode and configure an ACL.
2. Set the conditions for the ACL.
3. Configure nexthop addresses for ABF.

Configuration

To configure ACL-based forwarding for IPv4 packets, use the following configuration example:

```
/* Enter IPv4 access list configuration mode and configure an ACL: */
Router# configure
Router(config)# ipv4 access-list abf-acl

/* Set the conditions for the ACL and configure ABF: */
/* The next hop for this entry is specified. */
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
/* More than two nexthops */
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

/* VRF support on ABF */
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
```

To configure ACL-based forwarding for IPv6 packets, use the following configuration example:

```
/* Enter IPv6 access list configuration mode and configure an ACL: */
Router# configure
Router(config)# ipv6 access-list abf-acl

/* Set the conditions for the ACL and configure ABF: */
/* The next hop for this entry is specified. */
Router(config-ipv6-acl)# 10 permit ipv6 2001:db8::/32 any nexthop1 ipv6 2001:db8::2

/* More than two nexthops */
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

/* VRF support on ABF */
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
```

Running Configuration

```
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
```

Verification

Use the following command to verify the IP nexthop state in ABF to ensure that the expected nexthop is up:

```
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
```

Use the following command to verify if ABF is currently attached to any interfaces at any linecard:

```
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

Table 20: Feature History Table

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 income traffic matching a ACEs in the ABF to a VRF network and helps in load balancing and cost saving.

Starting with Cisco IOS XR Release 7.3.3, ACL-based Forwarding for IPv4 and IPv6 addresses supports virtual routing and forwarding (VRF) redirect. With this, you could configure the ABF ACEs to forward incoming traffic matching a ACEs in the ABF packets to a VRF by using it as a next-hop. ABF supports both VRF-select and VRF-aware redirect. In VRF-select, you could configure only a VRF as the next-hop, and in VRF-aware, you could configure both VRF and IP address as the next-hop. The ABF VRF redirect supports three next-hops per ACE.

When an incoming packet matches the permit ACE with VRF redirect, the router forwards that traffic to the first available VRF specified in the ACE. Then, the traffic injected to VRF is further processed by matching it against the routes available in the VRF. If the redirected traffic fails to match any routes available in VRF, a default route redirects the traffic back to the global routing table for lookup and forwarding. In any ACEs with VRF, if the VRFs are down, then the traffic matching such ACEs are dropped immediately.

The following example shows how to configure ABF with VRF-select:

```
Router# configure
Router(config)# ipv4 access-list abf-vrf-select
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
Router(config-ipv4-acl)# commit
```

The following example shows how to configure ABF with VRF-aware:

```
Router# configure
Router(config)# ipv4 access-list abf-vrf-aware
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
Router(config-ipv4-acl)# commit
```

The following commands display ABF-VRF configurations in the router:

```
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


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
```

Associated Commands

- ipv4 access-list
- ipv6 access-list
- show access-lists ipv4
- show access-lists ipv6

Associated Topics

- Configuring IPv4 ACLs
- Configuring IPv6 ACLs

Access Control List Counters

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

The ACL statistics counter feature allows you to track the count of packets that a router either permits or denies. A router permits or denies packets based on the ACL rules that you configure on a router interface. By default, the ACL statistics counter allows you to track only the the count of packets denied. By configuring

the command, **hw-module profile stats acl-permit**, 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. Support for ACL permit counters also allows you to track the ACE through which a router permits a packet.

Restrictions

- After you configure the **hw-module profile stats acl-permit** command on the router, based on the requirement, you must reload the router or the line cards. Configuring of the command followed by reloading the router or line cards enables the tracking of the permitted packet count on the router or line cards.
- If the hardware module statistics for ACL (**hw-module profile stats acl-permit**) is enabled, then the flowspec drop counter functionality will not work for the NP1-based platforms.

Configuration Example

To enable the tracking of the permitted packet count based on the ACL rules, use the following steps:

1. Enter global configuration mode and configure an ACL.
2. Enter interface configuration mode and attach the configured ACL on an interface.
3. Enable the tracking of the permitted packet count.
4. Based on the requirement, reload the router or line cards.

Configuration

```
/* Enter global configuration mode and configure an ACL. */
Router# configure
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)# exit

/* Enter the interface configuration mode and attach the configured ACL on an interface.
*/
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv4 access-group TEST ingress
Router(config-if)# commit
Router(config-if)# exit

/* Enable the tracking of the permitted packet count. */
Router(config)# hw-module profile stats acl-permit
Router(config)# commit

/* Reload the line cards. */
Router# reload location 0/0/CPU0
```

Running Configuration

```
Router# show running-configuration
ipv4 access-list TEST
10 permit ipv4 any any
20 deny udp any any
!
interface HundredGigE 0/0/0/0
ipv4 access-group TEST ingress
```

```
!
hw-module profile stats acl-permit
end
```

Verification

Use the **show access-lists ipv4** *acl-name* **hardware ingress location** *node-id* command to verify that you have enabled or disabled the tracking of the permitted packet count.

```
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)
```

Configuring ACLs with Fragment Control

The non-fragmented packets and the initial fragments of a packet were processed by IP extended access lists (if you apply this access list), but non-initial fragments were permitted, by default. However, now, the IP Extended Access Lists with Fragment Control feature allows more granularity of control over non-initial fragments of a packet. Using this feature, you can specify whether the system examines non-initial IP fragments of packets when applying an IP extended access list.

As non-initial fragments contain only Layer 3 information, these access-list entries containing only Layer 3 information, can now be applied to non-initial fragments also. The fragment has all the information the system requires to filter, so the access-list entry is applied to the fragments of a packet.

This feature adds the optional **fragments** keyword to the following IP access list commands: **deny** and **permit** . By specifying the **fragments** keyword in an access-list entry, that particular access-list entry applies only to non-initial fragments of packets; the fragment is either permitted or denied accordingly.

The behavior of access-list entries regarding the presence or absence of the **fragments** keyword can be summarized as follows:

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 3 and 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, and
	  • 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 will be ignored for the non-initial fragments.

You should 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 will 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. Therefore, you may need two access list entries for every deny entry. The first deny entry of the pair will not include the **fragments** keyword, and applies to the initial fragment. The second deny entry of the pair will include the **fragments** keyword and applies to the subsequent fragments. In the cases where 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.

Note The **fragments** cannot be configured for an access-list entry that contains any Layer 4 information.

Note Within the scope of ACL processing, Layer 3 information refers to fields located within the IPv4 header; for example, source, destination, 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, type and code for ICMP.

Configuring an IPv4 ACL to Match on Fragment Type

Most DoS (Denial of Service) attacks work by flooding the network with fragmented packets. By filtering the incoming fragments of the packet in a network, an extra layer of protection can be added 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. You can use the following sample configuration with the different fragment options:

```
/* Enter the global configuraton mode and configure an IPv4 access list */
Router# config
Router(config)# ipv4 access-list TEST
Router(config-ipv4-acl)# 10 permit tcp any any

/* Configure an ACE to match on the dont-fragment flag (indicates a non-fragmented packet)
 and forward the packet to the default (pre-configured) next hop  */
Router(config-ipv4-acl)# 20 permit tcp any any fragment-type dont-fragment default

/* Configure an ACE to match on the is-fragment flag (indicates a fragmented packet)
 and forward the packet to a next hop of 10.10.10.1  */
Router(config-ipv4-acl)# 30 permit udp any any fragment-type is-fragment nexthop1 ipv4
10.10.10.1

/* Configure an ACE to match on the first-fragment flag (indicates the first fragment of a
 fragmented packet)
 and forward the packet to a next hop of 20.20.20.1  */
Router(config-ipv4-acl)# 40 permit ospf any any fragment-type first-fragment nexthop1 ipv4
 20.20.20.1


/* Configure an ACE to match on the last-fragment flag (indicates the last fragment of a
fragmented packet)
 and forward the packet to a next hop of 30.30.30.1  */
Router(config-ipv4-acl)# 50 permit icmp any any fragment-type last-fragment nexthop1 ipv4
30.30.30.1
Router(config-ipv4-acl)# commit
```

Use Case: Configuring an IPv4 ACL to Match on the First Fragment and Last Fragment

This section describes an use case, where you configure an ACL to forward a fragment if it is the first fragment of the packet and discard a fragment if it is the last fragment of the packet.

In this configuration, 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.

```
/* Enter the global configuraton mode and configure an IPv4 access list */
Router# config
Thu Jan 11 11:56:27.221 IST
Router(config)# ipv4 access-list ACLFIRSTFRAG

/* Configure an ACE to match on the first fragment.
If the fragment offset value equals 0, the fragment is forwarded to the 192.168.1.2 next
hop */
Router(config-ipv4-acl)# 10 permit tcp any any fragment-type first-fragment nexthop1 ipv4
192.168.1.2

/* Configure an ACE to match on the last fragment, and drop the fragment at the interface.
 */
Router(config-ipv4-acl)# 20 deny tcp any any fragment-type last-fragment
Router(config-ipv4-acl)# commit
Thu Jan 11 12:01:33.297 IST

/* Validate the configuration */
Router(config-ipv4-acl)# do show access-lists
Thu Jan 11 12:05:23.646 IST
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.

Matching by Fragment Offset in ACLs

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 respectively at the interface. Fragment-offset filtering is supported only on ingress direction with compression mode of an ACL.

Note

- 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.

Configuring ACL Matching by Fragment Offset

To configure fragment-offset match in ACL, use the **fragment-offset** option in **permit** or **deny** command in IPv4 or IPv6 access-list configuration mode.

Note For fragment-offset filtering, you must attach the particular ACL to an interface with compression level 2. Else, the configuration is rejected.

Configuration

This example shows how to specify an ACL rule based on the fragment-offset per IPv4 header. Here, the packet is permitted only if the fragment-offset in the IPv4 header of the packet is within the range of 300-400. The value *300-400* is based on the 8-byte unit, which is same as fragment-offset of *2400-3200* bytes.

