

IP Addresses and Services Configuration Guide for Cisco NCS 560 Series Routers, IOS XR Release 26.1.x, 26.2.x, 26.3.x

First Published: 2026-03-01

Last Modified: 2026-09-08

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)

© 2026 Cisco Systems, Inc. All rights reserved.



CONTENTS

CHAPTER 1 **Implementing Network Stack IPv4 and IPv6** 1

Network Stack IPv4 and IPv6 Exceptions 1

IPv4 and IPv6 Functionality 2

Custom Prefix Length Selection 2

IPv6 for Cisco IOS XR Software 9

How to Implement Network Stack IPv4 and IPv6 9

Configuring IPv4 Addressing 9

Configuring IPv6 Addressing 10

IPv6 Multicast Groups 11

Assigning Multiple IP Addresses to Network Interfaces 15

Configuring IPv4 and IPv6 Protocol Stacks 16

Enabling IPv4 Processing on an Unnumbered Interface 17

IPv4 ICMP Rate Limiting 19

IPv6 ICMP Rate Limiting 20

Selecting Flexible Source IP 21

Configuring IPARM Conflict Resolution 21

Static Policy Resolution 22

Longest Prefix Address Conflict Resolution 23

Highest IP Address Conflict Resolution 23

Route-Tag Support for Connected Routes 24

Larger IPv6 Address Space 24

IPv6 Address Formats 25

IPv6 Address Type: Unicast 26

Aggregatable Global Address 26

Link-Local Address 27

IPv4-Compatible IPv6 Address 28

Simplified IPv6 Packet Header 28
Path MTU Discovery for IPv6 31
IPv6 Neighbor Discovery 32
IPv6 Neighbor Solicitation Message 32
IPv6 Router Advertisement Message 33
IPv6 Neighbor Redirect Message 35
Address Repository Manager 36
Address Conflict Resolution 36

CHAPTER 2 **Configuring ARP 37**
Configuring ARP 37
ARP Cache Entries 38
Defining a Static ARP Cache Entry 38
Proxy ARP and Local Proxy ARP 38
Enabling Proxy ARP 39
Enabling Local Proxy ARP 40
Configure Learning of Local ARP Entries 41
Information About Configuring ARP 42
Addressing Resolution Overview 42
Address Resolution on a Single LAN 42
Address Resolution When Interconnected by a Router 43
Policing Duplicate ARP Packets 44

CHAPTER 3 **Implementing Cisco Express Forwarding 45**
Implementing Cisco Express Forwarding 45
Verifying CEF 46
Per-Flow Load Balancing 49
Configuring Static Route 55
BGP Attributes Download 56
PPPoE 57
Configure PPPoE 58

CHAPTER 4 **Implementing Host Services and Applications 59**
Implementing Host Services and Applications 59

Network Connectivity Tools 59
Ping 59
Checking Network Connectivity 60
Checking Network Connectivity for Multiple Destinations 61
Traceroute 62
Checking Packet Routes 62
Domain Services 63
Configuring Domain Services 63
TFTP Server 64
Configuring a Router as a TFTP Server 65
File Transfer Services 65
FTP 66
Configuring a Router to Use FTP Connections 66
TFTP 66
Configuring a Router to Use TFTP Connections 67
SCP 67
Transferring Files Using SCP 68
Cisco inetd 68
Telnet 68
Syslog source-interface 69

CHAPTER 5 **Introduction to DHCP** 71
Prerequisites for Configuring DHCP Relay Agent 72
Limitations for DHCP Relay Feature 72
DHCP Option-82 with Circuit ID and Remote ID 72
DHCPv4 Relay Agent and Proxy Support for Segment Routing over IPv6 IPv4 L3VPN 73
How to Configure and Enable DHCP Relay Agent 74
Configuring and Enabling the DHCP Relay Agent 74
Enabling DHCPv6 Relay Agent on an Interface 75
Disabling DHCP Relay on an Interface 75
Configure a DHCP Relay Profile with Multiple Helper Addresses 75
DHCP Relay Agent Notification for Prefix Delegation 76
Configuring DHCP Stateful Relay Agent for Prefix Delegation 76
DHCPv6 Relay Over BVI for IANA Address Allocation 77

DHCP Relay Profile: Example 80
DHCP Relay on an Interface: Example 80
DHCP Relay on a VRF: Example 80
Relay Agent Information Option Support: Example 81
Relay Agent Giaddr Policy: Example 81
Configure a DHCP Proxy Profile 81
Configuring and Enabling the DHCP Relay Option-82 82
DHCP Server 83
Configuring DHCP Server Profile 84
Configuring Multiple Classes with a Pool 84
Excluding a Range of Addresses from DAPS 85
Configuring a Server Profile DAPS with Class Match Option 85
Configuring Server Profile without DAPS Pool Match Option 86
Configuring an Address Pool for Each ISP on DAPS 87
DHCP Client 88
Enabling DHCP Client on an Interface 88
DHCP Proxy Binding Table Reload Persistency 89
Configuring DHCP Relay Binding Database Write to System Persistent Memory 89
Jumbo Packet Handling for DHCPv6 90

CHAPTER 6 **Implementing Access Lists and Prefix Lists** 91
Understanding Access Lists 91
User-Defined TCAM Keys for IPv4 and IPv6 95
User-Defined Fields 96
IPv4 and IPv6 Key Formats for Traditional Ingress ACL 97
Configuring IPv4 ACLs 98
Configuring IPv6 ACLs 102
Single Pass IPv6 Egress ACL 107
Restrictions for Enabling Single-Pass IPv6 Egress ACL 107
Enable Single-Pass IPv6 Egress ACL 108
Configuring Chained ACLs 108
Modifying ACLs 110
Configuring ACL-based Forwarding 111
ACLs on Bridge Virtual Interfaces 114

Configuring ACLs with Fragment Control 116
Configuring an IPv4 ACL to Match on Fragment Type 118
Configuring ACL Filtering by IP Packet Length 120
Configuring Simple IPv4 ACLs to Filter by Packet Length 120
Configuring Simple IPv6 ACLs to Filter by Packet Length 121
Understanding Object-Group ACLs 122
Configuring an Object-Group ACL 123
Configuring TTL Matching and Rewriting for IPv4 ACLs 124
Configuring TTL Matching and Rewriting for IPv6 ACLs 125
Understanding IP Access List Logging Messages 126
Understanding Prefix Lists 129
Configuring Prefix Lists 129
Sequencing Prefix List Entries and Revising the Prefix List 130

CHAPTER 7 **Implementing HSRP 133**
Prerequisites for Implementing HSRP 133
Guidelines and Restrictions for Implementing HSRP 133
Information About Implementing HSRP 133
HSRP Overview 134
HSRP Groups 134
HSRP and ARP 136
Preemption 137
ICMP Redirect Messages 137
How to Implement HSRP 137
Enabling HSRP 137
Enabling HSRP for IPv6 139
Configuring HSRP Group Attributes 140
Configuring the HSRP Activation Delay 142
Disabling HSRP Support for ICMP Redirect Messages 143
Multiple Group Optimization (MGO) for HSRP 145
Customizing HSRP 145
Configuring a Primary Virtual IPv4 Address 146
Configuring a Secondary Virtual IPv4 Address 147
Configuring the Subordinate Group to Inherit its State from a Specified Group 148

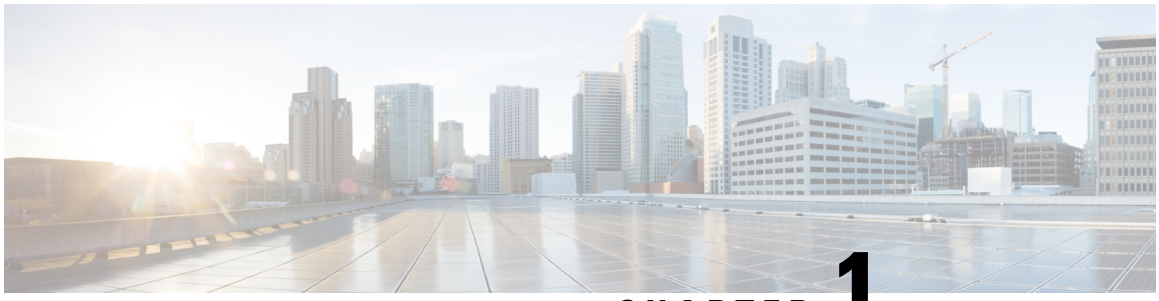
Configuring a Subordinate Primary Virtual IPv4 Address 149
Configuring a Secondary Virtual IPv4 address for the Subordinate Group 150
Configuring a Subordinate Virtual MAC Address 150
Configuring an HSRP Session Name 151
BFD for HSRP 152
Advantages of BFD 152
Guidelines and Restrictions for BFD for HSRP (Concept) 152
Enabling BFD 152
Modifying BFD timers (minimum interval) 153
Modifying BFD timers (multiplier) 153
Enhanced Object Tracking for HSRP and IP Static 154
Configuring object tracking for HSRP 154
Hot Restartability for HSRP 155
Configuration Examples for HSRP Implementation on Software 155
Configuring an HSRP Group: Example 155
Configuring a Router for Multiple HSRP Groups: Example 155

CHAPTER 8 **Implementing LPTS** 157
LPTS Overview 157
LPTS Policers 157
LPTS Domain Based Policers 162

CHAPTER 9 **Implementing VRRP** 165
Configuring VRRP 165
Understanding VRRP 165
Customizing VRRP 169
Enabling VRRP 170
Configuring a Global Virtual IPv6 Address 171
Configuring the Primary and Secondary Virtual IPv4 Addresses 172
Configuring a Virtual Link-Local IPv6 Address 172
BFD for VRRP 173
Advantages of BFD 173
BFD Process 174
Configuring BFD 174

Disabling State Change Logging 175
Enabling Multiple Group Optimization (MGO) for VRRP 175
Configuring SNMP Server Notifications for VRRP Events 177

CHAPTER 10 **Information About Configuring TCP, UDP Transports** 179
NSR Overview 179
Graceful Restart 180
TCP Overview 180
UDP Overview 180
Configuring Failover as a Recovery Action for NSR 180



CHAPTER 1

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, on page 1
- IPv4 and IPv6 Functionality, on page 2
- Custom Prefix Length Selection , on page 2
- IPv6 for Cisco IOS XR Software, on page 9
- How to Implement Network Stack IPv4 and IPv6, on page 9

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.

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.

Custom Prefix Length Selection

Feature Name	Release Information	Feature Description
Custom Prefix Length Selection	Release 7.4.1	By default, /48 prefix length is inserted in the LEM memory. This feature allows you to choose a custom IPv6 prefix length to be inserted into the largest exact match (LEM) memory. This feature introduces the **hw-module fib scale ipv6 custom-lem** command.

By default, /48 prefix length is inserted in the LEM memory. This feature allows you to choose a custom IPv6 prefix length to be inserted into the largest exact match (LEM) memory.

Restrictions

- Do not configure the IPv6 **internet-optimized-disable** command and the **hw-module custom-lem** command together.
- You can configure only one single length at a time. You can choose only one prefix length value to be put into the LEM memory.
- Make sure that the IPv6 length that you chose is nibble granular, that is multiples of 4.

Configuration Example

```
Router(config)# hw-module fib scale ipv6 custom-lem
```

Running Configuration

```
hw-module fib scale ipv6 custom-lem
```

Verification

Verify the prefix distribution with different prefix lengths.

```
Router# show dpa resources ip6route location 0/0/CPU0
Fri Jul  9 15:33:00.652 UTC

"ip6route" OFA Table (Id: 53, Scope: Global)
------------------------------------------------
IPv6 Prefix len distribution
Prefix   Actual       Prefix   Actual
/0       1            /1       0
 /2       0            /3       0
 /4       0            /5       0
 /6       0            /7       0
 /8       0            /9       0
 /10      1            /11      0
 /12      0            /13      0
 /14      0            /15      0
 /16      3            /17      0
 /18      0            /19      0
 /20      0            /21      0
 /22      0            /23      0
 /24      0            /25      0
 /26      0            /27      0
 /28      0            /29      0
 /30      0            /31      0
 /32      0            /33      0
 /34      0            /35      0
 /36      0            /37      0
 /38      0            /39      0
 /40      0            /41      0
 /42      0            /43      0
 /44      0            /45      0
 /46      0            /47      0
 /48      0            /49      0
 /50      0            /51      0
 /52      0            /53      0
 /54      0            /55      0
 /56      0            /57      0
 /58      0            /59      0
 /60      0            /61      0
 /62      0            /63      0
 /64      0            /65      0
 /66      0            /67      0
 /68      0            /69      0
 /70      0            /71      0
 /72      0            /73      0
 /74      0            /75      0
 /76      0            /77      0
 /78      0            /79      0
 /80      0            /81      0
 /82      0            /83      0
 /84      0            /85      0
 /86      0            /87      0
 /88      0            /89      0
 /90      0            /91      0
 /92      0            /93      0
 /94      0            /95      0
 /96      0            /97      0
 /98      0            /99      0
 /100     0            /101     0
 /102     0            /103     0
 /104     1            /105     0
 /106     0            /107     0
```

```
 /108     0            /109     0
 /110     0            /111     0
 /112     0            /113     0
 /114     0            /115     0
 /116     0            /117     0
 /118     0            /119     0
 /120     0            /121     0
 /122     0            /123     0
 /124     0            /125     0
 /126     0            /127     0
 /128     1

OFA Infra Stats Summary
                 Create Requests: 7
                 Delete Requests: 0
                 Update Requests: 0
                    Get Requests: 0

                  Backwalk Stats
                 Update Requests: 0
                  Update Skipped: 0

                          Errors
               Resolve Failures: 0
                Not Found in DB: 0
                   Exists in DB: 0
                No Memory in DB: 0
              Reserve Resources: 0
              Release Resources: 0
               Update Resources: 0
                 Retry Attempts: 0
          Recovered from error: 0
             Errors from bwalk: 0

                          NPU ID: NPU-0
         Create Server API Err: 0
         Update Server API Err: 0
         Delete Server API Err: 0
```

show dpa resources ip6route location 0/0/CPU0

```
Fri Jul  9 17:42:09.728 UTC

"ip6route" OFA Table (Id: 53, Scope: Global)
------------------------------------------------
IPv6 Prefix len distribution
Prefix   Actual       Prefix   Actual
 /0       1            /1       0
 /2       0            /3       0
 /4       0            /5       0
 /6       0            /7       0
 /8       0            /9       0
 /10      1            /11      0
 /12      0            /13      0
 /14      0            /15      0
 /16      3            /17      0
 /18      0            /19      0
 /20      0            /21      0
 /22      0            /23      0
 /24      0            /25      0
 /26      0            /27      0
 /28      0            /29      0
 /30      0            /31      0
 /32      0            /33      0
```

```
 /34      0            /35      0
 /36      0            /37      0
 /38      0            /39      0
 /40      0            /41      0
 /42      0            /43      0
 /44      0            /45      0
 /46      0            /47      0
 /48      0            /49      0
 /50      0            /51      0
 /52      0            /53      0
 /54      0            /55      0
 /56      0            /57      0
 /58      0            /59      0
 /60      0            /61      0
 /62      0            /63      0
 /64      2055         /65      0           >>>>>>>>> total prefixes with /64 length
 /66      0            /67      0
 /68      0            /69      0
 /70      0            /71      0
 /72      0            /73      0
 /74      0            /75      0
 /76      0            /77      0
 /78      0            /79      0
 /80      0            /81      0
 /82      0            /83      0
 /84      0            /85      0
 /86      0            /87      0
 /88      0            /89      0
 /90      0            /91      0
 /92      0            /93      0
 /94      0            /95      0
 /96      0            /97      0
 /98      0            /99      0
 /100     0            /101     0
 /102     0            /103     0
 /104     1            /105     0
 /106     0            /107     0
 /108     0            /109     0
 /110     0            /111     0
 /112     0            /113     0
 /114     0            /115     0
 /116     0            /117     0
 /118     0            /119     0
 /120     0            /121     0
 /122     0            /123     0
 /124     0            /125     0
 /126     0            /127     0
 /128     29

OFA Infra Stats Summary
                 Create Requests: 2090
                 Delete Requests: 0
                 Update Requests: 0
                    Get Requests: 0

                  Backwalk Stats
                 Update Requests: 0
                  Update Skipped: 0

                           Errors
                Resolve Failures: 0
                 Not Found in DB: 0
                    Exists in DB: 0
                 No Memory in DB: 0
```

```
               Reserve Resources: 0
               Release Resources: 0
                Update Resources: 0
                  Retry Attempts: 0
             Recovered from error: 0
               Errors from bwalk: 0

                          NPU ID: NPU-0
          Create Server API Err: 0
          Update Server API Err: 0
          Delete Server API Err: 0
```

Verify the configured prefix LEM length. In the below example, the configured LEM length is indicated as 48.

```
Router# show controller fia diagshell 0 "config" location 0/0/CPU0
Fri Jul  9 15:31:45.616 UTC
    custom_feature_bfd_ipv6_protection=1
    bfd_ipv6_trap_port=208
    bcm886xx_ipv6_tunnel_enable=1
    custom_feature_li_ipv6_disable=1
    bcm886xx_ipv6_ext_hdr_enable=1
    enhanced_fib_scale_prefix_length_ipv6_long=48
    mcs_load_uc0=bfd_ipv6
    l3_vrrp_ipv6_distinct=1
    custom_feature_kbp_ipv6_uc_no_rpf_dip_sip_sharing_from_fwd_header=1
    enhanced_fib_scale_prefix_length_ipv6_short=48

    bfd_ipv6_enable=1
```

Verify the configured prefix LEM length. In the below example, the configured LEM length is indicated as 64.

```
show controller fia diagshell 0 "config" location 0/0/cpu0 | i ipv6
Fri Jul  9 17:41:39.518 UTC
    custom_feature_bfd_ipv6_protection=1
    bfd_ipv6_trap_port=208
    bcm886xx_ipv6_tunnel_enable=1
    custom_feature_li_ipv6_disable=1
    bcm886xx_ipv6_ext_hdr_enable=1
    enhanced_fib_scale_prefix_length_ipv6_long=64
    mcs_load_uc0=bfd_ipv6
    l3_vrrp_ipv6_distinct=1
    custom_feature_kbp_ipv6_uc_no_rpf_dip_sip_sharing_from_fwd_header=1
    enhanced_fib_scale_prefix_length_ipv6_short=64
    bfd_ipv6_enable=1
```