Running Configuration

```
ipv4 access-list fragment-offset-acl
 10 permit ipv4 any any fragment-offset range 300 400
!
```

Verify Fragment-offset Match in ACL

```
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
```

Configuring ACL Filtering by IP Packet Length

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 respectively at the interface.

To configure packet length filtering in ACL, use the **packet-length** option in **permit** or **deny** command in IPv4 or IPv6 access-list configuration mode.

Restrictions

Packet length filtering feature in ACL is subjected to these restrictions:

- Packet length filtering is supported only on 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.

Configuring Scaled IPv4 ACLs to Filter by Packet Length

To configure a scaled ACL to filter by packet length in IPv4 networks, use the following steps.

1. Enable packet length filtering in the global configuration mode by using the `hw-module` command.

```
Router# config
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 the global configuration mode and create an object group for configuring a scaled ACL.

```
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 the global configuration mode, configure an IPv4 access list to filter packets by the packet length value.

   In this particular example, we 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.

```
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.

```
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# end
```

5. Apply the ACL to the required HundredGigE interface.

```
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.

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

7. Verify your configuration.

```
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.

```
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.

Configuring Scaled IPv6 ACLs to Filter by Packet Length

To configure a scaled ACL to filter by packet length in IPv6 networks, use the following steps.

1. Enable packet length filtering in the global configuration mode by using the `hw-module` command.

```
Router# config
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 the global configuration mode and create an object group for configuring a scaled ACL.

```
Router(config)# object-group  network ipv6 netobject2
Router(config-object-group-ipv6)# 2001::0/128
Router(config-object-group-ipv6)# commit
```

3. From the global configuration mode, configure a scaled IPv6 access list to filter packets by the packet length value.

   In this particular example, we 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.

```
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.

```
Router(config-ipv6-acl)# commit
Router(config-ipv6-acl)# end
```

5. Apply the ACL to the required HundredGigE interface.

```
Router# config
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.

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

7. Verify your configuration.

```
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.

```
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.

Configuring TTL Matching

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 multiple values.

TTL matching is supported for both ingress and egress ACLs.

Configuration

Use the following steps to configure TTL matching.

```
/* Configure an IPv4 ACL with the TTL parameters */
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
Thu Nov  2 12:22:58.948 IST

/* Attach the IPv4 ACL to the HundredGigE interface */
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
```

Running Configuration

Validate your configuration by using the **show run** command.

```
Router(config)# show run
Thu Nov  2 14:01:53.376 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Thu Nov  2 12:22:59 2017 by annseque
!
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.

Understanding IP Access List Logging Messages

Cisco IOS XR software can provide logging messages about packets permitted or denied by a standard IP access list. That is, 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.

However, 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. You might do this 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 caching it; 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, the logging facility should not be used as a billing tool or as an accurate source of the number of matches to an access list.

Enable Interface Logging on ACLs

Table 21: Feature History Table

Feature Name	Release Information	Feature Description
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

Feature Name	Release Information	Feature Description
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 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)

Enable Ingress Interface Logging

This section shows you how to configure an ACL with the optional keyword **log-input**. This 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 bridged-virtual interfaces (BVI).

```
Router#configure
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
Router(config)# interface FortyGigE0/0/0/22
```

```
Router(config-if)# ipv4 access-group test ingress
Router(config-if)# commit
```

The following snippet shows a sample log message when the you enable this option on an ACE:

```
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 Configuring IPv4 ACLs, on page 99 and Configuring IPv6 ACLs, on page 102.

Enable Egress Interface Logging

This section shows you how to configure an ACL with the optional keyword **log** for an egress traffic.

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

Configuration Guidelines and Limitations

Following are the configuration guidelines and limitations for the ACL logging for egress traffic:

- The router supports this feature for both IPv4 and IPv6 egress ACLs on main interfaces, sub-interfaces and bridged-virtual interfaces (BVI).
- This feature is supported only on the Cisco 8000 Series routers that use Q100 and Q200-based line cards.

Configuring ACL Logging for Egress Traffic

To enable egress interface ACL logging, perform the following steps as described in the following example. In this example, an IPv4 ACL is configured and then the ACL logging is enabled.

```
Router#configure
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
Router(config)# interface FourHundredGigE0/1/0/0
Router(config-if)# ipv4 access-group test-egress-acl egress
Router(config-if)# commit
```

Running Configuration

The following example shows the running configuration of an IPv4 egress ACL with logging enabled on an interface.

```
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
!
```

Verification

Verify the packet counts for the filtered egress packets on the interface.

```
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
```

Verify the ACL logging message for the egress traffic.

```
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
```

Per Interface Statistics

When binding ACL to interfaces, you can configure ACE drop counts by using the **interface-statistics** keyword in the per-interface mode.

In Cisco 8000 Series Routers, you can allocate up to 8 stats counters per NPU.

This also limits the number of the interface that can support the same ACL in per-interface-stats mode to 8. Additional binding of the same ACL in per-interface-stats mode will be rejected.

Note For a specific direction, all interfaces on a line card using the same ACL must be configured in the same stats mode, if not the subsequent binding will be rejected.

This is true for the bundle interface too. If you add any member to the existing bundle interface with a different stats mode, the binding will be rejected.

Configuration Example

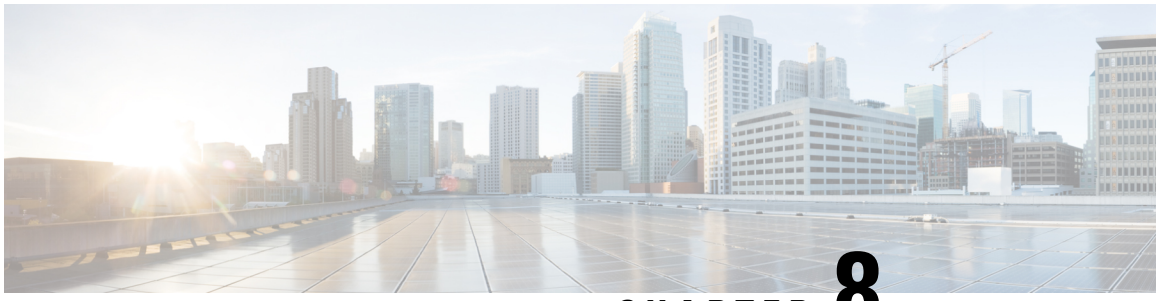
```
Router# interface HundredGigE 0/0/0/0
Router(config-if)#ipv4 address 1.1.1.1 255.255.0.0
Router(config-if)#ipv6 address 2001::1/64
Router(config-if)#ipv4 access-group test ingress interface-statistics
Router(config-if)#commit
```

Verification

To display deny stats on the interface:

```
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)
```



CHAPTER 8

Implementing Cisco Express Forwarding

- Implementing Cisco Express Forwarding, on page 143
- Prerequisites for Implementing Cisco Express Forwarding, on page 144
- Verifying CEF, on page 144
- Configuration Status of Cisco Express Forwarding Hardware Modules, on page 146
- Configuring Static Route, on page 146
- BGP Attributes Download, on page 147
- Proactive Address Resolution Protocol and Neighbor Discovery, on page 148
- Route Scale Improvements, on page 149

Implementing Cisco Express Forwarding

Cisco Express Forwarding (CEF) is an advanced, Layer 3 IP switching technology. CEF optimizes network performance and scalability for networks with large and dynamic traffic patterns, such as the Internet, on networks characterized by intensive web-based applications, or interactive sessions. CEF is an inherent feature and the users need not perform any configuration to enable it. If required, the users can change the default route purge delay and static routes. Cisco 8000 Series Routers supports only single stage forwarding.

Components

Cisco IOS XR software CEF always operates in CEF mode with two distinct components:

- Forwarding Information Base (FIB) database: The protocol-dependent FIB process maintains the forwarding tables for IPv4 and IPv6 unicast in the route processor and line card (LC). The FIB on each node processes Routing Information Base (RIB) updates, performing route resolution and maintaining FIB tables independently in the route processor and line card (LC). FIB tables on each node can be slightly different.
- Adjacency table—a protocol-independent adjacency information base (AIB)

CEF is a primary IP packet-forwarding database for Cisco IOS XR software. CEF is responsible for the following functions:

- Software switching path
- Maintaining forwarding table and adjacency tables (which are maintained by the AIB) for software and hardware forwarding engines

The following features are supported for CEF on Cisco IOS XR software:

- Bundle interface support
- Multipath support
- Route consistency
- High availability features such as packaging, restartability, and Out of Resource (OOR) handling
- OSPFv2 SPF prefix prioritization
- BGP attributes download

CEF Benefits

- Improved performance—CEF is less CPU-intensive than fast-switching route caching. More CPU processing power can be dedicated to Layer 3 services such as quality of service (QoS) and encryption.
- Scalability—CEF offers full switching capacity at each line card.
- Resilience—CEF offers an unprecedented level of switching consistency and stability in large dynamic networks. In dynamic networks, fast-switched cache entries are frequently invalidated due to routing changes. These changes can cause traffic to be process switched using the routing table, rather than fast switched using the route cache. Because the Forwarding Information Base (FIB) lookup table contains all known routes that exist in the routing table, it eliminates route cache maintenance and the fast-switch or process-switch forwarding scenario. CEF can switch traffic more efficiently than typical demand caching schemes.

The following CEF forwarding tables are maintained in Cisco IOS XR software:

- IPv4 CEF database—Stores IPv4 Unicast routes for forwarding IPv4 unicast packets
- IPv6 CEF database—Stores IPv6 Unicast routes for forwarding IPv6 unicast packets

Prerequisites for Implementing Cisco Express Forwarding

The following prerequisites are required to implement Cisco Express Forwarding:

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

Verifying CEF

To view the details of the IPv4 or IPv6 CEF tables, use the following commands:

- show cef {ipv4 | ipv6} summary

Displays a summary of the IPv4 or IPv6 CEF table.