Verify the usage of resources.

```
Routers# show controllers npu resources lem loc 0/5/CPU0
Mon Jul 12 16:17:48.751 UTC
HW Resource Information
    Name                            : lem
    Asic Type                       : Jericho Plus

NPU-0
OOR Summary
        Estimated Max Entries       : 786432
        Red Threshold               : 95 %
        Yellow Threshold            : 80 %
        OOR State                   : Green


Current Usage
        Total In-Use                : 26       (0 %)
```

```
       iproute                     : 0        (0 %)
        ip6route                    : 0        (0 %)
        mplslabel                   : 1        (0 %)
        l2brmac                     : 0        (0 %)


NPU-1
OOR Summary
        Estimated Max Entries       : 786432
        Red Threshold               : 95 %
        Yellow Threshold            : 80 %
        OOR State                   : Green


Current Usage
        Total In-Use                : 26       (0 %)
        iproute                     : 0        (0 %)
        ip6route                    : 0        (0 %)
        mplslabel                   : 1        (0 %)
        l2brmac                     : 0        (0 %)


NPU-2
OOR Summary
        Estimated Max Entries       : 786432
        Red Threshold               : 95 %
        Yellow Threshold            : 80 %
        OOR State                   : Green


Current Usage
        Total In-Use                : 26       (0 %)
        iproute                     : 0        (0 %)
        ip6route                    : 0        (0 %)
        mplslabel                   : 1        (0 %)
        l2brmac                     : 0        (0 %)


NPU-3
OOR Summary
        Estimated Max Entries       : 786432
        Red Threshold               : 95 %
        Yellow Threshold            : 80 %
        OOR State                   : Green


Current Usage
        Total In-Use                : 26       (0 %)
        iproute                     : 0        (0 %)
        ip6route                    : 0        (0 %)
        mplslabel                   : 1        (0 %)
        l2brmac                     : 0        (0 %)
```

Verify the usage of resources.

```
show controllers npu resources lem location 0/0/CPU0
Fri Jul  9 17:42:34.516 UTC
HW Resource Information
    Name                        : lem
    Asic Type                   : Jericho

NPU-0
OOR Summary
        Estimated Max Entries       : 786432
```

```
        Red Threshold               : 95 %
        Yellow Threshold            : 80 %
        OOR State                   : Green


Current Usage
        Total In-Use                : 4223     (1 %)
        iproute                     : 2172     (0 %)
        ip6route                    : 2055     (0 %). >>>>>>>> LEM resources allocation
should = dpa resources for /64
        mplslabel                   : 1        (0 %)
        l2brmac                     : 0        (0 %)
```

Verify the summary of the CEF table.

```
show cef ipv6 summary
Fri Jul  9 17:51:02.788 UTC

Router ID is 192.168.1.3

IP CEF with switching (Table Version 0) for node0_RP0_CPU0

  Load balancing: L4
  Tableid 0xe0800000 (0x8a4f2748), Vrfid 0x60000000, Vrid 0x20000000, Flags 0x1019
  Vrfname default, Refcount 2179
  2090 routes, 0 protected, 0 reresolve, 0 unresolved (0 old, 0 new), 317680 bytes
    2083 rib, 0 lsd, 0 aib, 0 internal, 0 interface, 6 special, 1 default routes
    Prefix masklen distribution:
        unicast: 28 /128, 2055 /64 , 1 /10 , 1 /0  >>>>>>> cef prefixes received to LC
hardware
      multicast: 1 /128, 1 /104, 3 /16
  61 load sharing elements, 37968 bytes, 2090 references
  Shared load sharing elements with 9664 bytes, 2032 references, including:
    3 Pathlist elements, 0 recursive, 0 platform shared, 0 in retry
    3 Loadinfo elements, 0 recursive, 0 platform shared
  Exclusive load sharing elements with 28304 bytes, 58 references, including:
    58 Pathlist elements, 0 recursive, 0 platform shared, 0 in retry
    58 Loadinfo elements, 0 recursive, 0 platform shared
  0 Drop Pathlist elements
  0 route delete cache elements
  156 local route bufs received, 0 remote route bufs received,  0 mix bufs received
  2083 local routes, 0 remote routes
  2155 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, 50 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
  22 next hops
    0 incomplete next hops
  0 PD backwalks on LDIs with backup path
```

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, I*nternet Protocol, Version 6 (IPv6) Specification* issued by the Internet Engineering Task Force (IETF). Further RFCs describe the architecture and services supported by IPv6.

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 192.168.1.27 and a network mask of "/8" is assigned to the **`interface HundredGigE 0/9/0/1`**.

Note The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means 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 HundredGigE0/9/0/1
Router(config)#interface HundredGigE 0/9/0/1
Router(config-if)#ipv4 address 192.168.1.27/8
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE0/9/0/1
 ipv4 address 192.168.1.27 255.0.0.0
!
```

Verification

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

```
Router# show ipv4 interface HundredGigE0/9/0/1

interface HundredGigE0/9/0/1 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.168.1.27/8
  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
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  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
```

Associated Commands

- ipv4 address
- show ipv4 interface

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 maximum number of characters allowed for the name of the prefix is 128, and any invalid value will be rejected.

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.

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 **interface HundredGigE 0/9/0/1**:

```
Router#configure
Router(config)#interface HundredGigE 0/9/0/1
Router(config-if)#ipv6 address 2001:0DB8:0:1::1/64
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE0/9/0/1

interface HundredGigE0/9/0/1
 ipv4 address 192.168.1.27 255.0.0.0
 ipv4 address 1.0.0.1 255.255.255.0 secondary
 ipv4 address 2.0.0.1 255.255.255.0 secondary
 ipv6 address 2001:db8:0:1::1/64
!
```

Verification

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

```
Router#show ipv6 interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
  Global unicast address(es):
```

```
    2001:db8:0:1::1, subnet is 2001:db8:0:1::/64
  Joined group address(es): ff02::1:ff00:1 ff02::1:ffa6:1c75 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  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
```

Associated Commands

- ipv6 address
- interface
- show ipv6 interface

Configuration Example

An IPv6 address of 2001:0DB8:0:1::/64 is assigned to the **interface HundredGigE 0/9/0/1**. 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 0/9/0/1
Router(config-if)#ipv6 address 2001:0DB8:0:1::/64 eui-64
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE0/9/0/1

interface HundredGigE0/9/0/1
 ipv4 address 192.168.1.27 255.0.0.0
 ipv4 address 1.0.0.1 255.255.255.0 secondary
 ipv4 address 2.0.0.1 255.255.255.0 secondary
 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 HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
  Global unicast address(es):
    2001:db8:0:1:c672:95ff:fea6:1c75, subnet is 2001:db8:0:1::/64
```

```
  Joined group address(es): ff02::1:ffa6:1c75 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  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
```

Associated Commands

- ipv6 address
- interface
- show ipv6 interface

Configuration Example

An IPv6 address of FE80::260:3EFF:FE11:6770 is assigned to the **interface HundredGigE 0/9/0/1**. 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 0/9/0/1
Router(config-if)#ipv6 address FE80::260:3EFF:FE11:6770 link-local
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE0/9/0/1

interface HundredGigE0/9/0/1
 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 HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::260:3eff:fe11:6770
  Global unicast address(es):
    2001:db8:0:1:260:3eff:fe11:6770, subnet is 2001:db8:0:1::/64
  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
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  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
```

Associated Commands

- ipv6 address
- interface
- show ipv6 interface

Configuration Example

Enable IPv6 processing on the **interface HundredGigE 0/9/0/1**; that has not been configured with an explicit IPv6 address.

```
Router#configure
Router(config)#interface HundredGigE 0/9/0/1
Router(config-if)#ipv6 enable
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE0/9/0/1


interface HundredGigE0/9/0/1
ipv6 enable
!
```

Verification

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

```
Router#show ipv6 interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
  No global unicast address is configured
  Joined group address(es): ff02::1:ffa6:1c75 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  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
```

Associated Commands

- ipv6 enable
- interface
- show ipv6 interface

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 192.168.1.27 is assigned to the Hundredgige interface-0/0/0/1.

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

```
Router# configure
Router(config)# interface HundredGigE 0/9/0/1
Router(config-if)# ipv4 address 192.168.1.27 255.255.255.0 secondary
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE0/9/0/1

interface HundredGigE0/9/0/1
 ipv4 address 192.168.1.27 255.255.255.0 secondary
!
```

Verification

```
Router#show ipv4 interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is unassigned
  Secondary address 192.168.1.27/24
  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
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  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
```

Associated Commands

- ipv4 address
- show ipv4 interface

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 192.168.99.1 and an IPv6 address of 2001:0DB8:c18:1::3/64 is configured on the **interface HundredGigE 0/9/0/1**.

```
Router#configure
Router(config)#interface HundredGigE 0/9/0/1
Router(config-if)#ipv4 address 192.168.99.1 255.255.255.0
Router(config-if)#ipv6 address 2001:0DB8:c18:1::3/64
Router(config-if)#commit
```

Running Configuration

```
Router# show running-config interface HundredGigE0/9/0/1
 ipv4 address 192.168.99.1 255.255.255.0
 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 HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.168.99.1/24
  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
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  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 HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
  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:ffa6:1c75 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  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
```

Associated Commands

- ipv4 address
- ipv6 address
- show ipv4 interface
- show ipv6 interface

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:

- Interfaces using High-Level Data Link Control (HDLC), PPP, and Frame Relay encapsulations can be unnumbered. Serial interfaces using Frame Relay encapsulation can also be unnumbered, but the interface must be a point-to-point sub-interface.
- 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.

If you are configuring Intermediate System-to-Intermediate System (IS-IS) across a serial line, you should configure the serial interfaces as unnumbered, which allows you to conform with RFC 1195, which states that IP addresses are not required on each 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 0/9/0/1
Router(config-if)#ipv4 unnumbered loopback 0
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE0/9/0/1
interface HundredGigE0/9/0/1
 ipv4 point-to-point
 ipv4 unnumbered Loopback0
!
```

Verification

```
Router#show interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 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 10.0.0.2/32
  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 run int lo 0
interface Loopback0
 ipv4 address 10.0.0.2 255.255.255.255
```

Associated Commands

- ipv4 unnumbered
- show interfaces

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 HundredGigE0/9/0/1
HundredGigE0/9/0/1 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
```

Associated Commands

- icmp ipv4 rate-limit unreachable
- show ipv4 traffic

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

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

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 1.1.1.1 255.255.255.255
!
…..
```

Verification

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

F tenGigE 0/11/0/0 192.85.1.2/24   HundredGigE0/9/0/1  192.85.1.1/24 default

Forced down interface         Up interface                   VRF
```

Associated Commands

- ipv4 conflict-policy
- ipv6 conflict-policy

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 tenGigE 0/11/0/0 192.85.1.2/24   HundredGigE0/9/0/1  192.85.1.1/24 default

Forced down interface         Up interface                   VRF
```

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 tenGigE 0/11/0/0 192.85.1.2/24   HundredGigE0/9/0/1  192.85.1.1/24 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 10.0.54.2/30 that has a route tag of 20 to the **interface HundredGigE 0/9/0/1**.

```
Router#configure
Router(config)#interface HundredGigE 0/9/0/1
Router(config-if)#ipv4 address 10.0.54.2/30 route-tag 1899
Router(config)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE0/9/0/1

interface HundredGigE0/9/0/1
 ipv4 address 10.0.54.2/30 route-tag 1899
!
```

Verification

Verify the parameters of the route.

```
Router#show route 10.0.54.2
Routing entry for 10.0.54.2/32
  Known via "local", distance 0, metric 0 (connected)
  Tag 1899
Routing Descriptor Blocks
    directly connected, via HundredGigE0/9/0/1
      Route metric is 0
  No advertising protos.
```

Associated Commands

- route-tag

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 1: Compressed IPv6 Address Formats, on page 25 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 1: 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 1: Compressed IPv6 Address Formats, on page 25 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 1: Compressed IPv6 Address Formats, on page 25 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:

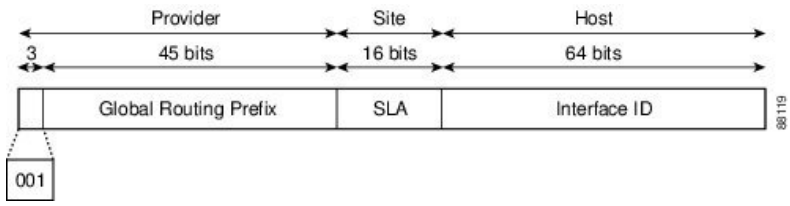
- Global aggregatable address
- Site-local address (proposal to remove by IETF)
- Link-local address
- IPv4-compatible IPv6 address

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:

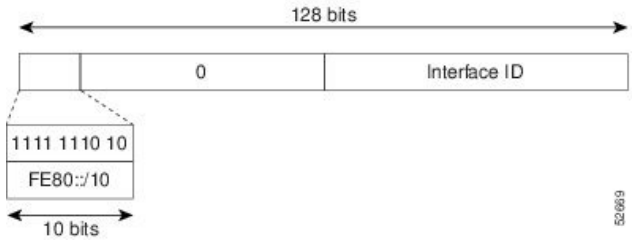
1. The router is queried for MAC addresses (from the pool of MAC addresses in the router).
2. If no MAC address is available, the serial number of the Route Processor (RP) or line card (LC) is used to form the link-local address.

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.

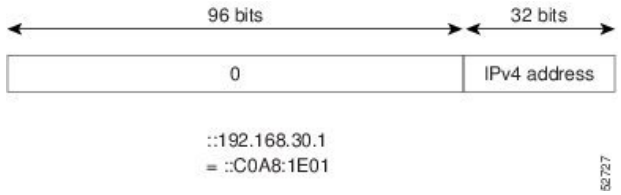
Figure 2: Link-Local Address Format



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.

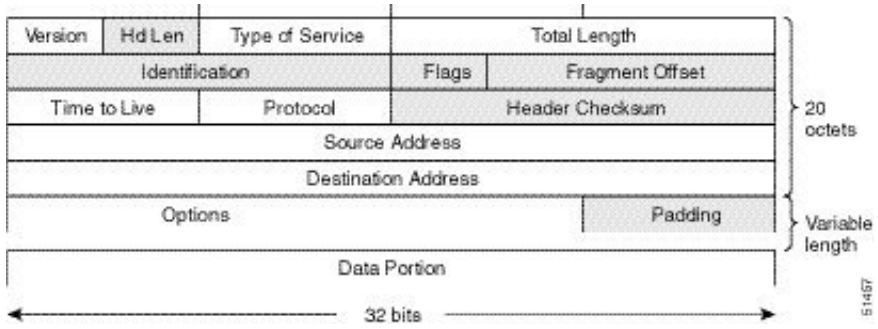
Figure 3: IPv4-Compatible IPv6 Address Format



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.

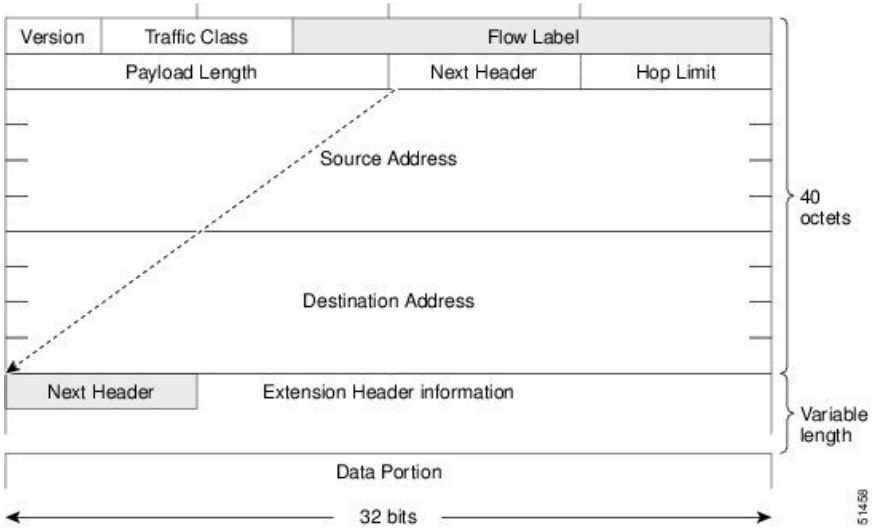
Figure 4: IPv4 Packet Header Format



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.

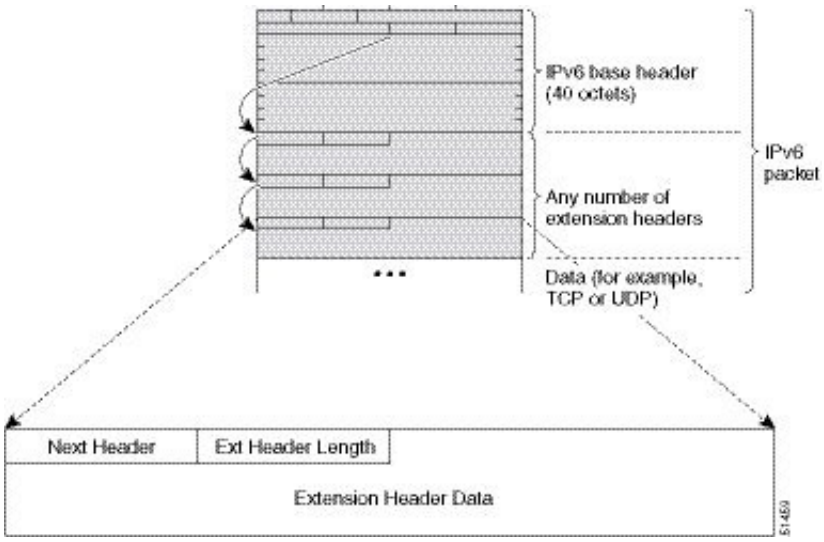
Table 2: Basic IPv6 Packet Header Fields

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.

Field	Description
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.
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 3: 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.

Header Type	Next Header Value	Description
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.
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.

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

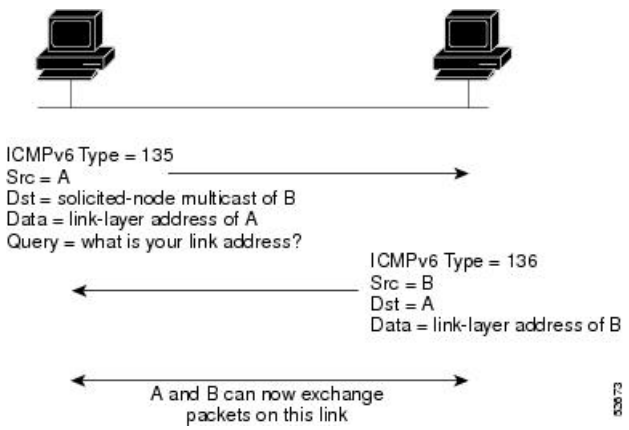
The IPv6 neighbor discovery (ND) process uses ICMP messages and solicited-node multicast addresses to determine the link-layer address of a neighbor on the same network (local link), verify the reachability of a neighbor, and keep track of neighboring routers.

As all incoming control traffic goes through LPTS policer, if the ND packets come in a burst they are policed according to the configuration. For more details on LPTS, see LPTS Overview, on page 157.

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.