```
Router#show cef ipv4 summary
Fri Nov 20 13:50:45.239 UTC
```

```
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
```

- show cef { ipv4 address | ipv6 address } detail

Displays the details of the IPv4 or IPv6 CEF table.

```
Router#show cef 203.0.1.2 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 Cisco Express Forwarding Hardware Modules

Table 22: Feature History Table

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.

To understand the configuration status of different CEF hardware modules, use the **show hw-module profile cef** command. Use the command in the XR EXEC mode to display the following information:

```
Router# show hw-module profile cef
Tue Oct 6 00:34:47.735 UTC
-----------------------------------------------------------------------------------------
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
```

You can get the following information from the output of the **show hw-module profile cef** command:

- The **Status** column indicates whether you have configured the corresponding hardware-module for CEF.
- The **Applied** column indicates whether you have applied the command on the router.
- The **Action** column indicates whether you must reload router to apply the configuration.

Restrictions

The Yang data model for the **show hw-module profile cef** command is not available.

Associated Commands

- show hw-module profile cef

Configuring Static Route

Routers forward packets using either route information from route table entries that you manually configure or the route information that is calculated using dynamic routing algorithms. Static routes, which define explicit

paths between two routers, cannot be automatically updated; you must manually reconfigure static routes when network changes occur. Static routes use less bandwidth than dynamic routes. Use static routes where network traffic is predictable and where the network design is simple. You should not use static routes in large, constantly changing networks because static routes cannot react to network changes. Most networks use dynamic routes to communicate between routers but might have one or two static routes configured for special cases. Static routes are also useful for specifying a gateway of last resort (a default router to which all unroutable packets are sent).

Configuration Example

Create a static route between Router A and B over a HundredGigE interface. The destination IP address is 203.0.113.0/24 and the next hop address is 192.0.2.1.

```
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
```

Running Configuration

```
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
 !
!
```

Associated Commands

- router static
- show cef

BGP Attributes Download

The BGP Attributes Download feature enables you to display the installed BGP attributes in CEF.

- The **show cef bgp-attribute** command displays the installed BGP attributes in CEF.
- The **show cef bgp-attribute** *attribute-id* command and the **show cef bgp-attribute** *local-attribute-id* command are used to view the specific BGP attributes by attribute ID and local attribute ID.

Verification

```
Router# show cef bgp-attribute
Wed Aug 21 14:05:51.772 UTC

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
```

Proactive Address Resolution Protocol and Neighbor Discovery

When CEF installs a route for which there is no layer 2 adjacency information, CEF creates an incomplete layer 3 next-hop and programs it on the hardware. Because of this incomplete programming, the first packet will be forwarded to the software forwarding path. The software forwarding in turn strips off the layer 2 header from the packet and forwards it to ARP (Address Resolution Protocol) or ND (Neighbor Discovery) in order to resolve the layer 2 adjacency information. In such a packet, if there is feature specific information present in the layer 2 header, the software forwarding path fails to strip off the layer 2 header completely and thus ARP or ND is unable to resolve the missing layer 2 adjacency information and thereby this results in traffic being dropped.

Proactive ARP and ND feature solves the above problem by ensuring that CEF proactively triggers ARP or ND in order to resolve the missing layer 2 adjacency information, retrying every 15 seconds until the next-hop information is resolved. Thus, when you configure a static route which has an incomplete next-hop information, this feature automatically triggers ARP or ND resolution.

Configuration

```
/* Enter the configuration mode and configure Proactive ARP/ND */
Router# configure
Router(config)# cef proactive-arp-nd enable
Router(config)# commit
```

Running Configuration

```
Show running-config
cef proactive-arp-nd enable
end
```

Route Scale Improvements

Table 23: 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.

The Forwarding Information Base (FIB) process maintains the forwarding database for IPv4 and IPv6 unicast in the route processor (RP) and line card (LC). Each routing protocol selects its set of suitable routes and installs those routes and their attributes in the Routing Information Base (RIB). RIB stores these routes and uses it to select an appropriate route for each routing protocol. The FIB on each RP or LC downloads those routes to the line cards and uses them for forwarding packets. The route scale is the capacity of the FIB database to store route entries for IPv4 and IPv6 traffic types.

With this feature, you can increase the routing scale for IPv4 and IPv6 traffic types. For IPv4, you can increase the FIB resources from 2 million to 3 million entries. In IPv6, you can increase the FIB resources from 0.5 million to 1 million entries. The router supports this scale improvement by reassigning the available resources within itself. Increasing the routing scale restricts resources for packet classification features such as Security ACL, QoS ACL, BGP Flowspec, and LPTS.

Restrictions

Enabling the route scale improvements results in resource crunch for packet classification features such as Security ACL, QoS ACL, BGP Flowspec, and LPTS.

Note Enabling route scale improvements for IPv4 and IPv6 traffic types, reduces the TCAM space for other packet classification features on the router by 512 and 256 entries respectively.

Configuration

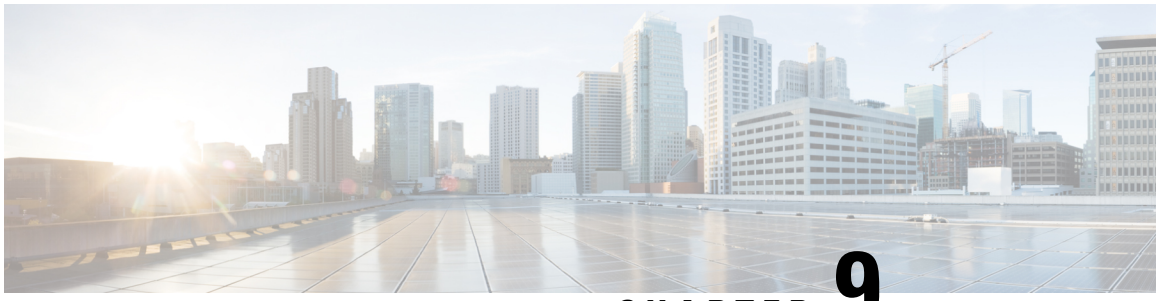
The following sections details the configuration to increase the route scale for IPv4 traffic type to 3 million and IPv6 traffic type to 1 million entries:

```
Router # config
Router(config)# hw-module profile route scale lpm tcam-banks
Router(config)# commit
Router(config)# reload location all
```

Note For restoring the route scale for IPv4 traffic type to 2 million and IPv6 traffic type to 0.5 million entries, use the **no** form of the **hw-module profile route scale lpm tcam-banks** command and execute the **reload location all** command.

Running Configuration

```
Router# show running-config
Building configuration...
hw-module profile route scale lpm tcam-banks
```



CHAPTER 9

Implementing LPTS

- LPTS Overview, on page 151
- LPTS Policers, on page 151
- LPTS and NPU Traps, on page 154
- Defining Dynamic LPTS Flow Type , on page 156
- User-managed control and management plane ACLs, on page 158

LPTS Overview

Local Packet Transport Services (LPTS) maintains tables describing all packet flows destined for the secure domain router (SDR), making sure that packets are delivered to their intended destinations.

LPTS uses two components to accomplish this task: the port arbitrator and flow managers. The port arbitrator and flow managers are processes that maintain the tables that describe packet flows for a logical router, known as the Internal Forwarding Information Base (IFIB). The IFIB is used to route received packets to the correct Route Processor for processing.

LPTS interfaces internally with all applications that receive packets from outside the router. LPTS functions without any need for customer configuration. However, the policer values can be customized if required. The LPTS show commands are provided that allow customers to monitor the activity and performance of LPTS flow managers and the port arbitrator.

LPTS Policers

Table 24: Feature History Table

Feature Name	Release Information	Description
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.

In Cisco IOS XR, the control packets, which are destined to the Route Processor (RP), are policed using a set of ingress policers in the incoming ports. These policers are programmed statically during bootup by LPTS components. The policers are applied based on the flow type of the incoming control traffic. The flow type is determined by looking at the packet headers. The policer rates for these static ingress policers are defined in a configuration file, which are programmed on the route processor during bootup. You can change the policer values based on the flow types of these set of ingress policers. You can configure the rate per policer per node.

Note When configuring the LPTS policer rates using the **lpts pifib hardware police** command, the system may program a rate that appears to differ from the configured value. The programmed rate will be closer to the configured rate and optimized for hardware performance.

- The discrepancy between the configured and programmed rate does not impact the overall functionality or performance of the system.
- You can get the default policer values and the effective current rates of the flow types using the **show lpts pifib hardware police** command.

Configuration Example

Configure the LPTS policer for the OSPF and BGP flowtypes with the following values globally for all nodes:

- ospf unicast default rate 3000
- bgp default rate 4000

```
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
```

Running Configuration

```
Router#show running-config lpts
lpts pifib hardware police
 flow ospf unicast default rate 3000
 flow bgp default rate 4000
!
```

Verification

```
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
```

Configuration Example

Configure the LPTS policer for the OSPF and BGP flow types with the following values on an individual node - 0/0/CPU0:

- ospf unicast default rate 3000
- flow bgp default rate 4000

```
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
```

Running Configuration

```
Router#show running-config lpts
lpts pifib hardware police location 0/0/CPU0
 flow ospf unicast default rate 3000
 flow bgp default rate 4000
!
```

Verification

The **show lpts pifib hardware police location 0/0/CPU0** command displays pre-Internal Forwarding Information Base (IFIB) information for the designated node.

```
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
```

Starting Cisco IOS XR Software Release 7.3.2, you can use `Cisco-IOS-XR-lpts-pre-ifib-oper` YANG data model across all IOS XR platforms to retrieve the policer statistics of the flow type. The following example shows the sample RPC request:

```
===== RPC request ======
<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 following example show the relevant snippet of the `ICMP-local` flow response to the RPC request:

```
<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>
```

The policer stats of each flow type is the aggregate of all the NPU counters. In the example, the NPU ID of 255 indicates that the value is an aggregate of all NPU stats and provides a simplified view of policer stats per flow type.

Associated Commands

- **lpts pifib hardware police**
- **flow ospf**
- **flow bgp**
- **show lpts pifib hardware police**

LPTS and NPU Traps

Network Processing Unit (NPU) traps are raised by the routers for inspection. NPU traps are raised in response to the type of packets received by the router and can indicate either exception packets, error packets, or non-LPTS control packets.

- Examples of exception packets include glean adjacency traffic or packets with IPv4 options.
- Examples of error packets include IPv4 packet with bad checksum or IPv6 packets with a hop count of zero.

- Examples of non-LPTS control packets include those packets that do not get processed through LPTS (for example, LACP, LLDP and other L2 control packets).

Each of the NPU traps are policed at a rate that is pre-programmed by the router's system design. Packets are policed per NPU and excess traffic is dropped by the NPU with respect to the system design. Some NPU trap packets that are allowed by NPU policers are sent to the CPU if they need additional processing. Others that exceed the NPU policer rate are dropped by the NPU.

Verification

Use the command `show controllers npu stats traps-all instance` *NPU-Number*`|all location RP|LC` command to check the NPU trap statistics for all the NPUs or per NPU of a router.

For fixed systems, the NPU trap statistics is available for the location 0/RP0/CPU0 and is provided through the command `show controllers npu stats traps-all instance all location 0/RP0/CPU0`. For distributed systems, NPU trap statistics is available for the line card locations and is provided through the command `show controllers npu stats traps-all instance all location 0/1/CPU0`. You can use the command `clear controller npu stats traps-all instance` *NPU-Number*`|all location RP|LC`

In the following example:

- **(D)** indicates the trap packets that are dropped in the NPU.
- **(D*)** indicates the trap packets that are dropped in NPU but are available for analysis.
- The **Accepted** count in the output indicates the ones that are available for analysis.

```
RP/0/RP0/CPU0: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_NO_SIP_MAPPING(D*)        0    6    0x0        1       0          0          0
ETHERNET_NO_VNI_MAPPING(D*)        0    7    0x0        1       0          0          0
ETHERNET_NO_VSID_MAPPING(D*)       0    8    0x0        1       0          0          0
ARP                                0    9    0x0        264     542        0          0
ETHERNET_SA_ERROR(D*)              0    11   0x0        266     100        0          0
ETHERNET_DA_ERROR(D*)              0    12   0x0        1       0          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_BFD_INCORRECT_TTL(D*)         0    157  0x0        412     100        0          0
OAMP_BFD_INVALID_PROTOCOL(D*)      0    158  0x0        413     100        0          0
OAMP_BFD_INVALID_UDP_PORT(D*)      0    159  0x0        414     100        0          0
```

```
OAMP_BFD_INCORRECT_VERSION(D*)      0    160   0x0        415      100        0          0
OAMP_BFD_INCORRECT_ADDRESS(D*)      0    161   0x0        416      100        0          0
OAMP_BFD_MISMATCH_DISCR             0    162   0x0        417      500000     0          0
OAMP_BFD_STATE_FLAG_CHANGE          0    163   0x0        418      500000     0          0
OAMP_BFD_SESSION_RECEIVED(D)        0    164   0x0        419      100        0          0
OAMP_PFC_LOOKUP_FAILED(D*)          0    165   0x0        420      100        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
```

Defining Dynamic LPTS Flow Type

The Dynamic LPTS flow type feature enables you to configure LPTS flow types and also enables you to define the maximum LPTS entries for each flow type in the TCAM. The dynamic LPTS flow type configuration is on per line card basis, hence you can have multiple profiles configured across line cards.

When the router boots, the default LPTS flow types are programmed in the TCAM. For each flow type the maximum flow entries are predefined. Later, at runtime, you have an option to choose the flow type based on network requirements and also configure the maximum flow entry value. The maximum flow entry value of zero denotes that a flow type is not configured.

Note You can get the default maximum flow values for both configurable flow and non-configurable flow from the output of the following show command:

```
show lpts pifib dynamic-flows statistics location <location specification>
```

The list of configurable and non-configurable flow types are listed in below tables. You can also use **show lpts pifib dynamic-flows statistics location** command to view the list of configurable and non-configurable flow types:

Note The sum of maximum LPTS entries configured for all flow types must not exceed 16000 entries per line card.

Configuration Example

In this example you will configure the BGP-known and ISIS-known LPTS flow type in the TCAM and define the maximum flow entries as 1800 and 500 for node location 0/1/CPU0. As the new maximum values are more than the default values, we have to create space in the TCAM by disabling other flow types so that the sum of maximum entries for all flow types per line card does not exceed 8000 entries. Hence RSVP-known flow type is set to zero in our example:

The maximum dynamic scale for any flow type should be configured such that all LPTS entries for that flow type are in hardware. One way to achieve that is to increase the dynamic scale. This may help avoid session flaps for NSR-enabled protocols like BGP and OSPF in case of triggers like RP fail overs.

```
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
```

Running Configuration

```
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
!
```

Verification

This show command displays dynamic flow statistics. You can see that the flow types BGP-known and ISIS-known are configured in the TCAM with newly configured maximum flow entry value. You can also see that the RSVP-known flow type is disabled:

```
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
 -------------------- -- ------- --------   -------/--------    -------- -
 Fragment             F        2       --         2/2                2
 OSPF-mc-known        T      600       --         2/600              2
 OSPF-mc-default      F        4       --         4/4                4
 OSPF-uc-known        T      300       --         1/300              1
 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
 BGP-cfg-peer         T      900       --         0/900              0
 BGP-default          F        4       --         4/4                4
 PIM-mcast-default    F       40       --         0/40               0
 PIM-mcast-known      T      300       --         0/300              0
 PIM-ucast            F       40       --         2/40               2
 IGMP                 T     1200       --         0/1200             0
 ICMP-local           F        4       --         4/4                4
 ICMP-control         F        5       --         5/5                5
 LDP-TCP-known        T      300       --         0/300              0
 LDP-TCP-cfg-peer     T      300       --         0/300              0
 LDP-TCP-default      F       40       --         0/40               0
 LDP-UDP              T      300       --         0/300              0
 All-routers          T      300       --         0/300              0
 RSVP-default         F        4       --         1/4                1
 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
 SHTTP-known          T       40       --         0/40               0
 TELNET-known         T       40       --         0/40               0
 TELNET-default       T        1       --         1/1                1
 UDP-known            T        0       --         0/0                0
 UDP-default          F        2       --         2/2                2
 TCP-known            T       40       --         0/40               0
 TCP-default          F        2       --         2/2                2
```

```
 Raw-default           F         2        --          2/2                 2
 GRE                   F         4        --          0/4                 0
 VRRP                  T       150        --          0/150               0
 DNS                   T        40        --          0/40                0
 NTP-known             T        40        --          0/40                0
 DHCPv4                T        40        --          0/40                0
 DHCPv6                T        40        --          0/40                0
 TPA                   T      1000        --          0/1000              0
 PM-TWAMP              T        10        --          0/10                0
-----------------------------------------------------
 Active TCAM Usage : 13421/16000 [Platform MAX: 16000]
 HWCnt/SWCnt       : 65/88
-----------------------------------------------------
```

In the above show command output, the last column **P** specifies the pending software flow entries for the flow type.

User-managed control and management plane ACLs

User-managed control and management plane ACLs are security features that

- enable creation of a virtual LPTS interface
- allow assignment of hybrid ACLs to the virtual LPTS interface for traffic filtering, and
- provide customization for control plane and management plane traffic.

Table 25: 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 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.

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**

Overview

- Data Plane: Handles all functions and processes involved in forwarding packets from one network interface to another. This is where actual user data flows through the device.
- Control Plane: Performs functions and processes that determine the path packets should take through the network, such as routing protocols and decisions.
- Management Plane: Uses functions and processes used to control and monitor the device itself, including device configuration and status monitoring.

Traditional ACLs control and manage data plane traffic but do not provide visibility or control over traffic destined for the control or management planes. This feature allows you to create a virtual (LPTS) interface in the router, assign hybrid ACLs for customizing the control and management planes to filter traffic, and set policer rates in the ACEs of a hybrid ACL with compression level 2 to control and manage the control plane and management plane traffic.

To enable LPTS ACL mode, use the **hw-module profile cef lpts acl** command. To disable the LPTS ACL mode, use the **no hw-module profile cef lpts acl** command.

From Release 7.10.1 onwards, you can configure the ACLs to include the Authentication Header (AH) and Encapsulating Security Payload (ESP) headers. The AH and ESP headers are used within IP Security Protocol (IPSec). While AH provides data integrity, ESP provides confidentiality of a packet.

Note

- We do not recommend using UMPP for BFD because BFD is hardware-offloaded, and UMPP does not fully support BFD.
- If you configure a UMPP ACL to deny BFD packets with logging enabled, BFD packets are not dropped, and sessions establish successfully.
- If you configure a UMPP ACL to deny BFD packets without logging enabled, BFD packets are dropped, and sessions do not establish.
- UMPP ACL is used only during the initial bringup of the BFD session, when the remote discriminator is 0. After the system learns the remote discriminator, it does not perform a UMPP ACL lookup on BFD packets for the session. If you apply UMPP after the BFD session is up, it does not affect BFD packets.

Best practices for control and management plane ACLs

Follow these best practices when configuring the control plane and management plane ACLs:

- Apply hybrid ACLs for control and management plane traffic to support object group match and policer actions. For more information, see Hybrid ACLs, on page 92 and LPTS Policers, on page 151.
- Manage control and management plane ACLs to support configuring policer rate and priority options in the ACE.
- Activate logging actions on relevant ACLs to ensure proper monitoring and handling of traffic.
- Use the **icmp-on** option to punt packets matching deny ACEs, which the router drops by default.
- Create UMPP-related object groups before applying the UMPP ingress ACL under the lpts0 interface.
- Reboot the router after including AH and ESP in your ACL configuration.
- Create a single LPTS interface for UMPP ACL and add ACEs to customize control and management plane traffic within the same IPv4 or IPv6 ACL. For more information, see Configuring Control Plane and Management Plane Traffic, on page 161.
- Use only the object group with Level 2 compression in LPTS ACL mode.

Restrictions for user-managed control and management plane ACLs

Follow these restrictions for control and management plane ACLs:

- Do not rely on hybrid ACLs for control and management plane traffic to filter BFD control packets if BFD sessions are hardware-offloaded.
- Do not use these protocol and port combinations in the UMPP configuration to permit or deny packets. If used, the protocols do not work as expected.

Protocols	Ports
PTP	UDP ports – 319, 320
BFD	UDP ports – 3784, 3785
IPLSA	• TCP port – 1167 • UDP ports – 1167, 1967
TWAMP	TCP – port 862

  If you want to deny packets while using any of these protocol and port combinations, use **deny ipv4 any any** or **deny ipv6 any any** in the ACL configuration.

- When you configure the **hw-module profile cef lpts acl** command, control plane packets such as BFD are blocked until the UMPP LPTS ACL is applied with permit ACEs.

Configuring Control Plane and Management Plane Traffic

Note When enabling the UMPP feature with the **hw-module profile cef lpts acl** command, it is required to configure an ACL and apply it to the LPTS interface.

Use the following configuration to customize control plane and management plane traffic:

```
/* Enable LPTS ACL mode */
Router (config)# hw-module profile cef lpts acl
Router(config-ipv4-acl)# commit
Router(config)# exit

/* Create IPv4 ACL */
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
Router(config-ipv4-acl)# exit

/* Create IPv6 ACL */
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-ipv4-acl)# priority-timeout 25
Router(config-ipv6-acl)# commit
Router(config-ipv6-acl)# exit

/*Assign the IPv4 and IPv6 ACLs to the virtual LPTS created on enabling the LPTS ACL mode*/
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
Router(config-if)# exit

/*Reboot the router*/
```

To disable the LPTS ACL mode, run the **no hw-module profile cef lpts acl** command.

Verification

Use the following commands to verify if the LPTS ACL mode is enabled in the router:

```
Router#show hw-module profile cef
Tue Apr  6 09:06:33.982 UTC
-----------------------------------------------------------------
Knob                          Status          Applied   Action
-----------------------------------------------------------------
CBF                           Unconfigured    N/A       None
BGPLU                         Unconfigured    N/A       None
LPTS ACL                      Configured      Yes       None
```

```
Dark Bandwidth                Unconfigured    N/A       None
IP Redirect Punt              Unconfigured    N/A       None
IPv6 Hop-limit Punt           Unconfigured    N/A       None
MPLS Per Path Stats           Unconfigured    N/A       None
Tunnel TTL Decrement          Unconfigured    N/A       None
High-Scale No-LDP-Over-TE     Unconfigured    N/A       None
LPTS Pifib Entry Counters     Unconfigured    N/A       None
```

```
Router# show access-lists test-umpp-v4-filter hardware ingress interface lpts 0 location
0/RP0/CPU0
ipv4 access-list test-umpp-v4-filter
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
ipv6 access-list test-umpp-v6-filter
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
```

Configuring ACLs for AH and ESP Headers

Use the following configuration to customize control plane and management plane traffic for AH and ESP headers:

```
/* Enable LPTS ACL mode */
Router (config)# hw-module profile cef lpts acl
Router(config-ipv4-acl)# commit
Router(config)# exit
```

```
/* Create IPv6 ACL for AH and ESP header*/
Router(config)# ipv6 access-list ipv6_umpp_access_list
Router(config-ipv6-acl)# 12 permit ahp any any
Router(config-ipv6-acl)# ipv6 access-list ipv6_umpp_access_list
Router(config-ipv6-acl)# 14 permit esp any any
Router(config-ipv6-acl)# commit
Router(config-ipv6-acl)# exit
```

```
/*Assign the IPv6 ACLs to the virtual LPTS created on enabling the LPTS ACL mode*/
Router(config-if)# ipv6 access-group ipv6_umpp_access_list ingress compress level 2
Router(config-if)# commit
Router(config-if)# exit

/*Reboot or reload the router*/
Router(config)# reload location 0/0/CPU0
```

Note Ensure that you reboot the router after you configure your ACLs to include the AH and ESP headers.

Running Configuration

```
Router#show running-config
Tue Apr  4 19:34:56.697 UTC
!! Building configuration...
!! IOS XR Configuration 7.10.1.18I
!! Last configuration change at Tue Apr  4 19:21:05 2023 by xyz
!
hostname abc
logging console disable
username xyz
 group root-lr
 group cisco-support
 secret 10
$6$9gZvb/PtyNiT4b/.$2I516Wdlhm7FGa35sKPhmKkGPEibS.chyRzycSzxmFhrR/kdo9JvKCVA7G8gDya1GBtvGDomubxYj1y75gw.g1
!
line template vty
 exec-timeout 0 0
!
line template test
 exec-timeout 0 0
!
line console
 timeout login response 30
 exec-timeout 0 0
!
line default
 exec-timeout 0 0
 absolute-timeout 0
 session-timeout 0
!
vty-pool default 0 99 line-template vty
call-home
 service active
 contact smart-licensing
 profile CiscoTAC-1
  active
  destination transport-method email disable
  destination transport-method http
 !
!
netconf-yang agent
 ssh
!
ipv4 virtual address 6.7.141.12/16
ipv6 access-list ipv6_umpp_access_list
 10 permit icmpv6 any any packet-too-big
 12 permit ahp any any
 14 permit esp any any
 20 permit icmpv6 any any time-exceeded
 30 permit icmpv6 any any parameter-problem
 40 permit icmpv6 any any echo
 50 permit icmpv6 any any echo-reply
 60 permit icmpv6 any any nd-ns
 70 permit icmpv6 any any nd-na
 80 permit icmpv6 any any
 90 permit udp any any eq bootps
 100 permit udp any any range 33434 33689
 110 permit udp any any eq 1985
 120 permit tcp any eq bgp any
 130 permit tcp any any eq bgp
 140 permit tcp any any eq 57400
 150 permit rsvp any any
 160 permit vrrp any any
```

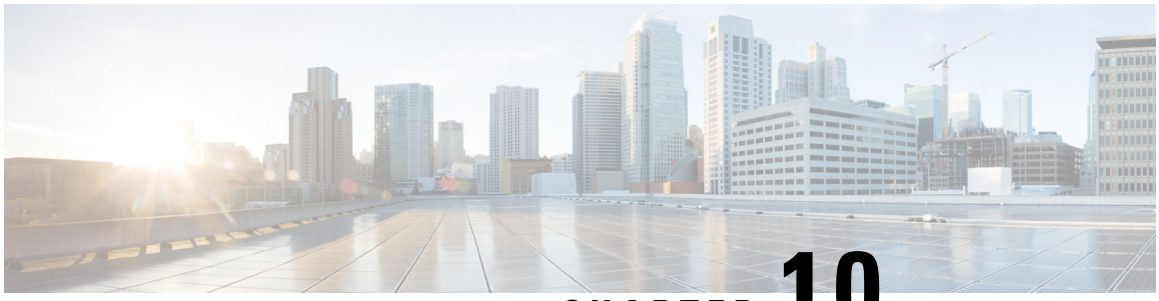
```
 170 permit udp any any range bfd 3785
 180 permit udp any any eq 4784
 190 permit udp any any eq 6784
 200 permit udp any any eq snmp
 210 permit tcp any any eq ssh
 220 permit tcp any any eq telnet
 230 permit udp any eq domain any range 1024 65535
 240 permit udp any any eq ntp
 250 permit udp any any range 1024 1030
 260 permit tcp any any eq 3220
 270 permit udp any eq 3503 any eq 3503
 280 permit tcp any any eq 6666
 290 permit pim any any
 300 permit ospf any any
 320 permit udp any any eq bootpc
 340 permit tcp any any eq www
 350 permit tcp any any eq https
 360 permit tcp any any range 5900 5910
 370 permit tcp any any range 50000 50100
 380 permit udp any any range 51000 51100
 390 permit udp any any eq 547
 400 permit udp any any eq 546
 402 permit tcp any any eq 6040
 403 permit tcp any any eq 9200
 404 permit tcp any any eq snmp
 405 permit udp any eq 7784 any eq 7784
 406 permit udp any any eq 7784
 407 permit tcp any eq tacacs any
 408 permit tcp any any eq tacacs
 409 permit udp any host ff02::1
 410 permit tcp any any eq 2018
 411 permit tcp any any eq 60100
 430 deny ipv6 any any
!
ipv4 access-list ipv4_umpp_access_list
 10 permit icmp any any fragments
 20 permit icmp any any echo
 30 permit icmp any any echo-reply
 40 permit icmp any any time-exceeded
 50 permit icmp any any unreachable
 60 permit udp any any eq ntp fragment-type first-fragment
 70 permit udp any any fragment-type last-fragment
 80 permit udp any any fragment-type is-fragment
 90 permit udp any any fragments
 100 permit tcp any any eq ssh fragment-type first-fragment
 110 permit tcp any any fragment-type last-fragment
 120 permit tcp any any fragment-type is-fragment
 130 permit udp any any fragments
 140 permit udp any any range 33434 33689
 150 permit rsvp any any
 160 permit tcp any any eq bgp
 170 permit pim any any
 180 permit igmp any any
 190 permit vrrp any any
 200 permit udp any any eq 1985
 210 permit udp any any eq bootps
 220 permit udp any any range bfd 3785
 230 permit udp any any range bfd 4784
 240 permit udp any any range bfd 6784
 250 permit tcp any any eq ssh
 260 permit udp any eq domain any range 1024 65535
 270 permit tcp any any eq telnet
 280 permit udp any eq 3503 any eq 3503
 290 permit tcp any any eq ldp
```