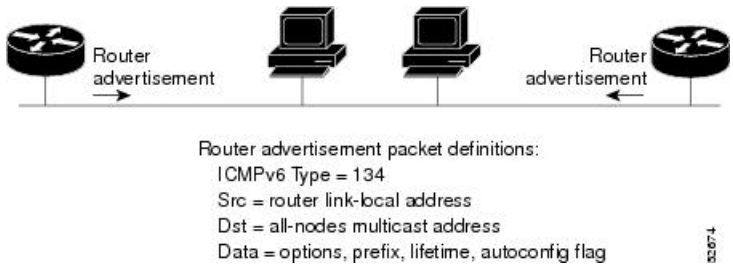
Neighbor solicitation messages are also used in the stateless autoconfiguration process to verify the uniqueness of unicast IPv6 addresses before the addresses are assigned to an interface. Duplicate address detection is performed first on a new, link-local IPv6 address before the address is assigned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

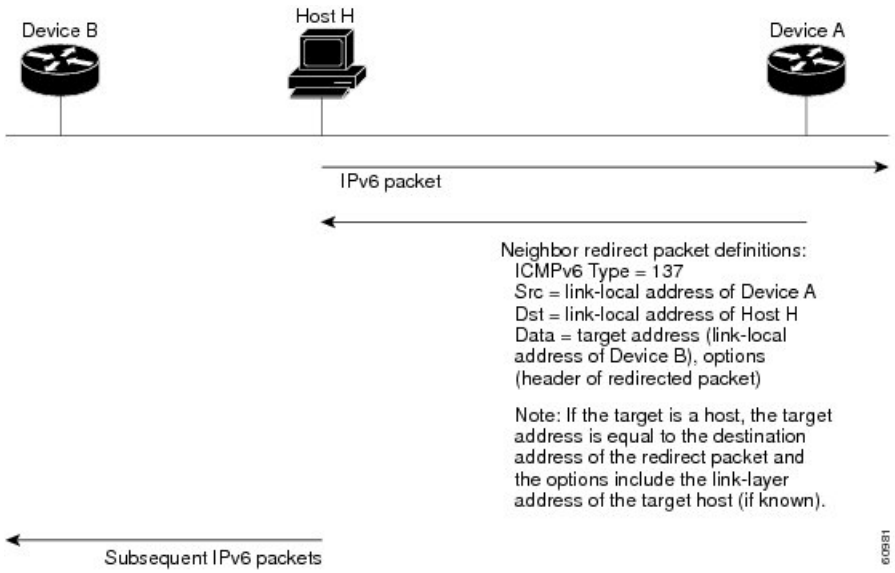
The configured parameters are specific to an interface. The sending of router advertisement messages (with default values) is automatically enabled on Ethernet and FDDI interfaces. For other interface types, the sending of router advertisement messages must be manually configured by using the **no ipv6 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.

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

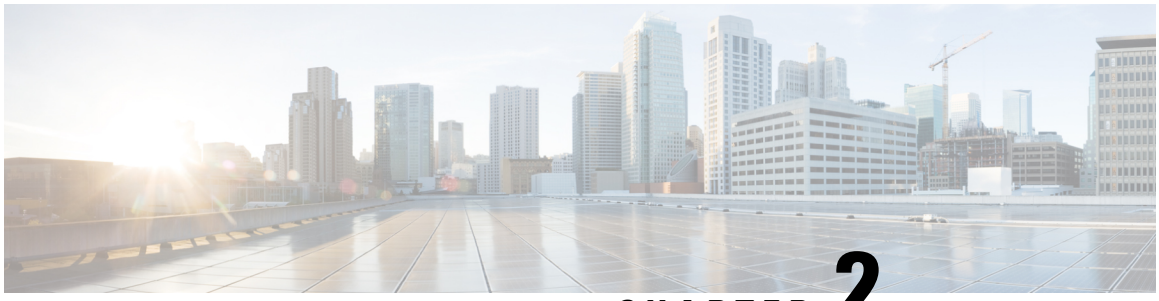
address 3: 10.3.1.1/16

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.



CHAPTER 2

Configuring ARP

- Configuring ARP, on page 37
- Information About Configuring ARP, on page 42
- Policing Duplicate ARP Packets, on page 44

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.

As all incoming control traffic goes through LPTS policer, if the ARP packets come in a burst they are policed according to the configuration. For more details on LPTS, see LPTS Overview, on page 157.

For more details on ARP, see Information About Configuring ARP, on page 42

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 38

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.

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 **203.0.1.2** 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 203.0.1.2 0010.9400.000c arPA
Router(config)#commit
```

Running Configuration

```
Router#show run arp 203.0.1.2 0010.9400.000c arpA
arp vrf default 203.0.1.2 0010.9400.000c ARPA
```

Verification

Verify that the State is static for proper functioning:

```
Router#show arp location 0/RP0/CPU0
Address         Age        Hardware Addr   State      Type  Interface
203.0.1.1       -          ea28.5f0b.8024  Interface  ARPA  HundredGigE0/9/0/0
203.0.1.2       -          0010.9400.000c  Static ARPA  HundredGigE0/9/0/0
```

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-0/9/0/0:

```
Router#configure
Router(config)#interface HundredGigE0/9/0/0
Router(config-if)#proxy-arp
Router(config-if)#commit
```

Running Configuration

```
Router# show running-config interface HundredGigE0/9/0/0
 mtu 4000
 ipv4 address 1.0.0.1 255.255.255.0
 proxy-arp
!
!
```

Verification

Verify that proxy ARP is configured and enabled:

```
Router# show arp idb interface HundredGigE0/9/0/0(0x08000038):
  IPv4 address 1.0.0.1, 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-0/9/0/0

```
Router#configure
Router(config)#interface HundredGigE0/9/0/0
Router(config-if)#local-proxy-arp
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface HundredGigE0/9/0/1

ipv4 address 1.0.0.1 255.255.255.0
local-proxy-arp
!
```

Verification

Verify that local proxy ARP is configured:

```
Router# show arp idb interface HundredGigE0/9/0/0 location 0/RP0/CPU0
 (0x08000038):
  IPv4 address 1.0.0.1, 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

Configure Learning of Local ARP Entries

You can configure an interface or a sub-interface to learn only the ARP entries from its local subnet.

Use the following procedure to configure local ARP learning on an interface.

1. Enter the interface configuration mode.

```
Router(config)# interface TenGigE 0/11/0/0
```

2. Configure the IPv4/IPv6 address for the interface.

```
Router(config-if)# ipv4 address 12.1.3.4 255.255.255.0
```

3. Configure local ARP learning on the interface.

```
Router(config-if)# arp learning local
```

4. Enable the interface and commit your configuration.

```
Router(config-if)# no shut
Router(config-if)# commit
RP/0/0/CPU0:Dec 12 13:41:16.580 : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : interface TenGigE
 0/11/0/0, changed state to Down
RP/0/0/CPU0:Dec 12 13:41:16.683 : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : interface TenGigE
 0/11/0/0 changed state to Up
```

5. Confirm your configuration.

```
Router(config-if)# show running-configuration
..
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Mon Dec 12 13:41:16 2016
!interface TenGigE 0/11/0/0
 ipv4 address 12.1.3.4 255.255.255.0
 arp learning local
!
```

6. Verify if local ARP learning is working as configured on the interface.

```
Router(config-if)# do show arp idb TenGigE 0/11/0/0 location 0/RP0/CPU0
Thu Dec 15 10:00:11.733 IST

TenGigE 0/11/0/0 (0x00000040):
  IPv4 address 12.1.3.4, Vrf ID 0x60000000
  VRF Name default
  Dynamic learning: Local
  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 not configured
  Packet IO layer is NetIO
  Srg Role : DEFAULT
  Idb Flag : 2146444
  IDB is Complete
```

7. (Optional) You can monitor the ARP traffic on the interface.

```
Router(config-if)# do show arp idb TenGigE 0/11/0/0 location 0/RP0/CPU0
Thu Dec 15 10:13:28.964 IST
```

```
ARP statistics:
  Recv: 0 requests, 0 replies
  Sent: 0 requests, 1 replies (0 proxy, 0 local proxy, 1 gratuitous)
  Subscriber Interface:
         0 requests recv, 0 replies sent, 0 gratuitous replies sent
  Resolve requests rcvd: 0
  Resolve requests dropped: 0
  Errors: 0 out of memory, 0 no buffers, 0 out of sunbet

ARP cache:
  Total ARP entries in cache: 1
  Dynamic: 0, Interface: 1, Standby: 0
  Alias: 0,   Static: 0,    DHCP: 0

  IP Packet drop count for GigabitEthernet0_0_0_1: 0
```

Information About Configuring ARP

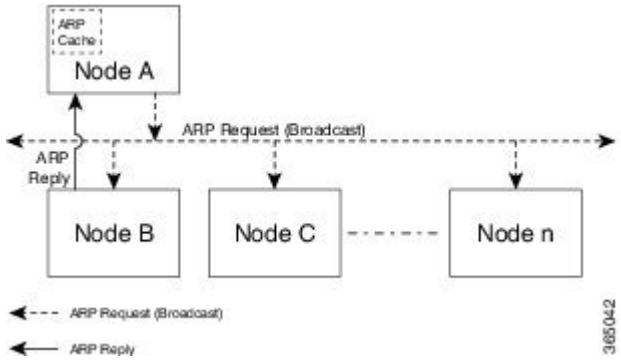
Addressing Resolution Overview

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

The following process describes address resolution when the source and destination devices are attached to the same LAN:

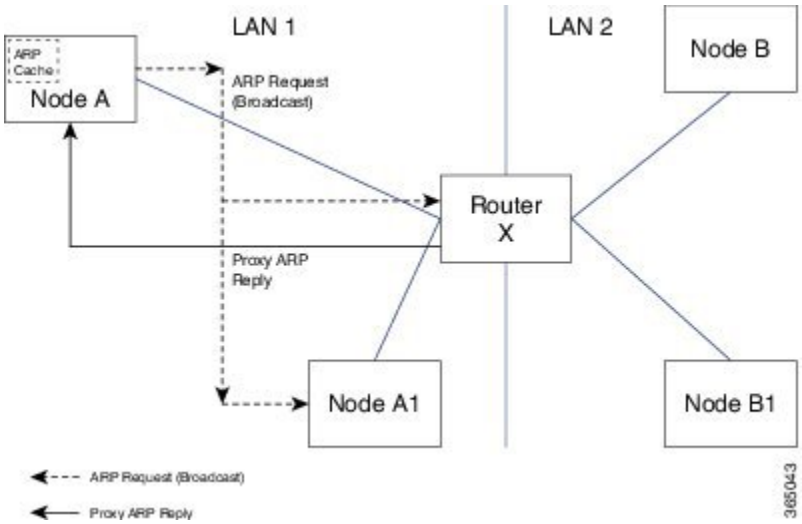


1. End System A (Node A) broadcasts an ARP request onto the LAN, attempting to learn the MAC address of End System B (Node B).
2. The broadcast is received and processed by all devices on the LAN, including End System B.

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.)
5. Whenever End System A (Node A) needs to communicate with End System B, it checks the ARP cache, finds the MAC address of System B, and sends the frame directly, without needing to first use an ARP request.

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.

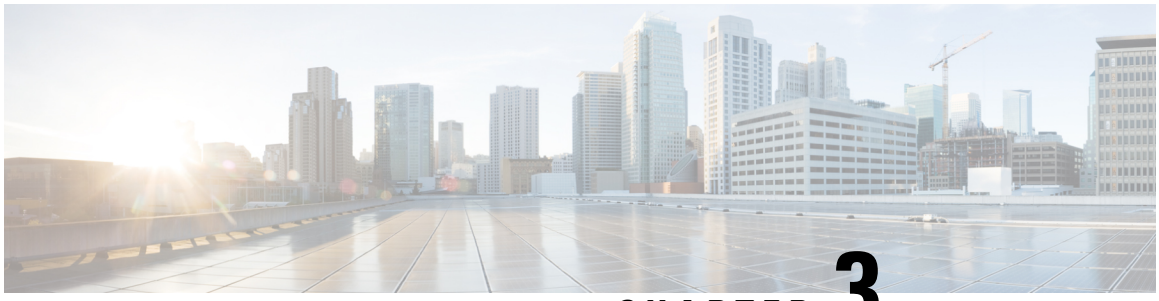
Policing Duplicate ARP Packets

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.



CHAPTER 3

Implementing Cisco Express Forwarding

- Implementing Cisco Express Forwarding, on page 45

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.

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 . The FIB on each node processes Routing Information Base (RIB) updates, performing route resolution and maintaining FIB tables independently in the route processor . 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
- MPLS LFD database—Stores MPLS Label table for forwarding MPLS packets

Verifying CEF

To view the details of the IPv4 or IPv6 CEF tables, use the following commands:

- show cef {*ipv4 address* | *ipv6 address*} hardware egress npu location {node-id}

Displays the IPv4 or IPv6 CEF table. The next hop and forwarding interface are displayed for each prefix. The output of the **show cef** command varies by location.

```
Router# show cef 203.0.1.2 hardware egress
  203.0.1.2/32, version 0, internal 0x1020001 0x0 (ptr 0x8d7db7f0) [1], 0x0 (0x8daeedf0),
0x0 (0x0)
 Updated Nov 20 13:33:23.557
 local adjacency 203.0.1.2
 Prefix Len 32, traffic index 0, Adjacency-prefix, precedence n/a, priority 15
   via 203.0.1.2/32, HundredGigE0/9/0/0, 3 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8cfc81a0 0x0]
    next hop 203.0.1.2/32
    local adjacency
```

```
Router# show cef ipv6 fccc:0:2:ce00:: hardware egress npu location 0/0/CPU0
Using NPU option is resource intensive and may result in system
instability and possibly result in traffic loss.
Do you really want to continue[confirm with only 'y' or 'n'] [y/n] : y
Do you really want to continue[confirm with only 'y' or 'n'] [y/n] :Do you really want to
continue[confirm with only 'y' or 'n'] [y/n] : y
Do you really want to continue[confirm with only 'y' or 'n'] [y/n] :Do you really want to
continue[confirm with only 'y' or 'n'] [y/n] :y
fccc:0:2:ce00::/64, version 725, SRv6 Headend, internal 0x1000001 0x220 (ptr 0x8dace0a8)
[1], 0x400 (0x8daa1138), 0x0 (0x8e664250)
 Updated Oct 12 20:35:51.433
 local adjacency to Bundle-Ether1301

 Prefix Len 64, traffic index 0, precedence n/a, priority 2
```

```
  gateway array (0x8d93b398) reference count 2, flags 0x0, source rib (7), 0 backups
                [4 type 3 flags 0x8401 (0x8d9efe68) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8daa1138, sh-ldi=0x8d9efe68]
  gateway array update type-time 1 Oct 10 11:22:17.595
 LDI Update time Oct 10 11:22:17.595
 LW-LDI-TS Oct 12 20:35:51.450
 Accounting: 0/0 packets/bytes output (per-prefix-per-path mode)
   via fe80::11/128, Bundle-Ether1301, 13 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x20049 [0x901b88b0 0x0]
    next hop fe80::11/128
    local adjacency
    Accounting: 0/0 packets/bytes output


 LEAF - HAL pd context :
 sub-type : IPV6, ecd_marked:0, has_collapsed_ldi:0
 collapse_bwalk_required:0, ecdv2_marked:0,
HW Walk:
LEAF:
    PI:0x308dace0a8 PD:0x308dace140 rev:2474 type: IPV6 (1) TBL: 0xe0800000
    LEAF location: LPM
    FEC key: 0x14b40000a0d0
    LEAF ip6route HW: <<<< New information
            npu:0x0 fec index:0x2000ffce

    LWLDI:
        PI:0x308daa1138 PD:0x308daa1180 rev:2473 p-rev:2381  ldi type:IP
        FEC key: 0x14b40000a0d0 fec index: 0x2000ffce(65486) num paths: 1 bkup paths: 0
        IMP pattern:3
        PI:0x308daa1138 PD:0x308daa1180 rev:2473 p-rev:2381  dpa-rev:1070462
        FEC key: 0x14b40000a0d0 fec index: 0x2000ffce(65486) num paths: 1 bkup paths: 0
        Path:0  fec index: 0x2000ffce(65486) DSP: 0xc000001
                SRV6 Encap key: 0xf1c0000040011844 SRV6 Encap Id: 0x40011844 Remote: 0
                Alloc Size: 1 SID:  Func: 48 (LOC_STATS)
                Hardware Programmed SID: {}
                uSID Shift slots:0
        FEC:
        npu:0x0 fec:0x2000ffce port:0xc000001 encap:0x40011844

        SHLDI:
            PI:0x308d9efe68 PD:0x308d9eff10 rev:2381 dpa-rev:195026 cbf_enabled:0
pbts_enabled:0 surpf_enable:0 flag:0x0
            FEC key: 0x14340000a0d0 fec index: 0x2000ffc5(65477) num paths: 1 bkup paths:
0
            p-rev:2364
            Path:0 fec index: 0x2000ffc5(65477) DSP:0xc000001 Dest fec index: 0x0(0)
            FEC:  <<<< New information
            npu:0x0 fec:0x2000ffc5 port:0xc000001 encap:0x40010001

            TX-NHINFO:
                PI: 0x30901b88b0 PD: 0x30901b8950 rev:2364 dpa-rev:195015 Encap hdl: (nil)
               Encap id: 0x40010001 Remote: 0 L3 int: 45 flags: 0x7 transport_encap_id:0x0

                npu_mask: 0x1 DMAC: fc:82:a0:51:04:82
                ENCAP:  <<<< New information
                npu:0x0 mac:0x820451a082fc vlan:0x2d


    Load distribution: 0 (refcount 4)

    Hash  OK  Interface                Address
    0     Y   Bundle-Ether1301         fe80::11
```

- 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 0, internal 0x1020001 0x0 (ptr 0x8d7db7f0) [1], 0x0 (0x8daeedf0), 0x0
 (0x0)
 Updated Nov 20 13:33:23.556
 local adjacency 203.0.1.2
 Prefix Len 32, traffic index 0, Adjacency-prefix, precedence n/a, priority 15
  gateway array (0x8d84beb0) reference count 1, flags 0x0, source aib (10), 0 backups
                [2 type 3 flags 0x8401 (0x8d99a598) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8daeedf0, sh-ldi=0x8d99a598]
  gateway array update type-time 1 Nov 20 13:33:23.556
 LDI Update time Nov 20 13:33:23.556
 LW-LDI-TS Nov 20 13:33:23.556
   via 203.0.1.2/32, HundredGigE0/9/0/0, 3 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8cfc81a0 0x0]
    next hop 203.0.1.2/32
    local adjacency
        Load distribution: 0 (refcount 2)

    Hash  OK  Interface                 Address
    0     Y   HundredGigE0/9/0/0        203.0.1.2
```

- show adjacency detail

Displays detailed adjacency information, including Layer 2 information for each interface. The output of the show adjacency command varies by location.

```
Router#show adjacency detail

------------------------------------------------------------------------------
0/RP0/CPU0
------------------------------------------------------------------------------
Interface                  Address                 Version  Refcount Protocol
Hu0/9/0/0                 (interface)                 13        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4


Hu0/9/0/1                 (interface)                 31        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4
```

Per-Flow Load Balancing

The system inherently supports the 7-tuple hash algorithm. Load balancing describes the functionality in a router that distributes packets across multiple links based on Layer 3 (network layer) and Layer 4 (transport layer) routing information. If the router discovers multiple paths to a destination, the routing table is updated with multiple entries for that destination.