```
 300 permit udp any any eq ldp
 310 permit udp any any eq snmp
 320 permit udp any any eq snmptrap
 330 permit tcp any any eq 3220
 340 permit tcp any any eq 6666
 350 permit udp any any eq ntp
 360 permit udp any any range 1024 65535
 370 deny ipv4 any any packet-length range 1 999 fragment-type first-fragment
 380 deny ipv4 any any fragment-offset eq 1
 390 permit ipv4 any any fragment-type first-fragment
 400 permit ipv4 any any fragment-type last-fragment
 420 permit ospf any any
 430 permit icmp any any packet-length range 1000 2999 fragment-type is-fragment
 440 permit udp any any packet-length range 1000 2999 fragment-type is-fragment
 450 permit tcp any any packet-length range 1000 2999 fragment-type is-fragment
 460 permit ipv4 any any packet-length range 1000 2999 fragment-type is-fragment
 470 permit tcp any any eq 57400
 480 deny ipv4 any any
!
interface MgmtEth0/RP0/CPU0/0
 ipv4 address 6.7.141.11 255.255.0.0
!
interface lpts0
 ipv4 access-group ipv4_umpp_access_list ingress compress level 2
 ipv6 access-group ipv6_umpp_access_list ingress compress level 2
!
interface HundredGigE0/0/0/0
 shutdown
!
interface HundredGigE0/0/0/1
 shutdown
!
interface HundredGigE0/0/0/2
 shutdown
!
<output truncated>
```

Verification

Use the **show access-lists ipv6 hardware interface location** command to verify if the AH and ESP packets are permitted or denied on the router. This example displays the AH and ESP packets that are permitted on the router. The **Accepted** value displays the number of packets permitted.

```
Router# show access-lists ipv6 ipv6_umpp_access_list hardware ingress interface lpts0
location 0/rp0/cpu0
Tue Mar 21 14:24:43.893 UTC
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)
```



CHAPTER 10

Implement VRRP

This module describes the concepts and tasks you will use to configure Virtual Router Redundancy Protocol (VRRP).

- Implement VRRP, on page 167
- Generic Restrictions for VRRP Configuration, on page 167
- Understand VRRP, on page 168
- Unicast VRRP, on page 172
- VRRP over BVI, on page 173
- View VRRP statistics in the Router, on page 179

Implement VRRP

The Virtual Router Redundancy Protocol (VRRP) feature allows for transparent failover at the first-hop IP router, enabling a group of routers to form a single virtual router. For more information on VRRP and related concepts, see Understand VRRP, on page 168.

Generic Restrictions for VRRP Configuration

These are some restrictions to consider before your configure VRRP on supported interfaces on the Cisco 8000 platform.

- 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.

Understand VRRP

To implement VRRP on Cisco IOS XR software, you need to understand the following concepts:

- VRRP Overview
- Multiple Virtula Router Support
- VRRP Router Priority
- VRRP Advertisements
- Benefits of VRRP
- Hot Restartability for VRRP

VRRP Overview

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.

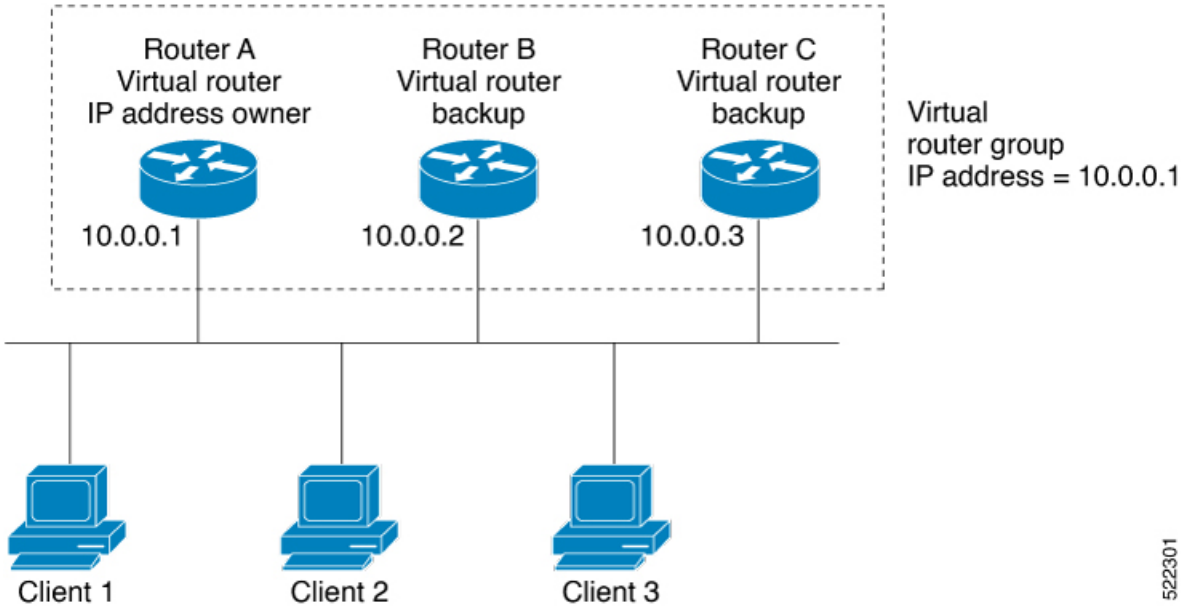
The Virtual Router Redundancy Protocol (VRRP) feature can solve the static configuration problem. VRRP is an IP routing redundancy protocol designed to allow for transparent failover at the first-hop IP router. VRRP enables a group of routers to form a single *virtual router* . The LAN clients can then 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, Basic VRRP Topology shows a LAN topology 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 13: 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.

Multiple Virtual Router Support

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

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 a IP address owner virtual router.

If no VRRP router owns the IP address, the priority of a VRRP router, combined with the reempt 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

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

The benefits of VRRP are as follows:

- 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

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.

Unicast VRRP

Table 26: Feature History Table

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)

You can now configure VRRP to support Layer 3 unicast transport, allowing it to enhance its capacity to send data to cloud networks. Pairwise router redundancy enables high availability in cloud network scenarios. The default route of the cloud native function needs a virtual IP (VIP) address because the paired routers do not have a pre-designated active member. Though HSRP provides a VIP, the cloud networks do not support Layer 2 multicast or broadcast transports. To overcome the limitations of Layer 2 multicast and broadcast transports, configure VRRP in Layer 3 unicast mode to support Layer 3 unicast transport.

This feature also enables VRRP to communicate state transition notifications using event-driven telemetry.

VRRP over BVI

Table 27: Feature History Table

Feature Name	Release Information	Feature Description
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.

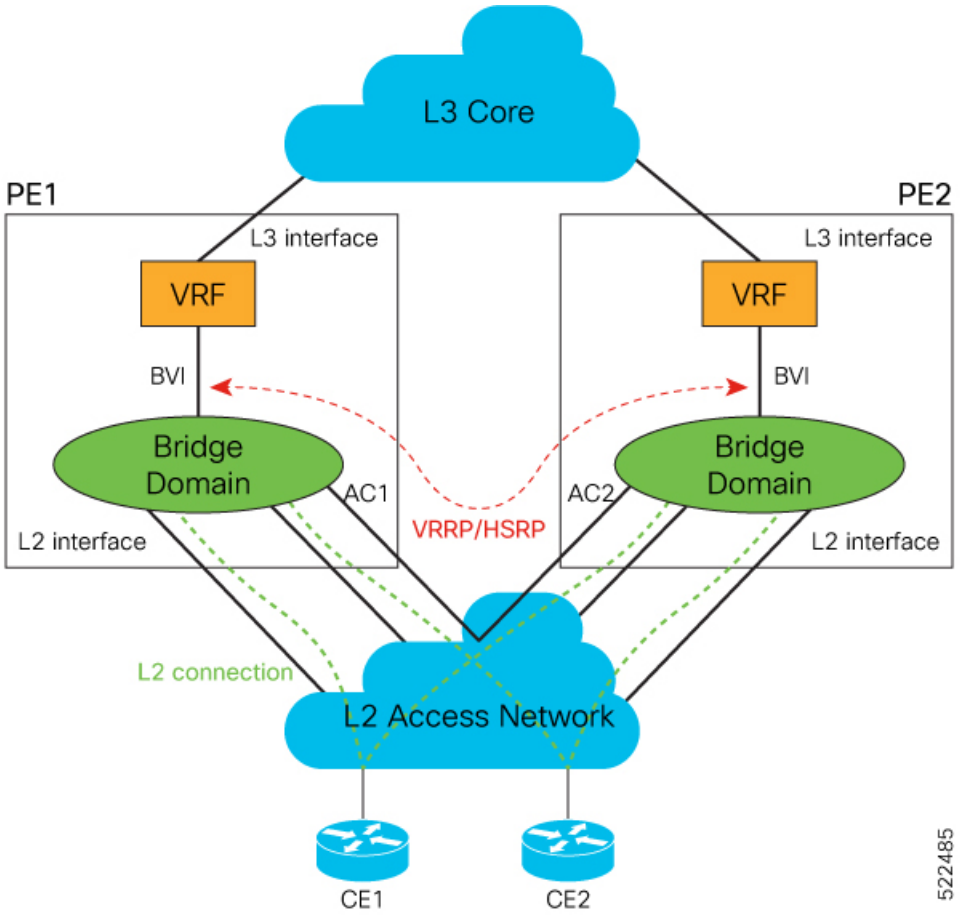
The Virtual Router Redundancy Protocol (VRRP) protocol provides default gateway redundancy for a LAN. It allows a group of routers to behave as a single virtual default gateway router. In the group of routers, the router that has the highest VRRP priority acts as the primary router and the remaining routers act as backup routers.

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.

Usually, VRRP sessions run on top of interfaces of the multiple routers which are in the same home network. You can configure VRRP session only over BVI.

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
- 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.

Restrictions

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.

Configure VRRP over BVI

To configure VRRP sessions over BVI, you must complete the following configurations on PE1 and PE2:

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.

Configuration Example