Per-flow load balancing performs these functions:

- Incoming data traffic is evenly distributed over multiple equal-cost connections.
- Incoming data traffic is evenly distributed over multiple equal-cost connections member links within a bundle interface.
- Layer 2 bundle and Layer 3 (network layer) load balancing decisions are taken on IPv4, IPv6, and MPLS flows . If it is an IPv4 or an IPv6 payload, then a 7-tuple hashing is done. If it is an MPLS payload with three or less labels, then the hardware parses the payload underneath and identifies whether the payload packet has an IPv4 or an IPv6 header. If it is an IPv4 or IPv6 header, then a 4-tuple hashing is performed based on the IP source, IP destination, router ID, and label stack; otherwise, an MPLS label based hashing is performed. In case of MPLS label-based hashing, the top 4 labels are used in hash computation..
- A 7-tuple hash algorithm provides more granular load balancing and used for load balancing over multiple equal-cost Layer 3 (network layer) paths. The Layer 3 (network layer) path is on a physical interface or on a bundle interface. In addition, load balancing over member links can occur within a Layer 2 bundle interface.
- The 7-tuple load-balance hash calculation contains:
  - Source IP address
  - Destination IP address
  - IP Protocol type
  - Router ID
  - Source port
  - Destination port

- Input interface

Load balancing decisions are taken based on a packet header, type of load balancing, type of scenario and platform specifics as follows:

- Packet header can contain one or many MAC, MPLS, IPv4 or IPv6 address, TCPor UDP headers, and so on.
- Load balancing can be done during ECMP or LAG (Bundle-Ether) forwarding.
- Scenarios can include IP forwarding, IP tunnel forwarding or decapsulation, MPLS forwarding or disaggregation, or Ethernet forwarding.
- The chipset type contains a packet's fields. These fields are considered for load balancing.

The following tables includes detailed list of options, list of scenarios, and headers fields to specify how ECMP or LAG load balancing is done.

Note:

- Only the fields that are highlighted in bold font are used for load balancing hash calculations. For example, for IP forwarding for IPv4 or IPv6 header and L4 (TCP or UDP) header, ECMP or LAG load balancing is done based on:
  - Source and destination IPv4 or IPv6 addresses
  - L4 source and destination ports
- To modify the hashing algorithm that is used for ECMP and bundle member selection, use the **hw-module profile load-balance algorithm** command in XR Config mode.

Table 4: ECMP or LAG Load Balancing for IP Forwarding

Header 4	Header 3	Header 2	Header 1
		IPv4	ETH
		IPv6	ETH
	L4	**IPv4**	ETH
	L4	**IPv6**	ETH
GTP	**L4**	**IPv4**	ETH
GTP	**L4**	**IPv6**	ETH

Table 5: ECMP or LAG Load Balancing for IP Tunnel Forwarding

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4	**IPv4**	ETH
	L4*	**IPv4**	**IPv4**	ETH

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv6	**IPv4**	ETH
		IPv6	**IPv4**	ETH
	IPv4	**MPLS[1..3]**	**IPv4**	ETH
L4	**IPv4**	**MPLS[1..3]**	**IPv4**	ETH
	IPv6	**MPLS[1..3]**	**IPv4**	ETH
L4	**IPv6**	**MPLS[1..3]**	**IPv4**	ETH
IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**IPv4**	ETH
IPv6	**MPLS[4..6]***	**MPLS[1..3]**	**IPv4**	ETH
MPLS[7..9]	**MPLS[4..6]***	**MPLS[1..3]**	**IPv4**	ETH
		IPv4	**IPv6**	ETH
	L4*	**IPv4**	**IPv6**	ETH
		IPv6	**IPv6**	ETH
	L4*	**IPv6**	**IPv6**	ETH



Note For MPLS packets, up to four labels are used for hash calculation.

Table 6: ECMP or LAG Load Balancing for IP Tunnel Decapsulation

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4	IPv4	ETH
	L4	**IPv4**	IPv4	ETH
		IPv6	IPv4	ETH
	L4	**IPv6**	IPv4	ETH
	IPv4	**MPLS[1..3]**	IPv4	ETH
L4	**IPv4**	**MPLS[1..3]**	IPv4	ETH
	IPv6	**MPLS[1..3]**	IPv4	ETH
L4	**IPv6**	**MPLS[1..3]**	IPv4	ETH
IPv4	**MPLS[4..6]**	**MPLS[1..3]**	IPv4	ETH
IPv6	**MPLS[4..6]**	**MPLS[1..3]**	IPv4	ETH

Header 5	Header 4	Header 3	Header 2	Header 1
MPLS[7..9]	**MPLS[4..6]**	**MPLS[1..3]**	IPv4	ETH

Note For MPLS packets, up to four labels are used for hash calculation.

Table 7: ECMP or LAG Load Balancing for MPLS Forwarding

Header 5	Header 4	Header 3	Header 2	Header 1
			MPLS[1..3]	ETH
	IPv4	**ETH**	**MPLS[1..3]**	ETH
	IPv6	**ETH**	**MPLS[1..3]**	ETH
		IPv4	**MPLS[1..3]**	ETH
		IPv6	**MPLS[1..3]**	ETH
	L4*	**IPv4**	**MPLS[1..3]**	ETH
	L4*	**IPv6**	**MPLS[1..3]**	ETH
	IPv4	**IPv4**	**MPLS[1..3]**	ETH
	IPv6	**IPv4**	**MPLS[1..3]**	ETH
L4	IPv4	**IPv4**	**MPLS[1..3]**	ETH
L4	IPv6	**IPv4**	**MPLS[1..3]**	ETH
		MPLS[4..6]	**MPLS[1..3]**	ETH
IPv4	ETH	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv6	ETH	**MPLS[4..6]**	**MPLS[1..3]**	ETH
	IPv4	**MPLS[4..6]**	**MPLS[1..3]**	ETH
	IPv6	**MPLS[4..6]**	**MPLS[1..3]**	ETH
L4	**IPv4**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
L4	**IPv6**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv4	**IPv4**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv6	**IPv4**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv4	**MPLS[7..9]**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv6	**MPLS[7..9]**	**MPLS[4..6]**	**MPLS[1..3]**	ETH

Note For MPLS packets with multiple labels, hash calculation is done based on the first five labels along other headers.

Table 8: ECMP or LAG Load Balancing for MPLS Deaggregation

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4	MPLS1	ETH
		IPv6	MPLS1	ETH
	L4*	**IPv4**	MPLS1	ETH
	L4*	**IPv6**	MPLS1	ETH
	IPv4	**IPv4**	MPLS1	ETH
	IPv6	**IPv4**	MPLS1	ETH
L4	IPv4	**IPv4**	MPLS1	ETH
L4	IPv6	**IPv4**	MPLS1	ETH
	IPv4	**ETH**	MPLS1	ETH
	IPv6	**ETH**	MPLS1	ETH
L4	**IPv4**	**ETH**	MPLS1	ETH
L4	**IPv6**	**ETH**	MPLS1	ETH

Table 9: ECMP or LAG Load Balancing for Ethernet Forwarding for IPoE Packets

Header 3	Header 2	Header 1
	IPv4	**ETH**
	IPv6	**ETH**
L4*	**IPv4**	**ETH**
L4*	**IPv6**	**ETH**

Table 10: ECMP or LAG Load Balancing Ethernet Forwarding for IPoE Packets with Complex Headers

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4*	**IPv4**	**ETH**
	L4	**IPv4***	**IPv4**	**ETH**
		IPv6*	**IPv4**	**ETH**

Header 5	Header 4	Header 3	Header 2	Header 1
	L4	**IPv6***	**IPv4**	**ETH**
	IPv4	**MPLS[1..3]***	**IPv4**	**ETH**
L4	IPv4	**MPLS[1..3]***	**IPv4**	**ETH**
	IPv6	**MPLS[1..3]***	**IPv4**	**ETH**
L4	IPv6	**MPLS[1..3]***	**IPv4**	**ETH**
IPv4	MPLS[4..6]	**MPLS[1..3]***	**IPv4**	**ETH**
IPv6	MPLS[4..6]	**MPLS[1..3]***	**IPv4**	**ETH**
MPLS[7..9]	MPLS[4..6]	**MPLS[1..3]***	**IPv4**	**ETH**

Note For MPLS packets with one through three labels, only the first label is used for load balancing along with other headers.

Table 11: ECMP or LAG Load Balancing Ethernet Forwarding for MPLS packets

Header 5	Header 4	Header 3	Header 2	Header 1
			MPLS[1..3]	**ETH**
		IPv4*	**MPLS[1..3]**	**ETH**
		IPv6*	**MPLS[1..3]**	**ETH**
	L4	**IPv4***	**MPLS[1..3]**	**ETH**
	L4	**IPv6***	**MPLS[1..3]**	**ETH**
	IPv4	**IPv4***	**MPLS[1..3]**	**ETH**
	IPv6	**IPv4***	**MPLS[1..3]**	**ETH**
L4	IPv4	**IPv4***	**MPLS[1..3]**	**ETH**
L4	IPv6	**IPv4***	**MPLS[1..3]**	**ETH**
		MPLS[4..6]*	**MPLS[1..3]**	**ETH**
	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
	IPv6	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
L4	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
L4	IPv6	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**

Header 5	Header 4	Header 3	Header 2	Header 1
IPv4	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
IPv6	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
IPv4	MPLS[7..9]	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
IPv6	MPLS[7..9]	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**

Note For MPLS packets with multiple labels, hash calculation is done based on first five labels along with other headers.

Per-Destination Load Balancing

Per destination load balancing is used for packets that transit over a recursive MPLS path (for example, learned through BGP 3107). Per-destination load balancing means the router distributes the packets based on the destination of the route. Given two paths to the same network, all packets for destination1 on that network go over the first path, all packets for destination2 on that network go over the second path, and so on. This preserves packet order, with potential unequal usage of the links. If one host receives the majority of the traffic all packets use one link, which leaves bandwidth on other links unused. A larger number of destination addresses leads to more equally used links.

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 203.0.1.2/32 and the next hop address is 1.0.0.2.




```
Router(config)#router static address-family ipv4 unicast
Router(config-static-afi)#203.0.1.2/32 HundredGigE 0/9/0/0 1.0.0.2
Router(config-static-afi)#commit
```

Running Configuration

```
Router#show running-config router static address-family ipv4 unicast
router static
 address-family ipv4 unicast
 203.0.1.2/32 HundredGigE 0/9/0/0 1.0.0.2
 !
!
```

Verification

Verify that the Next Hop Flags fields indicate COMPLETE for accurate functioning of the configuration.

The database, such as LPM, EXT-TCAM, and LEM, in which a prefix is updated is also provided through the output. Therefore, you can efficiently manage your network resources because you can understand the scaling of prefixes. You can also understand why a particular IP address configuration for a device fails and thereby debug easily.

```
Router#show cef 203.0.1.2/32 hardware egress details location 0/0/CPU0
Wed Nov  6 10:09:23.548 UTC
111.0.0.1/32, version 221, attached, internal 0x1000041 0x0 (ptr 0x8b00ea80) [1], 0x0
(0x8afd9768), 0x0 (0x0)
Updated Nov  6 10:08:07.424
Prefix Len 32, traffic index 0, precedence n/a, priority 2
  gateway array (0x8ae4baf0) reference count 1, flags 0x0, source rib (7), 0 backups
                [2 type 3 flags 0x40008441 (0x8af020c0) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8afd9768, sh-ldi=0x8af020c0]
  gateway array update type-time 1 Nov  6 10:08:07.423
LDI Update time Nov  6 10:08:07.423
LW-LDI-TS Nov  6 10:08:07.424
   via tunnel-ip1, 0 dependencies, recursive [flags 0x8]
    path-idx 0 NHID 0x0 [0x8ae0d728 0x0]
    local adjacency
```

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
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


VRF: default

_____________
Table ID: 0xe0000000. Total number of entries: 0
OOR state: GREEN. Number of OOR attributes: 0
```

Associated Commands

- show cef bgp-attribute

PPPoE

PPPoE provides a standard method of employing the authentication methods of the Point-to-Point Protocol (PPP) over an Ethernet network. When used by Internet Service Providers (ISPs), PPPoE allows authenticated assignment of IP addresses. In this type of implementation, the PPPoE client and server are interconnected by Layer 2 bridging protocols that run over a DSL or other broadband connection.

PPPoE has two phases:

- Active Discovery Phase - In this phase, the PPPoE client locates a PPPoE server, called an access concentrator. During this phase, the system assigns a session ID and then, a PPPoE layer is established.
- PPP Session Phase - In this phase, the system negotiates PPP options and performs authentication on IP addresses. Once the link setup is complete, PPPoE functions as a Layer 2 encapsulation method, which allows data transfer over the PPP link within PPPoE headers.

Following table list down the supported traffic types with enhanced hash for PPPoE over L2 bridging profile:

Traffic Type	Action
EXP-NULL-MPLS FWD	NONE

Traffic Type	Action
IPv4oPPPoEoETH	Load balance based on outer ETH src/dst MAC and inner ipv4 header
IPV6oPPPoEoETH	Load balance based on outer ETH src/dst MAC and inner ipv6 header
IPv4oMPLSoIPoETH FWD on MPLS	Load balance based on MPLS hdr and inner IP header (SIP/DIP)
IPv6oMPLSoIPoETH FWD on MPLS	Load balance based on MPLS hdr and inner IP header (SIP/DIP)
ETHoIPoMPLSx6oIPv4 FWD on MPLS	Packet with 4-6 Mpls labels, Load balanced based on inner IP header (SIP/DIP)
IPv4oMPLSoETH FWD on Inner-IP (decap case) IPv4oGREoIPvXoETH FWD on Inner-IP (decap case)	Load balance based on inner IP header (SIP/DIP)
IPv6oMPLSoETH FWD on Inner-IP (decap case) IPv6oGREoIPvXoETH FWD on Inner-IP (decap case)	Load balance based on inner IP header (SIP/DIP)
IPoPPPOEoMPLSoETH decap case of Pseudowire	Load balance based on Outer SMAC/DMAC and MPLS Labels.

Configure PPPoE

To configure the pppoe payload, use the following steps:

```
Router(config)#hw-module profile load-balance algorithm ppPoE
Thu Jul 22 09:22:52.962 UTC
reload of all chassis/all line cards is required only for PPPoE option configuration/removal
Router(config)#
Router(config)#commit
```



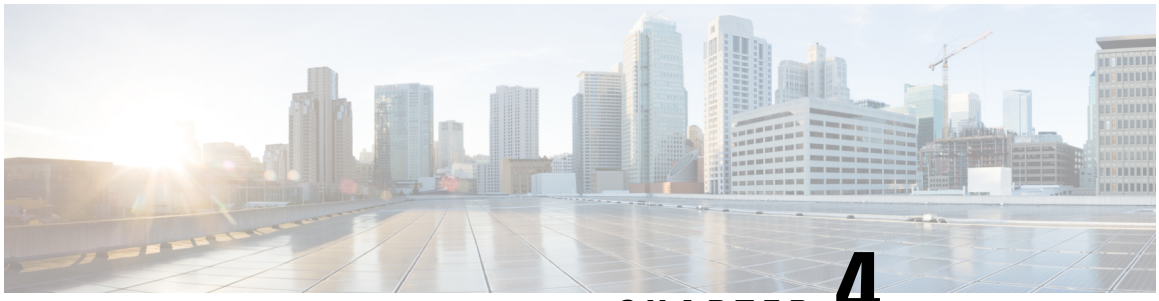
Note You must reload the chassis after you enable or disable pppoe payload feature.

Verify PPPoE

To verify the pppoe details, use the following command:

```
Router(config)#show running-config hw-module profile load-balance algorithm ppPoE
Thu Jul 22 09:23:14.694 UTC
hw-module profile load-balance algorithm PPPoE

Router(config)#
```



CHAPTER 4

Implementing Host Services and Applications

- Implementing Host Services and Applications, on page 59
- Network Connectivity Tools, on page 59
- Domain Services, on page 63
- TFTP Server, on page 64
- File Transfer Services, on page 65
- Cisco inetd, on page 68
- Telnet, on page 68
- Syslog source-interface, on page 69

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.

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 interface and destined for the Router B interface. If this ping succeeds, it is an indication that there is no routing problem. Router A knows how to get to the interface of Router B, and Router B knows how to get to the 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 interface of Router A and the interface of Router B is checked with the extended **ping** command.

With a normal ping from Router A to Router B's interface, the source address of the ping packet would be the address of the outgoing interface; that is the address of the interface, (10.0.0.2). When Router B replies to the ping packet, it replies to the source address (that is, 10.0.0.2). This way, only the connectivity between the interface of Router A (10.0.0.2) and the 10gige interface of Router B (10.0.0.1) is tested.

To test the connectivity between Router A's interface (10.0.0.2) and Router B's interface (10.0.0.1 .), 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 (10.0.0.2) and Router B (10.0.0.1) .

```
Router# ping 10.0.0.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.1, 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
Protocol [ipv4]:
Target IP address: 10.0.0.1
Repeat count [5]: 5
Datagram size [100]: 1000
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 1
Extended commands? [no]: no
```

```
Sweep range of sizes? [no]:
Type escape sequence to abort.
Sending 5, 1000-byte ICMP Echos to 10.0.0.1, timeout is 1 seconds:
!!!!!
Success rate is 100 percent (5/5)
Router#!!!!!
```

Associated Commands

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: 1.1.1.1
- 2: 2.2.2.2
- 3: 3.3.3.3

```
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: 1.1.1.1
2: 2.2.2.2
3: 3.3.3.3
4:
Starting pings...
Target IP address: 1.1.1.1
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 1.1.1.1, vrf is default, timeout is 1 seconds:
!!!!!
Success rate is 100 percent (5/5),
Target IP address: 2.2.2.2
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 1.1.1.1, vrf is default, timeout is 2 seconds:
```

```
!!!!!
Success rate is 100 percent (5/5),
Target IP address:  3.3.3.3
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 1.1.1.1, vrf is default, timeout is 1 seconds:
!!!!!
Success rate is 100 percent (4/5),
```

Associated Commands

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 10.0.0.2 to 20.1.1.1:

```
Router# traceroute 20.1.1.1
Type escape sequence to abort.
Tracing the route to 20.1.1.1
 1  10.0.0.1 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: 20.1.1.1
Source address: 10.0.0.2
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 20.1.1.1
 1  10.0.0.1 3 msec  *  3 msec
```

Associated Commands

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 (192.168.7.18 and 10.2.0.2 192.168.7.33) 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

TFTP Server

It is expensive and inefficient to have a machine that acts only as a server on every network segment. However, when you do not have a server on every segment, your network operations can incur substantial time delays across network segments. You can configure a router to serve as a TFTP server to reduce costs and time delays in your network while you use your router for its regular functions.

Typically, a router that you configure as a TFTP server enables the router to serve requests from client routers. This includes services such as providing client routers with system image or router configuration files from its flash memory. You can also configure the router to respond to other types of service requests.

Configuring a Router as a TFTP Server

The server and client router must be able to reach each other before the TFTP function can be implemented. Verify this connection by testing the connection between the server and client router (in either direction) using the **ping** command.

This task allows you to configure the router as a TFTP server so other devices acting as TFTP clients are able to read and write files from and to the router under a specific directory, such as slot0:, /tmp, and so on (TFTP home directory).



Note For security reasons, the TFTP server requires that a file must already exist for a write request to succeed.

The server and client router must be able to reach each other before the TFTP function can be implemented. Verify this connection by testing the connection between the server and client router (in either direction) using the **ping** command.

Configuration Example

Configure the router (home directory disk0:) as the TFTP server.

```
Router#configure
Router(config)#tftp ipv4 server homedir disk0
Router(config)#commit
```

Running Configuration

```
Router#show running-config tftp ipv4 server homedir disk0:
tftp vrf default ipv4 server homedir disk0:
```

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

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 0/9/0/0
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 0/9/0/0
ftp client source-interface HundredGigE 0/9/0/0
```

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 0/9/0/0 as the source address for TFTP connections:

```
Router#configure
Router(config)#tftp client source-interface HundredGigE 0/9/0/0
Router(config)#commit
```

Running Configuration

```
Router#show running-config tftp client source-interface HundredGigE 0/9/0/0
tftp client source-interface HundredGigE 0/9/0/0
```

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.

Telnet

Enabling Telnet allows inbound Telnet connections into a networking device.

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
```

Associated Commands

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

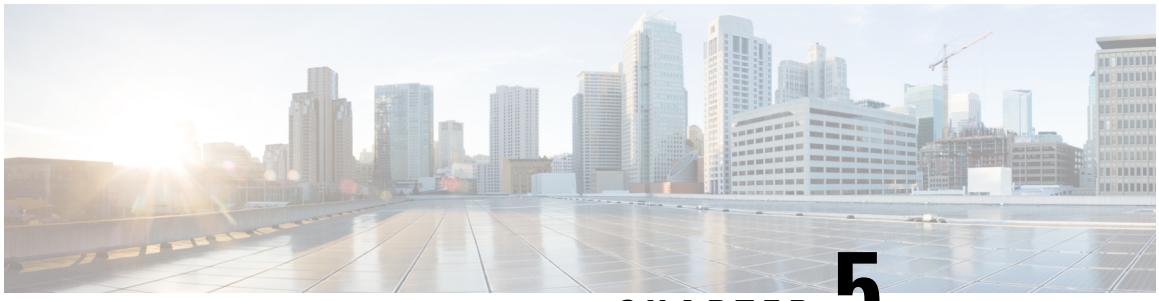
Router(config)#commit
```

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



CHAPTER 5

Introduction to DHCP

The Dynamic Host Configuration Protocol (DHCP) is a client-server network protocol that automatically assigns IP addresses and other configuration parameters to devices on a TCP/IP network. It enables hosts to obtain necessary network settings dynamically, reducing manual configuration and simplifying network management.

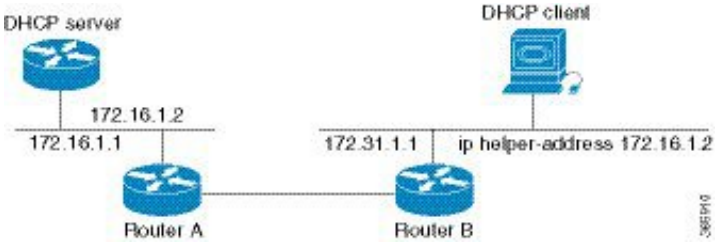
A DHCP relay agent is a host that forwards DHCP packets between clients and servers that do not reside on a shared physical subnet. Relay agent forwarding is distinct from the normal forwarding of an IP router where IP datagrams are switched between networks transparently.

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, a relay agent is needed on that network segment to ensure that DHCP packets reach the servers on another network segment. UDP broadcast packets are not forwarded, 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 by configuring a DHCP relay profile and configure one or more helper addresses in it. You can assign the profile to an interface or a VRF.

The figure below demonstrates the process. The DHCP client broadcasts a request for an IP address and additional configuration parameters on its local LAN. Acting as a DHCP relay agent, Router B picks up the broadcast, 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, on which the DHCP client's packets are received into the gateway address (giaddr) field of the DHCP packet, which enables the DHCP server to determine which subnet should receive 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, on page 72
- Limitations for DHCP Relay Feature , on page 72
- DHCP Option-82 with Circuit ID and Remote ID, on page 72

- DHCPv4 Relay Agent and Proxy Support for Segment Routing over IPv6 IPv4 L3VPN, on page 73
- How to Configure and Enable DHCP Relay Agent, on page 74
- Configure a DHCP Proxy Profile, on page 81
- Configuring and Enabling the DHCP Relay Option-82, on page 82
- DHCP Server, on page 83
- DHCP Client, on page 88
- DHCP Proxy Binding Table Reload Persistency, on page 89
- Jumbo Packet Handling for DHCPv6, on page 90

Prerequisites for Configuring DHCP Relay Agent

The following are the prerequisites to configure a DHCP relay agent:

- 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 configured and running DHCP client and DHCP server.
- Connectivity between the relay agent and DHCP server

Limitations for DHCP Relay Feature

These are the limitations for implementing DHCP relay feature:

- The multicast addresses are not supported. The **helper-address** command in DHCP relay profile submode supports valid unicast IP address as the helper address.

Note Configuring the **helper-address** command directly (not using profile) under a interface (such as BVI interface) is not supported.

- Only interface-id and remote-id DHCP option code are added by a relay agent while forwarding the packet to a DHCP server.

Note Configuring DHCP option code is not supported in DHCP relay profile submode.

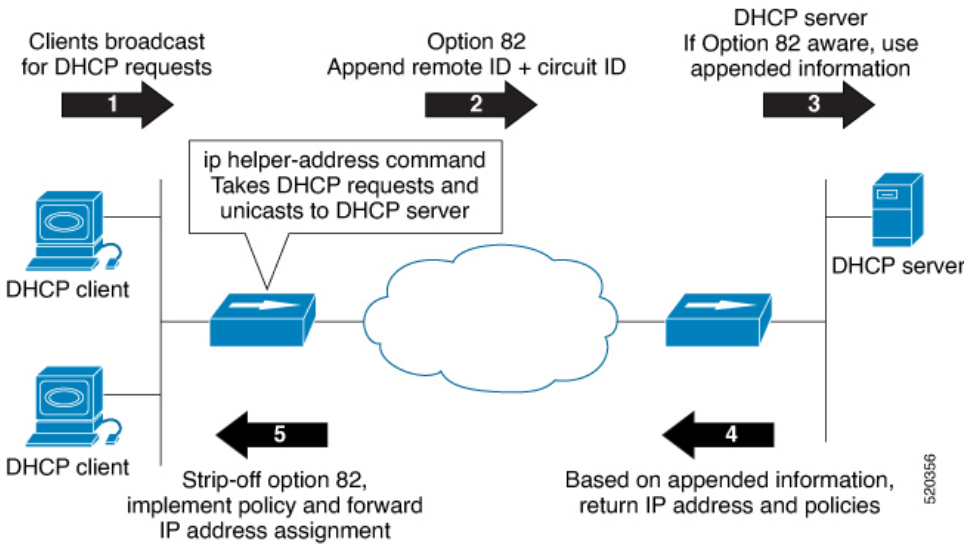
DHCP Option-82 with Circuit ID and Remote ID

The DHCP relay agent information option (option-82) enables a Dynamic Host Configuration Protocol (DHCP) relay agent to include information about itself when forwarding client-originated DHCP packets to a DHCP server. The DHCP server uses this information to implement IP address or other parameter-assignment policies.

This feature communicates information to the DHCP server using a sub-option of the DHCP relay agent information option called agent remote ID. The information sent in the agent remote ID includes an IP address identifying the relay agent and information about the ATM interface and the PVC over which the DHCP request came in. The DHCP server uses this information to make IP address assignments and security policy decisions.

The figure explains when the DHCP client generates a DHCP request and broadcasts it on the network. The DHCP relay agent intercepts the broadcast DHCP request packet, and inserts relay agent information option (option- 82) in the packet. The relay agent information option contains the related sub-options. The DHCP relay agent unicasts the DHCP packet to the DHCP server. The DHCP server receives the packet and uses the sub-options to assign IP addresses and other configuration parameters and forwards them back to the client. The sub-option fields are stripped off the packet by the relay agent while forwarding it to the client.

Figure 11: Forwarding Packets Using DHCP Relay Option-82



Effective with Cisco IOS XR Release 7.1.1, DHCP IPv4 relay-agent includes the remote ID and circuit ID information for DHCP IPv4 relayed packets.

The DHCP IPv4 packets relayed over a BVI interface displays the relay information option policy for a relay-agent. The DHCP IPv4 application includes the Option-82 remote ID and circuit ID information, and forwards it to the server.

DHCPv4 Relay Agent and Proxy Support for Segment Routing over IPv6 IPv4 L3VPN

DHCPv4 relay agent and proxy are supported on Segment Routing over IPv6 (SRv6) IPv4 L3VPN scenarios. See the How to Configure and Enable DHCP Relay Agent, on page 74 section for relay agent configuration. See the Configure a DHCP Proxy Profile, on page 81 section for proxy configuration.

For information about Segment Routing over IPv6, refer to the “Configure Segment Routing over IPv6 (SRv6)” chapter in the *Segment Routing Configuration Guide*.

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 10.10.10.1 giaddr 40.1.1.2
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 10.10.10.1 giaddr 40.1.1.2
  broadcast-flag policy check
  relay information option vpn
  relay information option vpn-mode rfc
  relay information option allow-untrusted
 !
```

Enabling DHCPv6 Relay Agent on an Interface

This task describes how to enable the Cisco IOS XR DHCPv6 relay agent on an interface.

Note On Cisco IOS XR software, the DHCPv6 relay agent is disabled by default.

```
RP/0/RP0/CPU0:router# configure terminal
RP/0/RP0/CPU0:router(config)# dhcp ipv6
RP/0/RP0/CPU0:router(config-dhcpv6)# interface type interface-instance relay profile
profile-name
RP/0/RP0/CPU0:router(config-dhcpv6-if)# commit
```

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 ipv6Router(config-dhcpv6)# no interface type name none
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 and verify the configured helper addresses.

```
Router#
show dhcp ipv6 relay statistics...
!
Profile: helper
Helper Addresses:
        2001:1:1::2, vrf default
Information Option: Disabled
Information Option Allow Untrusted: Disabled
Information Option VPN: Disabled
Information Option VPN Mode: RFC
Information Option Policy: Replace
Information Option Check: Disabled
GIADDR Policy: Keep
Broadcast-flag Policy: Ignore
VRF References:
Interface References:
```

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 by reviewing the contents of a DHCP RELAY-REPLY packet that is being relayed by the relay agent to the client. When the relay agent finds the prefix delegation option, the relay agent extracts the information about the prefix being delegated and inserts an IPv4 or IPv6 subscriber route matching the prefix delegation information onto the relay agent. Future packets destined to that prefix via relay are forwarded based on the information contained in the prefix delegation. The IPv4 or IPv6 subscriber route remains in the routing table until the prefix delegation lease time expires or the relay agent receives a release packet from the client releasing 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, and 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 can be 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 unicast reverse packet forwarding (uRPF) to work by allowing the router doing the reverse lookup to confirm that the Tpv4 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.

Configuring DHCP Stateful Relay Agent for Prefix Delegation

Perform this task to configure Dynamic Host Configuration Protocol DHCP relay agent notification for prefix delegation.

Configuration Example

1. Configure a DHCP profile.
2. Configure the DHCP relay agent.

3. Enable IPv4 or IPv6 DHCP on an interface that acts as an IPv4 or IPv6 DHCP stateful relay agent.
4. Configure the profile name.

Configuration

```
/* Enter the global configuration mode and then enter the DHCPv6 configuration mode. */
Router# config
Router(config)# dhcp ipv6
Router(config-dhcpv6)#

/* Enter the proxy profile configuration mode and configure the DHCPv6 relay agent. */
Router(config-dhcpv6)# profile downstream proxy
Router(config-dhcpv6-profile)# helper-address 2001:db8::1 GigabitEthernet 0/1/0/1

/* Exits from the proxy profile configuration mode and enable IPv6 DHCP on an interface.
*/
Router(config-dhcpv6-profile)# exit
Router(config-dhcpv6-if)# interface GigabitEthernet 0/1/0/0 proxy

/* Configure a profile name. */

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

DHCPv6 Relay Over BVI for IANA Address Allocation

DHCPv6 Relay agents relay all packets that are coming from DHCPv6 clients over the access-interfaces towards external DHCPv6 servers to 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 towards DHCPv6 clients over BVI interfaces. DHCPv6 Relay agents acts as stateless, by default, for DHCPv6 clients by not maintaining 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. Thereby, 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.
- Each DHCPv6 relay profile can be configured with upto 8 DHCPv6 server addresses.

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 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
```

DHCP Relay Profile: Example

The following example shows how to configure the DHCP relay profile:

```
dhcp ipv4
 profile client relay
  helper-address vrf foo 10.10.1.1
 !
! ...
```

DHCP Relay on an Interface: Example

The following example shows how to enable the DHCP relay agent on an interface:

```
dhcp ipv4
 interface GigabitEthernet 0/1/1/0 relay profile client
!
```

DHCP Relay on a VRF: Example

The following example shows how to enable the DHCP relay agent on a VRF:

```
dhcp ipv4
 vrf default relay profile client
```

```
!
```

Relay Agent Information Option Support: Example

The following example shows how to enable the relay agent and the insertion and removal of the DHCP relay information option:

```
dhcp ipv4
 profile client relay
relay information option

 !
!
```

Relay Agent Giaddr Policy: Example

The following example shows how to configure relay agent giaddr policy:

```
dhcp ipv4
 profile client relay
  giaddr policy drop
 !
!
```

Configure a DHCP Proxy Profile

The DHCP proxy performs all the functions of a relay and also provides some additional functions. The DHCP proxy conceals DHCP server details from DHCP clients. The DHCP proxy modifies the DHCP replies such that the client considers the proxy to be the server. In this state, the client interacts with the proxy as if it is the DHCP server.

Configuration Example

1. Enter DHCP IPv4 or DHCP IPv6 profile proxy submode.
2. Forward UDP broadcasts, including DHCP.



Note

- The value of the *address* argument can be a specific DHCP server address or a network address (if other DHCP servers are on the destination network segment). Using the network address enables other servers to respond to DHCP requests.
- For multiple servers, configure one helper address for each server.

Configuration

```
/* Enter the DHCP IPv4 profile proxy submode. */
Router(config)# dhcp ipv4
Router(config-dhcpv4)# profile client proxy

/* Forward UDP broadcastrs, including DHCP */
Router(config-dhcpv4-proxy-profile)# helper-address vrf vrf1 foo 10.10.1.1
Router(config-dhcpv4-proxy-profile)# commit
```

Configuring and Enabling the DHCP Relay Option-82

Configuration Example

The following examples is a DHCP IPv4 relay-agent configuration with the remote ID and circuit ID information. Configure the remote ID configuration within DHCP IPv4 relay profile configuration that is associated with the BVI interface. Also, configure the circuit ID information to the DHCP IPv4 packet configuration of the Layer 2 transport ACs.

```
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 test relay
/* Enables DHCP relay profile */

Router(config-dhcpv4-relay-profile)# helper-address vrf1 10.20.0.4 giaddr 10.20.0.1
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 check
Router(config-dhcpv4-relay-profile)# relay information option
Router(config-dhcpv4-relay-profile)# relay information policy drop
Router(config-dhcpv4-relay-profile)# relay information option vpn-mode rfc
Router(config-dhcpv4-relay-profile)# relay information option remote-id format-type hex 12
/*Configures the remote ID on DHCP relay */
/* 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 test1
Router(config-dhcpv4)# interface Bundle-Ether1.29 relay information option circuit-id
format-type ascii 120

/*Configures the circuit ID on the DHCP relay */
Router(config-dhcpv4)# commit
/* Configures DHCP relay on a BVI interface and commits the configuration */
```

Running Configuration

```
Router#show running-config
dhcp ipv4

 !
 profile test1 relay
  helper-address vrf vrf1 10.20.0.4 giaddr 10.20.0.1
  relay information option vpn
  relay information check
  relay information option
  relay information policy drop
  relay information option vpn-mode rfc
  relay information option remote-id format-type hex 12
  relay information option allow-untrusted


interface BVI2 relay profile test1
interface Bundle-Ether1.29 relay information option circuit-id format-type ascii 120

interface Bundle-Ether1.29 l2transport
 encapsulation dot1q 29
 rewrite ingress tag pop 1 symmetric
!

l2vpn
 bridge group IRB
  bridge-domain DHCP1
   interface Bundle-Ether1.29
    split-horizon group
   !
```

DHCP Server

A DHCP server accepts address assignment requests and renewals, and assigns the IP addresses from predefined groups of addresses contained within Distributed Address Pools (DAPS). DHCP servers can also be configured to supply additional information to the requesting client such as subnet mask, domain-name, the IP address of the DNS server, the default router, and other configuration parameters. DHCP servers can accept broadcasts from locally attached LAN segments or from DHCP requests that have been forwarded by other DHCP relay agents within the network.

The DHCP proxy performs all the functions of a relay and also provides some additional functions. The DHCP proxy conceals DHCP server details from DHCP clients. The DHCP proxy modifies the DHCP replies such that the client considers the proxy to be the server. In this state, the client interacts with the proxy as if it is the DHCP server.

When the DHCP server receives an address assignment request, it assigns the IP addresses from groups of IP addresses for DHCP in Distributed Address Pools (DAPS). The IP address used by the DHCP server to complete such requests is automatically excluded from the DAPS so that the DHCP server can safely assume that all the IP addresses available for its use in the DAPS are free.

DHCP Service-based Mode Selection

As part of DHCP service-based mode selection feature, a new mode called DHCP base is introduced. If an interface is configured in the DHCP base mode, then the DHCP selects either the DHCP proxy or the DHCP server mode to process the client request by matching option 60 (class-identifier) value of the client request with the configured value under the DHCP base profile.

The pool is configured under server-profile mode and server-profile-class submode. The class-based pool selection is always given priority over profile pool selection.

The DHCPv6 server-profile-class submode supports configuring DHCP options except few (0, 12, 50, 52, 53, 54, 58, 59, 61, 82, and 255).