```
/* Enter the global configuration mode and configure a set of interfaces as Layer 2 interfaces
 and a set of VLAN sub-interfaces */
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

/* Enter the Layer 2 VPN configuration mode and configure a bridge group */
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

/* Configure a BVI in the global configuration mode */
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

/* Configure VRRP over BVI in the global configuration mode for IPv4 address */
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


/* Configure VRRP over BVI in the global configuration mode for IPv6 address */
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
```

Verification

Use the following command to verify the bridge domain details:

```
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
```

Use the following command to show the VRRP details:

```
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
```

```
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
```

View VRRP statistics in the Router

Table 28: Feature History Table

Feature Name	Release Information	Feature Description

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's 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's 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)

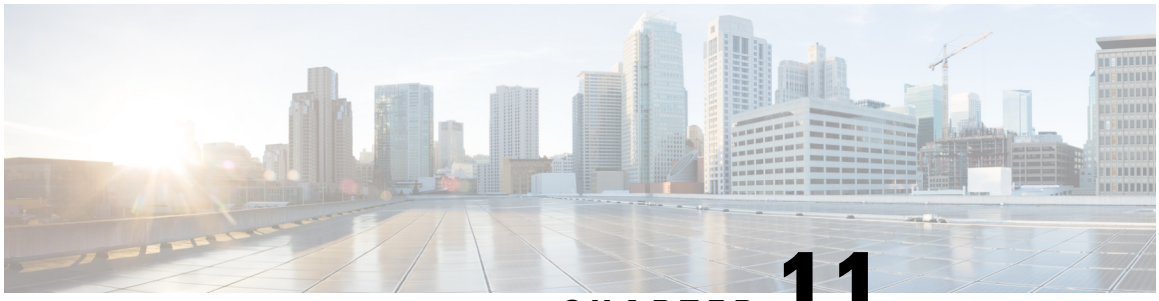
You can view statistics of one or all Virtual Router Redundancy Protocol (VRRP) virtual routers using the **show vrrp statistics** command as follows:

```
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 can clear the VRRP statistics using the **clear vrrp statistics** as follows:

```
Router# clear vrrp statistics
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
```



CHAPTER 11

Implement HSRP

This module describes the concepts and tasks you will use to configure Hot Standby Router Protocol (HSRP).

- Implement HSRP, on page 181
- General Restrictions for HSRP Configuration, on page 181
- Information About Implementing HSRP, on page 181

Implement HSRP

The Hot Standby Router Protocol (HSRP) is an IP routing redundancy protocol designed to allow for transparent failover at the first-hop IP router. HSRP provides high network availability, because it routes IP traffic from hosts on networks without relying on the availability of any single router. HSRP is used in a group of routers for selecting an active router and a standby router. (An active router is the router of choice for routing packets; a standby router is a router that takes over the routing duties when an active router fails, or when preset conditions are met.)

General Restrictions for HSRP Configuration

These are some restrictions to consider before you implement HSRP on supported interfaces on the Cisco 8000 platform.

- 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.

Information About Implementing HSRP

To implement HSRP on Cisco IOS XR software, you need to understand the following concepts:

HSRP Overview

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.

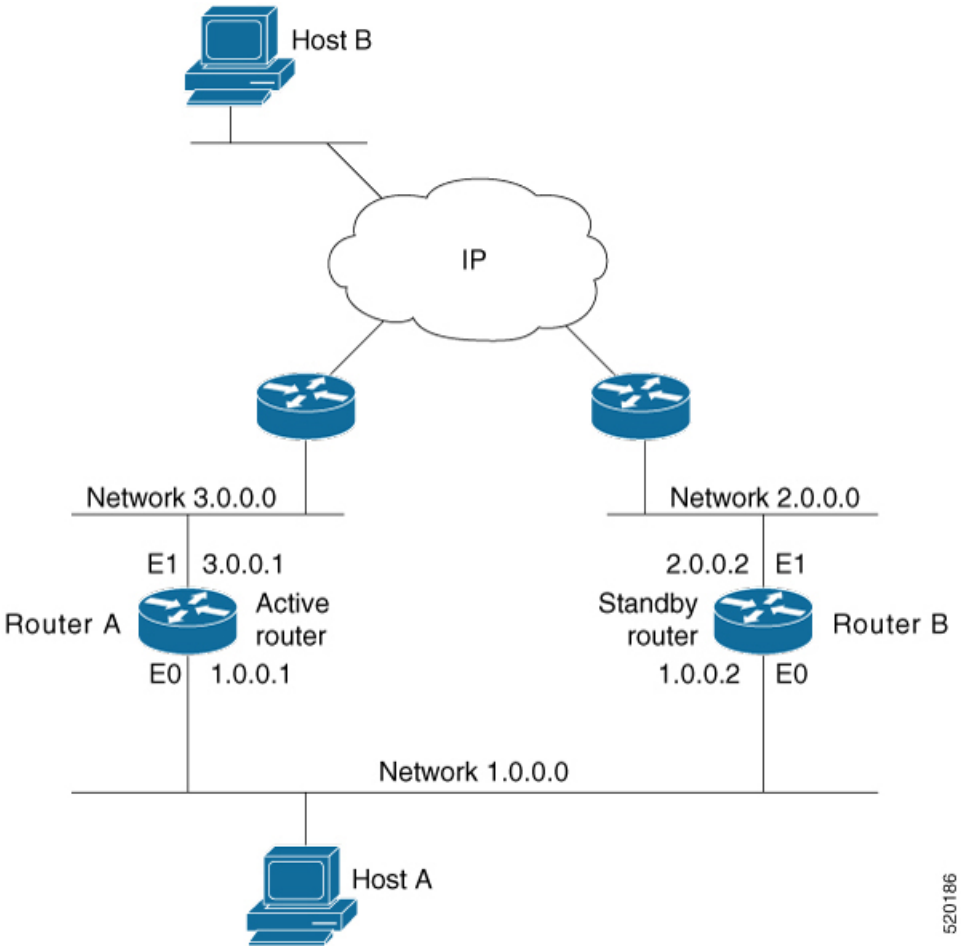
HSRP Groups

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.

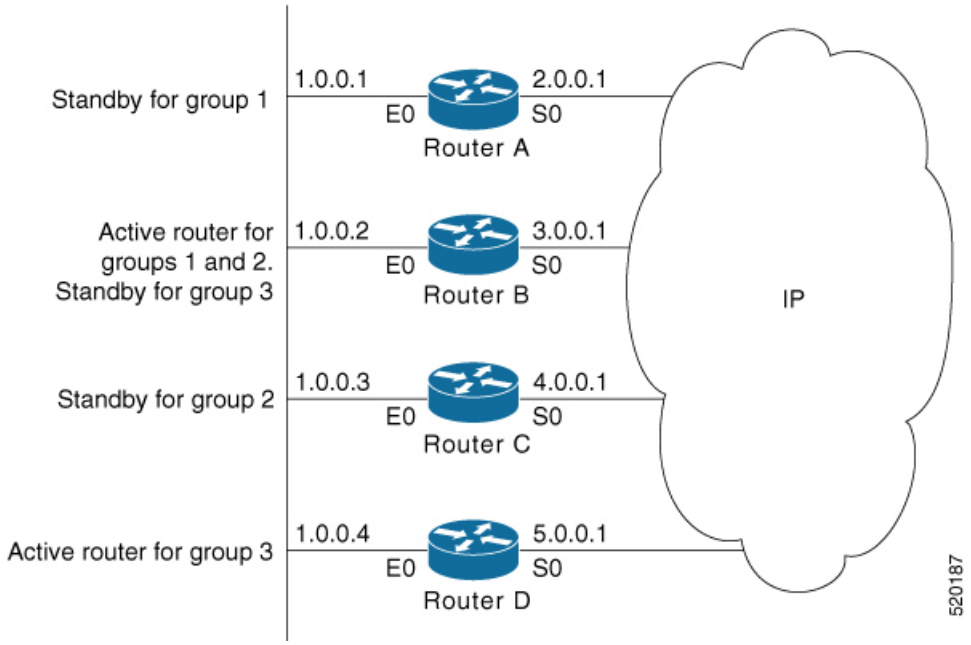
Figure 14: Routers Configured as an HSRP Group



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 15: 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

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

The HSRP preemption feature enables the router with highest priority to immediately become the active router. Priority is determined first by the priority value that you configure, and then by the IP address. In each case, a higher value is of greater priority.

When a higher-priority router preempts a lower-priority router, it sends a coup message. When a lower-priority active router receives a coup message or hello message from a higher-priority active router, it changes to the speak state and sends a resign message.

ICMP Redirect Messages

Internet Control Message Protocol (ICMP) is a network layer Internet protocol that provides message packets to report errors and other information relevant to IP processing. ICMP provides many diagnostic functions and can send and redirect error packets to the host. When running HSRP, it is important to prevent hosts from discovering the interface (or real) MAC addresses of routers in the HSRP group. If a host is redirected by ICMP to the real MAC address of a router, and that router later fails, then packets from the host are lost.