```
dhcp ipv6
profile DHCP_BASE base
  match option 60 41424344 profile DHCPv6_PROXY proxy
  match option 60 41424355 profile DHCPv6_SERVER server
  default profile DEFAULT_PROFILE server
  relay information authenticate inserted
   !
profile DHCPv6_PROXY proxy
  helper-address vrf default 10.10.10.1 giaddr 0.0.0.0
!
profile DHCPv6_SERVER server
  lease 1 0 0
  pool IP_POOL
!
profile DEFAULT_PROFILE server
  lease 1 0 0
  pool IP_POOL
!
!
interface TenGigE 0/11/0/0 base profile DHCP_BASE
```

Configuring DHCP Server Profile

You can configure routers with DHCPv4 or DHCPv6 server profile.

Perform this task to configure the DHCPv6 server profile.

```
Router# configure
Router(config)# dhcp ipv6
Router(config-dhcpv6)# profile profile-name server
Router(config-dhcpv6-server-profile)# bootfile boot-file-name
Router(config-dhcpv6-server-profile)# broadcast-flag policy unicast-always
Router(config-dhcpv6-server-profile)# class class-name
Router(config-dhcpv6-server-profile-class)# exit
Router(config-dhcpv6-server-profile)# default-router address1 address2 ... address8
Router(config-dhcpv6-server-profile)# lease {infinite | days minutes seconds }
Router(config-dhcpv6-server-profile)# limit lease {per-circuit-id | per-interface|
per-remote-id} value
Router(config-dhcpv6-server-profile)# netbios-name server address1 address2 ... address8
Router(config-dhcpv6-server-profile)# netbios-node-type {number |b-node|h-node |m-node
|p-node}
Router(config-dhcpv6-server-profile)# option option-code {ascii string | hex string |ip
address}
Router(config-dhcpv6-server-profile)# pool pool-name
Router(config-dhcpv6-server-profile)# requested-ip-address-check disable
Router(config-dhcpv6-server-profile)# commit
```

Configuring Multiple Classes with a Pool

Perform this task to configure multiple classes with a pool.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# dhcp ipv6
RP/0/RP0/CPU0:router(config-dhcpv6)# profile profile-name server
RP/0/RP0/CPU0:router(config-dhcpv6-server-profile)# pool pool-name
```

```
RP/0/RP0/CPU0:router(config-dhcpv6-server-profile)# class class-name
RP/0/RP0/CPU0:router(config-dhcpv6-server-class)# pool pool-name
RP/0/RP0/CPU0:router(config-dhcpv6-server-class)# match option option [ sub-option sub-option]
 [ ascii asciiString | hex hexString ]
RP/0/RP0/CPU0:router(config-dhcpv6-server-class)# exit
RP/0/RP0/CPU0:router(config-dhcpv6-server-profile)# class class-name
RP/0/RP0/CPU0:router(config-dhcpv6-server-class)# pool pool-name
RP/0/RP0/CPU0:router(config-dhcpv6-server-class)# match vrf vrf-name
RP/0/RP0/CPU0:router(config-dhcpv6-server-class)# commit
```

Excluding a Range of Addresses from DAPS

This section provides an example of how to exclude a range of addresses from a configured DAPS .

Configuration Example

```
Router# configure
Router(config)# pool vrf EXAMPLE ipv4 EXAMPLE_POOL
/* Configures an IPv4 pool for the specifed VRF or all VRFs. Use the 'ipv6' keyword for
IPv6 pool. */
Router(config-pool-ipv4)# network 10.10.10.0/24 default-Router 10.10.10.1
/* Specifies network for allocation, along with the default Router. */
Router(config-pool-ipv4)# exclude 10.10.10.1 10.10.10.10
/* Specifies the range of addresses that are to be excluded */
Router(config-pool-ipv4)# commit
```

Verification Example

```
Router# show running-config pool
pool vrf EXAMPLE ipv4 EXAMPLE_POOL
 network 10.10.10.0/24 default-router 10.10.10.1
 exclude 10.10.10.1 10.10.10.10
```

Configuring a Server Profile DAPS with Class Match Option

This section discusses configuring a server profile DAPS with class match option.

Configuration Example

```
router#configure

router(config)#dhcp ipv4
/* The 'dhcp ipv6' command configures DHCP for IPv6 and enters the DHCPv6 configuration
submode. */

router(config-dhcpv4)#profile ISP1 server
/* Enters the server profile configuration mode. */

router(config-dhcpv4-server-profile)#pool ISP1_POOL
/* Configures the DAPS pool name. */

router(config-dhcpv4-server-profile)#class ISP1_CLASS
/* Creates and enters server profile class configuration submode. */

router(config-dhcpv4-server-profile-class)#pool ISP1_CLASS_POOL
/* Configures the pool name. */

router(config-dhcpv4-server-profile-class)#match option 60 hex PXEClient_1
/* DHCP server selects a pool from a class by matching options in the received DISCOVER
packet with the match option. */
```

```
router(config-dhcpv4-server-profile-class)#exit

router(config-dhcpv4-server-profile)#exit

router(config-dhcpv4)#profile ISP2 server
/* Enters the server profile configuration mode. */

router(config-dhcpv4-server-profile)#dns-server 10.20.3.4
/* Configures the name of the DNS server or the IP address. */

router(config-dhcpv4-server-profile)#pool ISP2_POOL
/* Configures the pool name. */

router(config-dhcpv4-server-profile)#class ISP2_CLASS
/* Creates and enters the server profile class. */

router(config-dhcpv4-server-profile-class)#pool ISP2_CLASS_POOL
/* Configures the pool name. */

router(config-dhcpv4-server-profile-class)#match option 60 hex PXEClient_2
/* DHCP server selects a pool from a class by matching options in the received DISCOVER
packet with the match option. */

router(config-dhcpv4-server-profile-class)#exit

router(config-dhcpv4-server-profile)#exit

router(config-dhcpv4)#commit
```

Running Configuration

```
Router#show running-config dhcp ipv4
dhcp ipv4
profile ISP1 server
pool ISP1_POOL
class ISP1_CLASS
pool ISP1_CLASS_POOL
match option 60 hex PXEClient_1
exit
exit
profile ISP2 server
dns-server 10.20.3.4
pool ISP2_POOL
class ISP2_CLASS
pool ISP2_CLASS_POOL
match option 60 hex PXEClient_2
exit
exit
!
```

Configuring Server Profile without DAPS Pool Match Option

This section discusses configuring a server profile without DAPS pool match option.

Configuration Example

```
router#configure

router(config)#dhcp ipv4
```

```
/* The 'dhcp ipv6' command configures DHCP for IPv6 and enters the DHCPv6 configuration
submode. */

router(config-dhcpv4)#profile ISP1 server
/* Enters the server profile configuration mode. */

router(config-dhcpv4-server-profile)#dns-server 10.10.10.10
/* Configures the name of the DNS server or IP address. */

router(config-dhcpv4-server-profile)#exit

router(config-dhcpv4)#profile ISP2 server
/* Enters the server profile configuration mode. */

router(config-dhcpv4-server-profile)#dns-server 11.11.11.11
/* Configures the name of the DNS server or IP address. */

router(config-dhcpv4-server-profile)#exit

router(config-dhcpv4)#commit
```

Running Configuration

```
Router#show running-config dhcp ipv4
dhcp ipv4
 profile ISP1 server

dns-server 10.10.10.10
  exit
 profile ISP2 server

dns-server 11.11.11.11
  exit
!
```

Configuring an Address Pool for Each ISP on DAPS

This section discusses configuring an address pool for each ISP on DAPS.

Configuration Example

```
router#configure

router(config)#pool vrf ISP_1 ipv4 ISP1_POOL
/* Configures an IPv4 pool for the specifed VRF or all VRFs. Use the 'ipv6' keyword for
IPv6 pool. */

router(config-pool-ipv4)#network 10.10.10.0
/* Specifies network for allocation. */

router(config-pool-ipv4)#exit

router(config)#pool vrf ISP_2 ipv4 ISP2_POOL
/* Configures an IPv4 pool for the specifed VRF or all VRFs. */

router(config-pool-ipv4)#network 10.20.20.0
/* Specifies network for allocation. */

router(config-pool-ipv4)#exit
```

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

Running Configuration

```
Router#show running-config pool
pool vrf ISP_1 ipv4 ISP1_POOL
 network 10.10.10.0 255.255.255.0
 exit
pool vrf ISP_2 ipv4 ISP2_POOL
 network 10.20.20.0 255.255.255.0
!
```

DHCP Client

The Dynamic Host Configuration Protocol (DHCP) client functionality enables the router interfaces to dynamically acquire the IPv4 or DHCPv4 or DHCPv6 server, and forwards the responses back to the correct Layer 2 address so that the correct device gets the correct configuration information.

DHCP has the ability to allocate IP addresses only for a configurable period of time, called the lease period. If the client is required to retain this IP address for a longer period beyond the lease period, the lease period must be renewed before the IP address expires. The client renews the lease based on configuration that was sent from the server. The client unicasts a REQUEST message using the IP address of the server. When a server receives the REQUEST message and responds with an ACK message. The lease period of the client is extended by the lease time configured in the ACK message.

Restrictions and Limitations

- DHCPv4 or DHCPv6 client can be enabled only on management interfaces.
- Either DHCPv4, DHCPv6, static IPv4, or static IPv6 can be configured on an interface.

Enabling DHCP Client on an Interface

The DHCPv4 or DHCPv6 client can be enabled at an interface level. The DHCP component receives a notification when DHCPv4 or DHCPv6 is enabled or disabled on an interface.

```
Router# configure
Router(config)# interface MgmtEth rack/slot/CPU0/port
Router(config)# interface interface_name ipv6 address dhcp
```

Associated Commands

- ipv6 address dhcp-client-options
- clear dhcp ipv6 client
- show dhcp ipv6 client
- show tech-support dhcp ipv6 client

DHCP Proxy Binding Table Reload Persistency

The Cisco IOS-XR Dynamic Host Configuration Protocol (DHCP) application is responsible for maintaining the DHCP binding state for the DHCP leases allocated to clients by the DHCP application. These binding states are learned by the DHCP application (proxy/relay/snooping). DHCP clients expect to maintain a DHCP lease regardless of the events that occur to the DHCP application.

Note From Release 6.2.2 onwards, 200K sessions are supported on a proxy or server running DHCPv4 or DHCPv6.

This feature enables the DHCP application to maintain bind state through the above events:

- Process restart – Local checkpoint
- RP failover – Hot standby RP through checkpoint
- LC IMDR – Local checkpoint
- LC OIR – Shadow table on RP
- System restart – Bindings saved on local disk

Configuring DHCP Relay Binding Database Write to System Persistent Memory

Perform this task to configure the DHCP relay binding database write to the system persistent memory. This helps to recover the DHCP relay binding table after a system reload. The file names used for a full persistent file write are dhcpv4_srpb_{nodeid}_odd or dhcpv6_srpb_{nodeid}_odd and dhcpv4_srpb_{nodeid}_even or dhcpv6_srpb_{nodeid}_even. The nodeid is the actual node ID of the node where the file is written. The incremental file is named the same way as the full file, with a _inc appended to it.

```
Router# configure
Router(config)# dhcp ipv6
Router(config-dhcpv6)# database [relay] [full-write-interval full-write-interval]
[incremental-write-interval incremental-write-interval]
Router(config-dhcpv6)# commit
```

Jumbo Packet Handling for DHCPv6

Table 12: Feature History Table

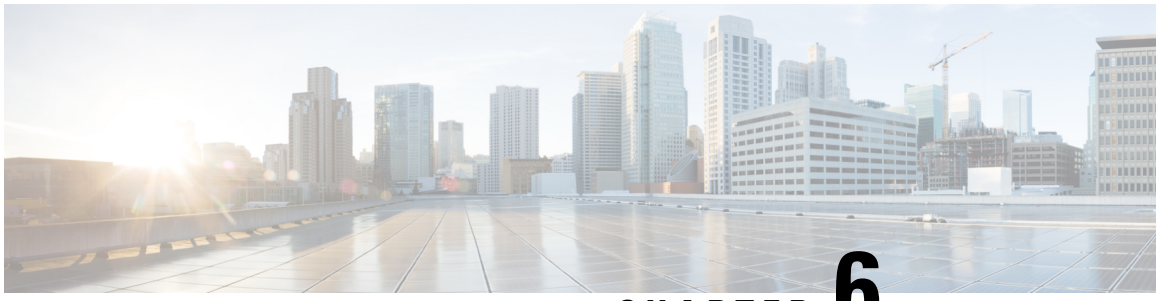
Feature Name	Release Information	Feature Description
Jumbo packet handling for DHCPv6	Release 7.4.1	This release introduces the **handle-jumbo-packet** configuration command under the dhcp ipv6 mode. This command enables processing of incoming DHCPv6 packets greater than 1280 bytes and upto 12,800 bytes in size. Prior to this release, the router discarded incoming DHCPv6 packets greater than 1280 bytes. The newly introduced command is: • handle-jumbo-packet

By default, the router allows incoming DHCPv6 packets with maximum size of 1280 bytes and drops any packet that is larger. If you configure the **handle-jumbo-packet** command under **dhcp ipv6** configuration mode, then the router allows incoming DHCPv6 packets upto 12,800 bytes in size. The router drops incoming packets larger than 12,800 bytes. You can configure this command for all modes of DHCPv6, that is, server, proxy and relay, as well as for both BNG (Broadband Network Gateway) and non-BNG networks.

Configuration Example

This example shows you how to configure **handle-jumbo-packet**:

```
Router# config
Router(config)# dhcp ipv6
Router(config-dhcpv6)# handle-jumbo-packet
Router(config-dhcpv6)# commit
```



CHAPTER 6

Implementing Access Lists and Prefix Lists

- Understanding Access Lists , on page 91
- User-Defined TCAM Keys for IPv4 and IPv6, on page 95
- Configuring IPv4 ACLs, on page 98
- Configuring IPv6 ACLs, on page 102
- Single Pass IPv6 Egress ACL, on page 107
- Configuring Chained ACLs, on page 108
- Modifying ACLs, on page 110
- Configuring ACL-based Forwarding, on page 111
- ACLs on Bridge Virtual Interfaces, on page 114
- Configuring ACLs with Fragment Control, on page 116
- Configuring ACL Filtering by IP Packet Length, on page 120
- Understanding Object-Group ACLs, on page 122
- Configuring TTL Matching and Rewriting for IPv4 ACLs, on page 124
- Configuring TTL Matching and Rewriting for IPv6 ACLs, on page 125
- Understanding IP Access List Logging Messages, on page 126
- Understanding Prefix Lists, on page 129

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. This profile can then be referenced by Cisco IOS XR software features such as traffic filtering, route filtering, QoS classification, and access control.

ACL compression refers to compressing the ACL in hardware in order to save TCAM space for large ACLs.

Traditional ACLs don’t support compression. Object-group ACLs use compression to accommodate the large number of ACEs. However, traditional ingress IPv4 and IPv6 ACLs are configured on external TCAM of NC57-18DD-SE line cards for both NCS 5500 and NCS 5700. Configuration of ACLs on external TCAM provides more space in the internal TCAM for other configurations.

Note

- Deny statistics are by default available.
- QoS stats do not work when the ACL-permit hw-module command is enabled.

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.

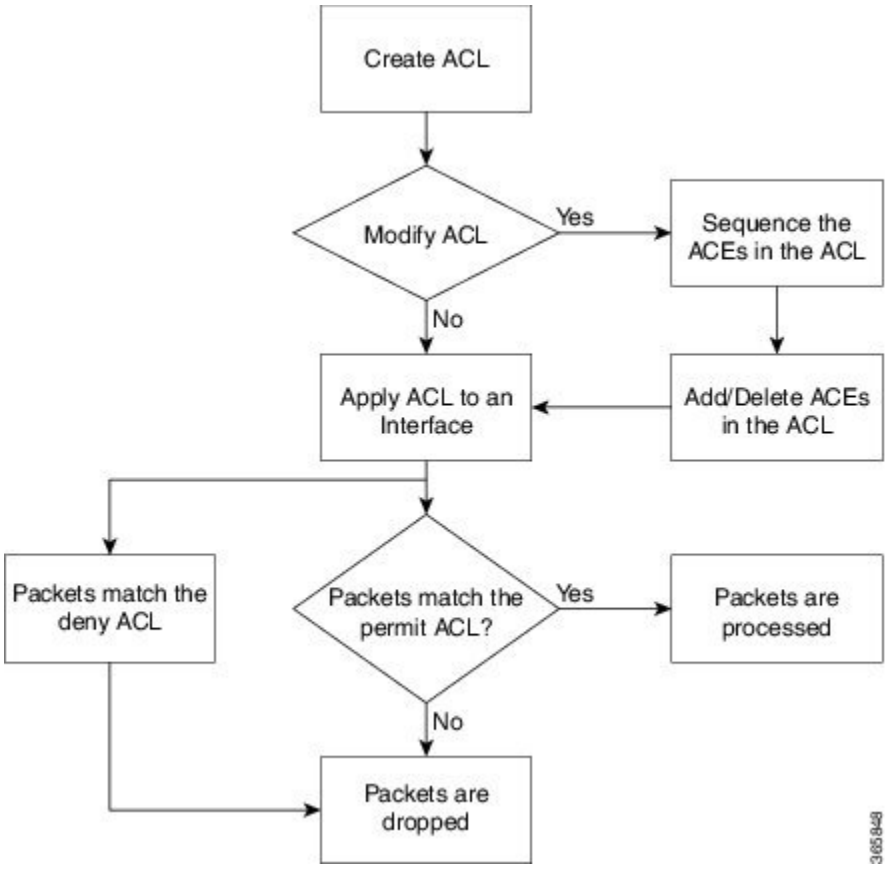
An access list can be configured and named, but it is not in effect until the access list is referenced by a command that accepts an access list. Multiple commands can reference the same access list. An access list can control traffic arriving at the router or leaving the router, but not traffic originating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

ACL Workflow

The following image illustrates the workflow of an ACL.



ACL Protocols

The following ACL protocol types are supported on NCS5500:

1. IPv4
2. IPv6
3. L2 (Ethernet-Services)

When an ACL of a particular protocol type is attached to an interface in a given direction, that interface must be configured for the specific protocol and the ACL only applies to traffic of that protocol type flowing on the interface.

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.
- The packet is matched with ACE within the ACL in the order of the sequence number.
- 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.
- 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 to 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 cannot 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 access group** command.
- ACL-based Forwarding (ABF) is not supported in common ACLs.
- Filtering of MPLS packets with the explicit-null or de-aggregation label is supported on the ingress direction.
- 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.
- 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.
- **Layer 2 ACL (with drop all ACE) to permit L2CP packets** - When the Layer 2 ACL is set to **drop all**, it drops any associated L2CP (Layer 2 Control Plane) packets. To process L2CP packets instead of dropping them, configure a permit L2CP ACE rule with a higher priority in the Layer 2 ACL, placing it before the matching Layer 2 ACE rule that drops all packets.

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 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.

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.

Limitations

- For Cisco NCS 5700 Series platforms and NC57 line cards in native mode, if you configure an IPv4 ACL with an ACE that permits UDP packets on the L2TP reserved port, the UDP packets on the port are implicitly denied despite meeting the permit conditions.

User-Defined TCAM Keys for IPv4 and IPv6

Access-lists on the 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. Not all available qualifier and action fields can be included in each key definition.

The key definitions are specific to a given ACL type, which can depend on the following attributes of the access-list:

- Direction of attachment, whether ingress or egress
- Protocol type (IPv4/IPv6/L2)
- Compression level (0:uncompressed, 3:compressed)

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 one of the three possible reasons (for ingress, traditional, IPv4/IPv6 ACL only):

- To include qualifier fields which are not included in the default TCAM key
- To change the ACL mode from *shared* to *unique* to support a greater number of unique ACLs, unique counters, etc.
- To reduce the size of the TCAM key (number of banks consumed)

A UDK can be configured using the following command:

```
hw-module profile tcam format access-list [ipv4 | ipv6] qualifiers [location rack/slot/cpu0]
```

This hw-module configuration will require reload of the chassis. If you want to use common ACL when a UDK is configured, you can add the `common-acl` option to the UDK.

User-Defined Fields

A TCAM key consists of several qualifiers, where the set of qualifiers are used to filter packets for a given ACL. The User-Defined Field (UDF) allows you to define a custom qualifier by specifying the location and size of the field, using the following UDF command:

```
udf udf-name header [ inner | outer ] [ l2 | l3 | l4 ] offset byte-offset length no of bytes
```

The UDF can then be added to a UDK as follows.

```
hw-module profile tcam format access-list [ipv4 | ipv6] qualifiers [udf1 udf-name udf2 udf-name] [location rack/slot/cpu0]
```

Note

- Up to 8 UDFs can be defined system wide. Currently, UDFs are globally defined.
- Cisco NCS 5500 and NCS 5700 Series Routers do not support ACL User Defined Field in slowpath. When packets punt to slowpath, the ACEs with UDF are applied without UDF. You must design the ACL accordingly to avoid packet drops in slowpath. For example, "`deny ipv4 any any udf udf-gtp 0x1 0xffffffff`" is specified as "`deny ipv4 any any`" in slowpath.

IPv4 and IPv6 Key Formats for Traditional Ingress ACL

User-defined TCAM key (UDK) definition is supported for ingress, traditional (uncompressed) IPv4 and IPv6 ACLs.

The following table shows the qualifier fields that are supported in the IPv4 and IPv6 key formats. If the default TCAM key is set as *Enabled*, then the Qualifier field is enabled by default. If the default TCAM key is set as *Disabled*, then the Qualifier field must use a UDK.

Table 13: Qualifier Fields Supported in IPv4 and IPv6 Key Formats

Parameter	Default TCAM Key	
	IPv4	IPv6
Source Address	Enabled	Enabled
Destination Address	Enabled	Enabled
Source Port	Enabled	Enabled
Destination Port	Enabled	Enabled
Port Range	Enabled	Not supported
Protocol/Next Header	Enabled	Enabled
Fragment bit	Enabled	Not supported
Packet length	Disabled	Disabled
Precedence/DSCP	Disabled	Enabled
TCP Flags	Enabled	Enabled
TTL Match	Disabled	Disabled
Interface based	Disabled	Not supported
UDF 1-7	Disabled	Disabled
ACL ID	Enabled	Enabled
common ACL bit	Enabled by default for IPv4/IPv6 on shared mode. Disabled by default for IPv4/IPv6 on unique mode.	Enabled by default for IPv4/IPv6 on shared mode. Disabled by default for IPv4/IPv6 on unique mode.
Interface-based (RIF)	Disabled	Disabled

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

Note You cannot configure QoS groups for ingress ACLs after a User-Defined TCAM Key (UDK) is configured because the command, **permit ipv4 any any set qos-group**, is not supported.

Table 14: Action Fields Supported in IPv4 and IPv6 Key Formats

Parameter	Default Action Field	
	IPv4	IPv6
Permit	Enabled	Enabled
Deny	Enabled	Enabled
Log	Enabled	Enabled
Capture	Enabled	Enabled
Stats Counter	Deny stats is always Enabled (permit stats has its own hw-module command)	Deny stats is always Enabled
TTL Set	Enabled	Enabled

To enable the monitoring of the packet count that is permitted based on the ACL rules, use the following configuration, and then reload the line card or router as required:

```
/* Enable permit statistics for the egress ACL (by default, only deny statistics are shown)
 */
Router(config)# hw-module profile stats acl-permit
Router(config)# commit
Router(config)# end
Router# reload location all
Wed Apr 5 23:05:46.193 UTC
Proceed with reload? [confirm]
```

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.

- Permit ACL matching stats are not supported.
- Ingress IPv4 ACLs are supported on all interfaces except management interfaces.
- ACL-based Forwarding (ABF) is supported only in the ingress direction.
- The total number of ingress ACLs allowed per NPU is 31 in shared ACL mode and more than 31 in unique ACL mode.

- The following line card limits apply for traditional ingress ACLs:
  - A maximum of 127 different traditional ingress ACLs per LC in shared ACL mode.
  - Unique ACL mode allows you to configure more than 128 different traditional ingress ACLs per LC.
  - The number of attached ACEs allowed per line card is 4096 or 2048, based on the NCS540 variant selected.
  - Starting with Cisco IOS XR Release 7.6.1, the router supports ACL logging with input interface (using the **log-input** keyword). Releases prior to Cisco IOS XR Release 7.6.1 does not support ACL logging with input interface on the router.
- Ingress ACL matching stats using **show access-lists ipv4** <*ACL name*> **hardware ingress location 0/RP0/CPU0** command may take more latency (up to 15 seconds) to reflect.
- The SE variant of the NCS 5700 line cards cannot match the IP-in-IP tunnel decapsulation packet outer header with Ingress ACL including the Hybrid ACL in external TCAM.

Notes and Restrictions for Configuring IPv4 Egress ACLs

IPv4 egress ACLs are characterized by the following behavior.

- Egress IPv4 ACLs are supported on main physical interfaces and bundle interfaces.

Note If you configure an egress ACL on a main interface that has sub-interfaces, the ACL action is also applied to the sub-interface traffic. This egress ACL behavior holds true even if the sub-interfaces are configured after the ACL is applied to the main interface.

- To configure egress ACLs on sub-interfaces, you must enable hw-module profile qos hqos-enable hardware profile.

  For more information on hw-module profile qos hqos-enable, refer to *Modular QoS Command Reference for Cisco NCS 5500 Series, Cisco NCS 540 Series, and Cisco NCS 560 Series Routers*
- The total number of egress ACLs allowed per NPU is 255.
- ACL is not supported on Management interface on egress direction.
- Apart from the throughput limitation, router-generated traffic is not be affected by egress IPv4 ACLs.
- The number of attached ACEs allowed per line card is 4096 or 2048, based on the NCS540 Variant selected.
- Filtering for egress IPv4 multicast traffic is not supported if H-QoS is configured on the router.
- IPv4 egress ACL does not filter packets based on TCP control flag and IPv4 fragment qualifiers due to hardware limitation when the packets are ingressed on BVI interface.

Configuring an Ingress IPv4 ACL on a Gigabit Ethernet Interface

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

```
/* Configure a GigE interface with an IPv4 address */
Router(config)# interface gigabitEthernet 0/0/0/0interface TenGigE 0/11/0/0
Router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
Router(config-if)# no shut
Router(config-if)# commit
Thu Jan 25 10:07:54.700 IST
Router(config-if)# exit

/* Verify if the interface is up */
Router(config)# do show ipv4 interface brief
Thu Jan 25 10:08:49.087 IST

Interface                      IP-Address      Status          Protocol Vrf-Name
GigabitEthernet0/0/0/0TenGigE0/11/0/0         10.1.1.1        Up              Up
default

/* Configure an IPv4 ingress ACL */
Router(config)# ipv4 access-list V4-ACL-INGRESS
Router(config-ipv4-acl)# 10 permit tcp 10.2.1.1 0.0.0.255 any
Router(config-ipv4-acl)# 20 deny udp any any
Router(config-ipv4-acl)# 30 permit ipv4 10.2.0.0 0.255.255.255 any
Router(config-ipv4-acl)# commit
Thu Jan 25 10:16:11.473 IST

/* Verify the ingress ACL creation */
Router(config)# do show access-lists ipv4
Thu Jan 25 10:25:19.896 IST
...
ipv4 access-list V4-ACL-INGRESS
 10 permit tcp 10.2.1.0 0.0.0.255 any
 20 deny udp any any
 30 permit ipv4 10.0.0.0 0.255.255.255 any


/* Apply the ingress ACL to the GigE interface */
Router(config)# interface GigabitEthernet0/0/0/0TenGigE0/11/0/0
Router(config-if)# ipv4 access-group V4-ACL-INGRESS ingress
Router(config-if)# commit
Thu Jan 25 10:28:19.671 IST
Router(config-if)# exit

/* Verify if the ingress ACL has been successfully applied to the interface */
Router(config)# do show ipv4 interface
Thu Jan 25 10:29:44.944 IST
GigabitEthernet0/0/0/0TenGigE0/11/0/0 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 10.1.1.1/24
  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 Gigabit Ethernet interface.

Configuring an Egress IPv4 ACL on a Gigabit Ethernet Interface

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

```
/* Configure a GigE interface with an IPv4 address */
Router(config)# interface gigabitEthernet 0/0/0/0interface TenGigE 0/11/0/0
Router(config-if)# ipv4 address 20.1.1.1 255.255.255.0
Router(config-if)# no shut
Router(config-if)# commit
Thu Jan 25 10:08:38.767 IST
Router(config-if)# exit

/* Verify if the interface is up */
Router(config)# do show ipv4 interface brief
Thu Jan 25 10:08:49.087 IST

Interface                      IP-Address      Status          Protocol Vrf-Name
GigabitEthernet0/0/0/0TenGigE0/11/0/0         10.1.1.1        Up              Up
default
GigabitEthernet0/0/0/0TenGigE0/11/0/1         20.1.1.1        Up              Up
default

/* Configure an IPv4 egress ACL */
Router(config)# ipv4 access-list V4-ACL-EGRESS
Router(config-ipv4-acl)# 10 permit ipv4 10.2.0.0 0.255.255.255 20.2.0.0 0.255.255.255
Router(config-ipv4-acl)# 20 deny ipv4 any any
Router(config-ipv4-acl)# commit
Thu Jan 25 10:25:04.655 IST

/* 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 10.0.0.0 0.255.255.255 20.0.0.0 0.255.255.255
 20 deny ipv4 any any
...

/* Apply the egress ACL to the GigE interface */
Router(config)# interface gigabitEthernet 0/0/0/1interface TenGigE 0/11/0/1
Router(config-if)# ipv4 access-group V4-ACL-EGRESS egress
Router(config-if)# commit
Thu Jan 25 10:28:45.937 IST
Router(config-if)# exit

/* Verify if the egress ACL has been successfully applied to the interface */
Router(config)# do show ipv4 interface
Thu Jan 25 10:29:44.944 IST
GigabitEthernet 0/0/0/1TenGigE 0/11/0/1 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 20.1.1.1/24
  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 Gigabit Ethernet interface.

Configuring IPv6 ACLs

This section describes the steps to configure ingress and egress IPv6 ACLs over gigabit ethernet and bundle interfaces.

Notes and Restrictions for Configuring IPv6 Ingress ACLs

IPv6 ingress ACLs are characterized by the following behavior.

- ACL-based Forwarding (ABF) is supported only in the ingress direction.
- The following line card limits apply for traditional ingress ACLs:
- The number of attached ACEs allowed per line card is 2048 in the ingress direction.
- Packet Length (using the **pkt-length** keyword) is not supported.
- Ingress ACL matching stats using **show access-lists ipv6** *<ACL name>* **hardware ingress location 0/RP0/CPU0** command may take more latency (up to 15 seconds) to reflect.
- The SE variant of the NCS 5700 line cards cannot match the IP-in-IP tunnel decapsulation packet outer header with Ingress ACL icluding the Hybrid ACL in external TCAM.

Notes and Restrictions for Configuring IPv6 Egress ACLs

IPv6 egress ACLs are characterized by the following behavior:

- Configuring packet length is not supported on egress ACLs.
- TCP flags are not supported on egress ACLs.
- Egress ACLs are not supported on BVI interfaces and L2 interfaces.
- Configuring **qos-group** is not supported on egress ACLs.
- A throughput of 50% or less is supported on egress ACLs.
- Apart from the throughput limitation, router-generated traffic is not be affected by egress IPv6 ACLs.
- The total number of egress ACLs allowed per NPU is 255.
- The total number of attached ACEs allowed per line card is 2048 in the egress direction.
- Configuring dynamic TCAM key is not supported on egress ACLs.
- Upto 160GB of total IPv6 egress ACL is supported per NPU because the Egress IPv6 ACLs take the recycle path.
- Filtering for egress IPv6 multicast traffic is not supported if H-QoS is configured on the router.
- 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.
- Applying both IPv6 ingress and egress ACLs simultaneously is not supported on these platform variants due to limited TCAM resources:
    - N540X-6Z18G-SYS-A

- N540-8Z12G-SYS-D
- N540X-8Z16G-SYS-A

However, in single pass mode, it is possible to have an IPv6 ingress ACL and an IPv6 egress ACL active at the same time on these platforms.

Configuring an Ingress IPv6 ACL on a Gigabit Ethernet Interface

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

```
/* Configure a GigE interface with an IPv6 address */
Router(config)# interface gigabitEthernet 0/0/0/0interface TenGigE 0/11/0/0
Router(config-if)# ipv6 address 1001::1/64
Router(config-if)# no shut
Router(config-if)# commit
Thu Jan 25 10:07:54.700 IST
Router(config-if)# exit

/* Verify if the interface is up */
Router(config)# do show ipv6 interface brief
Thu Jan 25 12:38:35.742 IST
GigabitEthernet 0/0/0/0TenGigE 0/11/0/0 [Up/Up]
    fe80::bd:b9ff:fea9:5606
    1001::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 Jan 25 11:31:24.488 IST
Router(config-ipv6-acl)# exit


/* Verify the ingress ACL creation */
Router(config)# do show access-lists ipv6
Thu Jan 25 11:34:56.911 IST
ipv6 access-list V6-INGRESS-ACL
 10 permit ipv6 any any
 20 deny udp any any


/* Apply the ingress ACL to the GigE interface */
Router(config)# interface gigabitEthernet 0/0/0/0interface TenGigE 0/11/0/0
Router(config-if)# ipv6 access-group V6-INGRESS-ACL ingress
Router(config-if)# commit
Thu Jan 25 11:32:55.194 IST
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
GigabitEthernet 0/0/0/0TenGigE 0/11/0/0 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):
    1001::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 Gigabit Ethernet interface.

Configuring an Egress IPv6 ACL on a Gigabit Ethernet Interface

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

```
/* Configure a GigE interface with an IPv6 address */
Router(config)# interface gigabitEthernet 0/0/0/1interface TenGigE 0/11/0/1
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 Jan 25 12:38:35.742 IST
GigabitEthernet 0/0/0/0TenGigE 0/11/0/0 [Up/Up]
    fe80::bd:b9ff:fea9:5606
    1001::1
GigabitEthernet 0/0/0/1TenGigE 0/11/0/1 [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 Jan 25 11:44:03.969 IST
Router(config-ipv6-acl)# exit

/* Verify the egress ACL creation */
Router(config)# do show access-lists ipv6
Thu Jan 25 11:45:53.823 IST
ipv6 access-list V6-EGRESS-ACL
 10 permit ipv6 any any
 20 deny udp any any
…

/* Apply the egress ACL to the GigE interface */
Router(config)# interface gigabitEthernet 0/0/0/1interface TenGigE 0/11/0/1
Router(config-if)# ipv6 access-group V6-EGRESS-ACL egress
Router(config-if)# commit
```

```
Thu Jan 25 11:45:12.682 IST
Router(config-if)# exit

/* Verify if the egress ACL has been successfully applied to the interface */
Router(config)# do show ipv6 interface
Thu Jan 25 11:46:43.234 IST
…
GigabitEthernet 0/0/0/1TenGigE 0/11/0/1 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 Gigabit Ethernet 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 3001::1/64
Router(config-if)# no shut
Router(config-if)# commit
Thu Jan 25 13:53:47.435 IST
Router(config-if)# exit

/* Configure an IPv6 egress ACL */
Router(config)# ipv6 access-list V6-EGRESS-ACL-bundle interface
Router(config-ipv6-acl)# 100 permit tcp any any eq www
Router(config-ipv6-acl)# 110 permit tcp any any eq https
Router(config-ipv6-acl)# 120 permit tcp any any eq ssh
Router(config-ipv6-acl)# 130 permit udp any any eq snmp
Router(config-ipv6-acl)# commit
Thu Jan 25 13:57:14.960 IST
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 Jan 25 13:59:23.198 IST
```

```
Router(config-ipv6-acl)# exit

/* Verify the egress and ingress ACL creation */
Router(config)# do show access-lists ipv6
Thu Jan 25 14:00:24.055 IST
ipv6 access-list V6-DENY-INGRESS-ACL
 10 deny ipv6 any any
ipv6 access-list V6-EGRESS-ACL-bundle
 100 permit tcp any any eq www
 110 permit tcp any any eq https
 120 permit tcp any any eq ssh
 130 permit udp any any eq snmp
...

/* 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-bundle egress
Router(config-if)# ipv6 access-group V6-DENY-INGRESS-ACL ingress
Router(config-if)# commit
Thu Jan 25 14:04:19.536 IST
Router(config-if)# exit

/* Verify if the ACLs have been successfully applied to the interface */
Router(config)# do show ipv6 interface
Thu Jan 25 11:46:43.234 IST
…
Thu Jan 25 14:04:51.322 IST
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):
    3001::1, subnet is 3001::/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.

Single Pass IPv6 Egress ACL

Table 15: Feature History Table

Feature Name	Release Information	Feature Description
Single Pass IPv6 Egress ACL	Release 7.10.1	You can now experience faster packet processing and save NPU cycles by avoiding the recycling of packets within the router. This is made possible by enabling the single-pass egress ACL which avoids multiple round-trips of packets in the ingress-to-egress path, thereby eliminating the need for additional packet processing. Also, because the match criteria requirement for a single-pass egress IPv6 ACL is reduced, the TCAM key size is reduced. This feature introduces the **hw-module profile acl ipv6 single-pass-egress-acl** command.

Due to egress PMF (Protected Management Frames) resource limitation, the packet can only be recycled over ingress PMF. Earlier, each IPv6 packet that was subjected to an egress ACL used to require two passes through the NPU (Network Processing Units). That means, the packet moved from the first pass ingress to the first pass egress pipeline. Then, the packet was recycled and reinjected to the second pass ingress and then to the second pass egress pipeline. Moving a packet four times resulted in high consumption of NPU bandwidth and power.