ICMP redirect messages are automatically enabled on interfaces configured with HSRP. This functionality works by filtering outgoing ICMP redirect messages through HSRP, where the next-hop IP address may be changed to an HSRP virtual IP address.

To support ICMP redirects, redirect messages are filtered through HSRP, where the next-hop IP address is changed to an HSRP virtual address. When HSRP redirects are turned on, ICMP interfaces with HSRP do this filtering. HSRP keeps track of all HSRP routers by sending advertisements and maintaining a real IP address to virtual IP address mapping to perform the redirect filtering.

HSRP over BVI

Table 29: 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

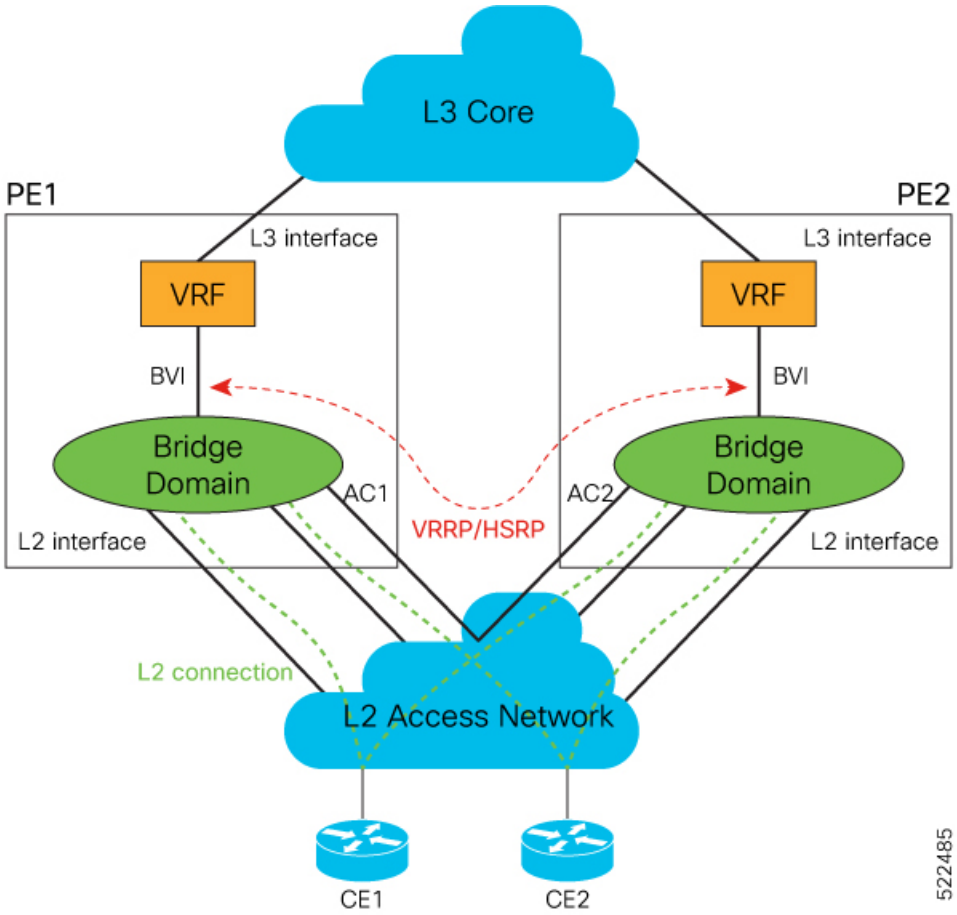
HSRP over BVI	Release 7.5.2	Hot Standby Router Protocol (HSRP) runs on top of interfaces of multiple routers in the same home network that has only Cisco routers. It allows a group of routers to behave as a single virtual default gateway router, thereby providing default gateway redundancy and minimizing 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.

The Hot Standby Router Protocol (HSRP) allows multiple routers to act as a single virtual router in a LAN which is resilient to the failure of any single one of them. The participating routers share a virtual IP address and associated virtual MAC address, used by the hosts as the default first-hop. The protocol ensures that one and only one router in the group, which is the active router is forwarding packets on behalf of the virtual router. A second router, which is the standby router is elected to replace the active router should it fail.

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.

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
- 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.

Restrictions

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.

Configure HSRP over BVI

To configure HSRP sessions over BVI, you must complete the following configurations on PE1 and PE2:

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.

Configuration Example

```
/* Enter the global configuration mode and configure a set of interfaces as Layer 2 interfaces
 and a set of VLAN sub-interfaces */
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

/* Enter the Layer 2 VPN configuration mode and configure a bridge group */
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

/* Configure a BVI in the global configuration mode */
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

/* Configure HSRP over BVI in the global configuration mode for IPv4 address */
```

```
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


/* Configure HSRP over BVI in the global configuration mode for IPv6 address */
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
```

Verification

Use the following command to verify the bridge domain details:

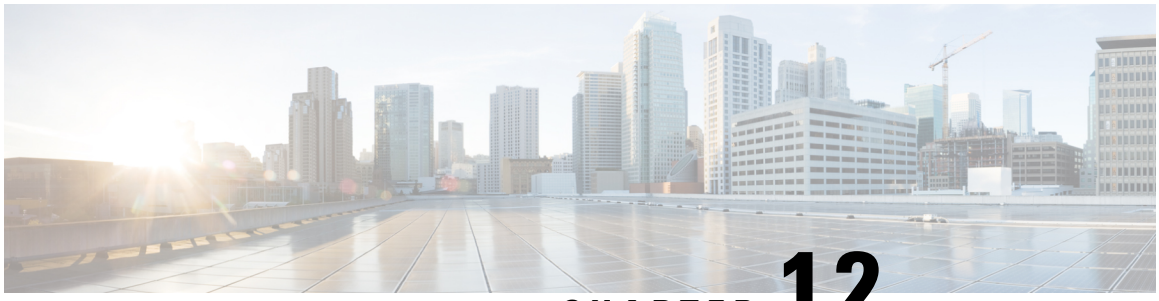
```
Router# show l2vpn bridge-domain detail

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
```

Use the following command to show the hsrp details:

```
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

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

Router# show hsrp 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
```



CHAPTER 12

Configuring Transports

- Information About Configuring NSR, TCP, UDP Transports, on page 193

Information About Configuring NSR, TCP, UDP Transports

To configure NSR, TCP, UDP, and RAW transports, you must understand the following concepts:

NSR Overview

Nonstop Routing (NSR) is provided for Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), and Label Distribution Protocol (LDP) protocols for the following events:

- Route Processor (RP) failover
- Process crash for either OSPF, LDP, or TCP
- Online insertion removal (OIR)

In the case of the RP failover, NSR is achieved by for both TCP and the applications (OSPF, BGP, or LDP).

Nonstop Routing (NSR) provides high availability for NSR-enabled routing processes by keeping sessions and transport state synchronized between the active and standby RPs. For example, TCP state for NSR-enabled protocols is mirrored so that, on an RP failover, active TCP sessions migrate to the standby RP without notifying peers. If NSR is not available, the protocols on the new active RP must reestablish sessions; Graceful Restart (GR) can reduce traffic loss during failover but has known limitations.

You can use the nsr process-failures switchover command to allow RP failover as a recovery action when an active NSR-enabled process (for example, TCP) crashes. If a standby NSR-enabled process restarts or crashes, only NSR capability is lost until the standby instances are up and sessions are resynchronized; the sessions themselves remain up. For an active process failure, a fault-management policy can be used. However, the failing process will not generate the logs required to determine the root cause of the failure.

For more information, refer to chapter *Implementing OSPF Routing Configuration Guide for Cisco 8000 Series Routers*.

Restrictions for NSR

These restrictions apply for NSR:

- Even if you configure the **nsr process-failures switchover** command, the router does not trigger Route Processor (RP) failover when a process restarts.
- The router triggers RP failover only when a process crashes.
- The NSR process-failover switchover feature supports these NSR clients:
  - L2VPN
  - VPDN
  - TCP
  - IS-IS
  - OSPF
  - MPLS-LDP
  - BGP

TCP Overview

TCP is a connection-oriented protocol that specifies the format of data and acknowledgments that two computer systems exchange to transfer data. TCP also specifies the procedures the computers use to ensure that the data arrives correctly. TCP allows multiple applications on a system to communicate concurrently, because it handles all demultiplexing of the incoming traffic among the application programs.

UDP Overview

The User Datagram Protocol (UDP) is a connectionless transport-layer protocol that belongs to the IP family. UDP is the transport protocol for several well-known application-layer protocols, including Network File System (NFS), Simple Network Management Protocol (SNMP), Domain Name System (DNS), and TFTP.

Any IP protocol other than TCP and UDP is known as a RAW protocol.

For most sites, the default settings for the TCP, UDP, and RAW transports need not be changed.

Prerequisites for Configuring NSR, TCP, UDP, Transports

The following prerequisites are required to implement NSR, TCP, UDP, Transports:

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

Configuring Failover as a Recovery Action for NSR

When the active TCP or the NSR client of the active TCP crashes, the TCP sessions go down. To continue to provide NSR, failover is configured as a recovery action. If failover is configured, a switchover is initiated if the active TCP or an active application (for example, LDP, OSPF, and so forth) crashes.

For information on how to configure MPLS Label Distribution Protocol (LDP) for NSR, refer to the *MPLS Configuration Guide for Cisco 8000 Series Routers*.

For information on how to configure NSR on a per-process level for each process, refer to the *Routing Configuration Guide for Cisco 8000 Series Routers*.

Configuration Example

Configure failover as a recovery action for active instances to switch over to a standby to maintain nonstop routing.

```
Router#configure
Router(config)#nsr process-failures switchover
Router(config)#commit
```

Running Configuration

```
Router#show running-configuration nsr process-failures switchover
nsr process-failures switchover
```

Associated Commands

- nsr process-failures switchover