Starting Cisco IOS XR Software Release 7.10.1, you can enable single-pass IPv6 egress ACL. Now, with single-pass, the packet can flow into ingress pipeline and exit after egress pipeline, without getting reinjected to ingress pipeline again. With this feature, you can implement ACL on the egress pipeline and match packets based on fields like DSCP, precedence, and protocol.

Restrictions for Enabling Single-Pass IPv6 Egress ACL

These restrictions apply in configuring the single-pass IPv6 Egress ACL:

- You can only match fields like DSCP, precedence, and protocol on the single-pass egress IPv6 ACL.
- You can’t match the following fields on the single-pass egress IPv6 ACL:
  - Source port
  - Source IPv6 address
  - Destination port
  - Destination IPv6 address

Enable Single-Pass IPv6 Egress ACL

Configuration Example

Use the **hw-module profile acl ipv6 single-pass-egress-acl** command to configure the single-pass IPv6 egress ACL:

```
Router#configure terminal
Router(config)# hw-module profile acl ipv6 single-pass-egress-acl
Router(config)#commit
```

After configuring the **hw-module profile acl ipv6 single-pass-egress-acl** command, you must reload the router using the **reload** command.

```
Router#admin hw-module location all reload
```

The configuration takes effect, when the router reloads successfully after the configuration.

Running Configuration

Use the **show running configuration** command to see the running configuration.

```
Router#show running configuration!
Building configuration...
!! IOS XR Configuration 7.10.1
!! Last configuration change at Tue May 16 15:52:16 2023 by root
!
.
.
.
hw-module profile acl ipv6 single-pass-egress-acl
```

Verification

Use the **show controllers npu internaltcam location** command to verify that configuration is successful.

Without enabling the **hw-module profile acl ipv6 single-pass-egress-acl** command, you can see the following output:

```
Router#show controllers npu internaltcam location
0 0 160b egress_acl 2026 5 31 EGRESS_ACL_IPV6
0 6\7 320b pmf-0 2044 4 91 RCY_ACL_L3_IPV6
```

After enabling the **hw-module profile acl ipv6 single-pass-egress-acl** command, you can see the following output:

```
Router#show controllers npu internaltcam location
0 0 160b egress_acl 2026 5 31 EGRESS_ACL_IPV6
```

Configuring Chained ACLs

Chained ACLs also known as Multi-ACL enables you to apply more than one IPv4 or IPv6 (common and interface) ACL on an interface for packet filtering. This feature allows you to manage and configure different ACLs on an interface efficiently.

A typical ACL on the edge box for an ISP has two sets of ACEs:

- Common ISP specific ACEs (ISP protected address block)

- Customer/interface specific ACEs (Customer source address block)

The purpose of these address blocks is to deny access to ISP's protected infrastructure networks and anti-spoofing protection by allowing only customer source address blocks. This results in the configuration of unique ACLs per interface. Most of the ACEs are common across all the ACLs on a router. Therefore, ACL provisioning and modification is very cumbersome. Any changes to the ACE impacts every customer interface. This process also wastes the HW/TCAM resources as the common ACEs are being replicated in all ACLs.

This feature provides Ternary Content Addressable Memory(TCAM)/HW scalability. You can configure more than one ACL on a single interface. Therefore, you can separate various types of ACLs for management and other reasons, yet apply both of them on the same interface, in a defined order.

Restrictions

The following restrictions apply while implementing Common ACLs:

- Common ACLs are supported in only ingress direction and for L3 interfaces only.
- Only one common IPv4 and IPv6 ACL is supported on each line card.
- The common ACL option is not available for Ethernet Service (ES) ACLs.
- Packets are filtered through the common ACL configuration before the interface ACL configuration.
- You can edit common ACLs, but atomic replacement of the common ACLs is not supported.
- You cannot configure a common ACL on the same line card on which a compressed ACL is configured.
- The following line card limits apply for traditional and hybrid ingress ACLs:
  - A maximum of 127 different traditional ingress ACLs per LC in shared ACL mode.
  - Unique ACL mode allows you to configure more than 128 different traditional ingress ACLs per LC.

  A maximum of 512 different traditional ingress ACLs per LC.
- If you configure chained ACLs, the sequence of each ACL is reduced and the maximum sequence number that you can configure is 1 million.
- The **compress** option is not supported for common ACLs.
- Object-groups are not supported with common ACLs.

Configuration

You can use the following steps to configure chained ACLs:

1. Enter the interface configuration mode, and then configure an interface.
2. Configure a common acl and an interface acl, for example **common-1** and **interface-1** on the interface.

Configuration Example

```
/* Enter the interface configuration mode, and then configure an interface. */
Router# configure
Router(config)# interface TenGigE 0/0/0/0
```

```
/* Configure a common acl and an interface acl, for example common-1 and interface-1 on the
 interface. */
Router(config-if)# ipv4 access-group common common-1 interface-1 ingress
Router(config-if)# commit
```

Associated Commands

- ipv4 access-group
- ipv6 access-group

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 new entries, one with a sequence number "15" and another without a sequence number
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


*/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*/

*/Verify the entries of the ACL:*/
Router(config)#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*/
```

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.

Configuring ACL-based Forwarding

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 packet based on 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.

Feature Highlights

- ABF is only supported on ingress ACL.
- 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 will ensure that the packets take the right path towards the nexthop.

- VRF-aware ABF is supported for IPv4 and IPv6 with up to three next hops.
- IPv4 ABF nexthops routed over GRE interfaces are supported.
- IPv4 ABF nexthops routed over IPv6 are not supported.

- IPv6 ABF ACL does not work on bundle interface when applied in ingress direction.
- As ABF is ACL-based, packets that do not match an existing rule (ACE) in the ACL are subject to the default ACL rule (drop all). If the ACL is 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.

  As ABF is ACL-based, packets that do not match an existing rule (ACE) in the ACL are subject to the default ACL rule (drop all). If the ACL is 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.
- VRF-select (where only the VRF is configured for the nexthop) is not supported.
- VRF-select (where only the VRF is configured for the nexthop) is not supported.
- 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.
- Packets destined to the local IP interface ("for-us" packets) are subjected to redirect if they match the rule containing the ABF action. This can be avoided by either designing the rule to be specific enough to avoid matching the “for-us” packets or placing an explicit permit ACE rule (with higher priority) into the ACL before the matching ABF rule.

Configuration Example

To configure ACL-based forwarding, 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
```

```
/* (Optional) Display ACL information: */
Router# show access-lists ipv4 abf-acl
```

Running Configuration

```
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
commit
!


ipv4 access-list TEST
 10 permit ipv4 60.1.1.5 0.0.0.255 any nexthop1 vrf VRF1 nexthop2 vrf VRF2 nexthop3 vrf
VRF3
!
```

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 0/0/CPU0
Wed Jan 24 14:18:58.667 UTC

ACL name : abf-acl
ACE seq.   NH-1   NH-2   NH-3
 --------- --------------- --------------- ---------------
 10        192.168.13.2
 status          UP
 at status  Not Present
 exist      No
 vrf        default
 track
 pd     ctx  Present
 25   192.168.14.2    192.168.11.1  192.168.12.1
 status   UP   Down   Down
 at status  Not Present  Not Present  Not Present
 exist   No   Yes   Yes
 vrf   default  default  default
 track
 pd ctx   Present  Not present  Not present
 30   192.168.15.1  192.168.12.7
 status   Unknown  Unknown
 at status  Not Present  Not Present
 exist   No   Yes
 vrf   vrf1_ipv4  vrf1_ipv4
 track
 pd ctx   Not present  Not present
```

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

```
show access-lists usage pfilter location all
```

ACLs on Bridge Virtual Interfaces

Bridge Virtual Interfaces (BVIs) provide a bridge between the routing and bridging domains on a router. A BVI is configured with an IP address and 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.

Note Do not delete an ACL attached to a BVI interface when the BVI interface is not part of a bridge domain. An undefined ACL attached to a BVI implies a "deny all" action when you add the BVI to a bridge domain.

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.

- For ingress ACLs, the TCAM entries for the same ACL are shared across interfaces on the same NPU.
- For egress ACLs, the TCAM entries for the same ACL are unique for all interfaces. This leads to greater consumption of TCAM resources.

Restrictions for Configuring ACLs on BVIs

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

- Egress ACL over BVI are not supported on NC57 cards in compatibility mode.
- When an egress ACL is enabled on a BVI through the **hw-module** command, no other interface types are supported for the ACL (non-BVI interfaces are not supported for the ACL in this mode).

Prerequisites for Configuring Egress ACLs on BVIs

By default, an egress ACL on a BVI is disabled, and ACL filtering does not take place even when the ACL is attached to the BVI. Hence, we use the **hw-module** command, which enables the ACL when the line cards are reloaded.

Note IPv4 and IPv6 ingress ACLs do not require this configuration.

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 sample configuration.

1. Configure the HW module.

```
Router(config)# hw-module profile acl egress layer3 interface-based
Router(config)# commit
Router(config)# exit
```

2. 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
```

3. 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
```

4. Configure the Gigabit Ethernet interface that must be mapped to the BVI, and enable it for Layer 2 transport.

```
Router(config)# interface GigabitEthernet 0/0/0/0
Router(config-if)# l2transport
Router(config-if-l2)# commit
```

5. 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
```

6. Configure the bridge domain with the Gigabit Ethernet 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 GigabitEthernet 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
```

7. Confirm that your configuration has been successfully committed.

```
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 GigabitEthernet0/0/0/0
 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 GigabitEthernet0/0/0/0
   !
   routed interface BVI1
  !
 !
!
end
```

You have successfully configured and enabled IPv4 ingress and egress ACL on a BVI.

Verification

This section explains how to verify the IPv4 ACL configuration.

In Executive Privilege mode, confirm that the ACLs are in operation.

```
Router# show access-lists interface bvi1
Tue May 9 10:01:25.732 EDT
Input ACL (common): GigabitEthernet 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.

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 for IPv4 ACLs 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** The **fragments** keyword cannot be configured for an access-list entry that contains any Layer 4 information.

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** keyword cannot solve all cases involving access lists and IP fragments.

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.

Configuration

You can use the following configuration to configure the **fragments** keyword for an IPv4 access list:

```
/* Configure an Access List */
Router# configure
Router(config)# ipv4 access-list IPv4_Fragments

/* Configure the fragments keyword for the IPv4 access list */
Router(config-ipv4-acl)# 10 permit ipv4 any any fragments
Router(config-ipv4-acl)# commit
```

Associated Commands

- deny (IPv4)
- deny (IPv6)
- permit (IPv4)
- permit (IPv6)

Associated Topics

- Configuring an IPv4 ACL to Match on Fragment Type
- Matching by Fragment Offset in ACLs

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.

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 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.

Associated Commands

- fragment-type

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.
- 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.
- Packet length filtering is not supported in the default TCAM key, but instead requires a User-Defined TCAM Key (UDK) that can be specified using the `hw-module profile tcam format` command as described in the configuration section.

Associated Commands

- deny (IPv4)
- deny (IPv6)
- packet-length
- permit (IPv4)
- permit (IPv6)

Configuring Simple IPv4 ACLs to Filter by Packet Length

To configure a simple 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 dst-addr dst-port proto
 packet-length frag-bit port-range
```

2. Enter the global configuration mode and configure a simple IPv4 access list to filter packets by the packet length value.

   In this particular example, we configure a set of statements 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
Router(config)# ipv4 access-list pktlen-v4
Router(config-ipv4-acl)# 10 permit tcp any any packet-length eq 1664
Router(config-ipv4-acl)# 20 permit udp any any packet-length range 1600 2000
Router(config-ipv4-acl)# 30 deny ipv4 any any
```

3. Commit the ACL and exit the IPv4 ACL configuration mode.

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

4. Apply the ACL to the required Ethernet interface.

```
Router(config)# interface Te0/0/0/0
Router(config-if)# ipv4 access-group pktlen-v4 ingress
```

5. Commit the configuration and exit the interface configuration mode.

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

6. Verify your configuration.

```
Router# show access-lists pktlen-v4


ipv4 access-list pktlen-v4
10 permit ipv4 host 10.0.0.10 any packet-length lt 1008
20 permit ipv4 host 10.0.0.9 any packet-length gt 992
```

7. Verify the ACL matches in hardware.

```
Router# show access-lists pktlen-v4 hardware ingress location 0/RP0/CPU0

ipv4 access-list pklen-v4
10 permit ipv4 host 10.0.0.10 any packet-length lt 1008
20 permit ipv4 host 10.0.0.9 any packet-length gt 992
```

You have successfully configured a simple IPv4 ACL to filter by packet length.

Configuring Simple 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 dst-addr 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 Gigabit Ethernet interface.

```
Router# config
Router(config)# interface Te/0/0/0/3
Router(config-if)# ipv6 access-group scaled_acl2 ingress
```

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 0/0/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.

Understanding Object-Group ACLs

You can use object-group ACLs to classify users, devices, or protocols into groups so 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.

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.

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/Layer 4 protocols.

Compressing ACLs

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

- Source IP prefix
- Destination IP prefix
- Source port number

From IOS XR Release 7.7.1 onwards, you can enable egress, which allows you to separate address prefixes and ports into network object groups and port object groups respectively.

There are only two 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. Internal TCAM resources are utilized and there will be a huge impact on system resources and time taken for processing the ACL.

- **Compress level 3**: All three fields (source IP, destination IP, and source port) in an ACE are compressed.

  In this mode, external TCAM is used for prefix lookup, and internal TCAM is used for ACE lookup. This mode supports 16-bit based packet length filtering and fragment offset filtering.

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.
- Object-group is not supported in common ACLs.
- Nested object-groups are not supported from Release 6.2.1.
- Any inline ACE update to an object group ACL clears complete stats of the ACL.

Configuring TTL Matching and Rewriting for IPv4 ACLs

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 range of TTL values. You can also rewrite the TTL value in the IPv4 header by using the **set ttl** command.

Limitations for using TTL matching and rewriting for IPv4 ACLs

Using TTL matching and rewriting for IPv4 ACLs is known to have the following limitations.

- TTL matching is supported only for ingress ACLs.
- ACL logging is not supported for ingress ACLs after a User-Defined TCAM Key (UDK) is configured with the **enable-set-ttl** option.
- If a TTL rewrite is applied to the outer IPv4 header of an IP-in-IP header, then when the outer IPv4 header is decapsulated, (by GRE decapsulation) the TTL rewrite is also applied to the inner IPv4 header.
- TTL matching is not supported in the default TCAM key, but instead requires a User-Defined TCAM Key (UDK) using the **hw-module profile tcam format** command as described in the configuration section.

Configuration

Use the following steps to configure TTL matching and rewriting for IPv4 ACLs.

```
/* Enable TTL matching and rewriting in the global configuration mode by using the hw-module
 command */
Router(config)# hw-module profile tcam format access-list ipv4 dst-addr dst-port proto
port-range enable-set-ttl ttl-match

/* 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 set ttl 200
Router(config-ipv4-acl)# 30 permit tcp any any ttl neq 100 set ttl 255
Router(config-ipv4-acl)# commit
Thu Nov  2 12:22:58.948 IST

/* Attach the IPv4 ACL to the GigE interface */
Router(config)# interface Te0/0/0/0
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
!
hw-module profile tcam format access-list ipv4 dst-addr dst-port proto port-range
enable-set-ttl ttl-match
!
ipv4 access-list acl-v4
 10 deny tcp any any ttl eq 100
 20 permit tcp any any ttl range 1 50 set ttl 200
 30 permit tcp any any ttl neq 100 set ttl 255
!
interface Te0/0/0/0
 ipv4 address 15.1.1.1 255.255.255.0
 ipv4 access-group acl-v4 ingress
!
```

You have successfully configured TTL matching and rewriting for IPv4 ACLs.

Configuring TTL Matching and Rewriting for IPv6 ACLs

You can configure ACLs to match on the TTL value specified in the IPv6 header. You can specify the TTL match condition to be based on a single value, or a range of TTLs or TTL not equal to any value. You can also rewrite the TTL value in the IPv6 header by using the **set ttl** command.

Note A reboot of the line cards is required after entering the **hw-module profile** command to activate the command.

Limitations for using TTL matching and rewriting for IPv6 ACLs

Using TTL matching and rewriting for IPv6 ACLs is known to have the following limitations.

- TTL matching is supported only for ingress ACLs.
- ACL logging is not supported for ingress ACLs after a User-Defined TCAM Key (UDK) is configured with the **enable-set-ttl** option.
- If a TTL rewrite is applied to the outer IPv6 header of an IP-in-IP header, then when the outer IPv6 header is decapsulated, (by GRE decapsulation) the TTL rewrite is also applied to the inner IPv6 header.
- TTL matching is not supported in the default TCAM key, but instead requires a User-Defined TCAM Key (UDK) using the **hw-module profile tcam format** command as described in the Configuration section.

Configuration

Use the following steps to configure TTL matching and rewriting for IPv6 ACLs.

```
/* Enable TTL matching and rewriting in the global configuration mode by using the hw-module
 command */
```

```
Router(config)# hw-module profile tcam format access-list ipv6 dst-addr dst-port src-port
next-hdr enable-set-ttl ttl-match

/* Configure an IPv6 ACL with the TTL parameters */
Router(config)# ipv6 access-list acl-v6
Router(config-ipv6-acl)# 10 deny tcp any any ttl eq 50
Router(config-ipv6-acl)# 20 permit tcp any any ttl lt 50 set ttl 255
Router(config-ipv6-acl)# 30 permit tcp any any ttl gt 50 set ttl 200
Router(config-ipv6-acl)# commit
Thu Nov  2 12:22:58.948 IST

/* Attach the IPv6 ACL to the GigE interface */
Router(config)# interface Te0/0/0/0
Router(config-if)# ipv6 address 2001:2:1::1/64
Router(config-if)# ipv6 access-group acl-v6 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
!hw-module profile tcam format access-list ipv6 dst-addr dst-port src-port next-hdr
enable-set-ttl ttl-match
!
ipv6 access-list acl-v6
 10 deny tcp any any ttl eq 50
20 permit tcp any any ttl lt 50 set ttl 255
30 permit tcp any any ttl gt 50 set ttl 200
!
interface Te0/0/0/0
 ipv6 address 2001:2:1::1/64
 ipv6 access-group acl-v6 ingress
!
```

You have successfully configured TTL matching and rewriting for IPv6 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.

Note ACL logging isn't supported for ingress MPLS packets.

Note ACL logging isn't supported for ingress MPLS packets.

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.

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 isn't 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's when a threshold isn't 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 shouldn't be used as a billing tool or as an accurate source of the number of matches to an access list.

Enable Logging on ACE

This section shows you how to enable the ACE of an ACL to log informational messages when it matches incoming packets, using the optional keyword **log**. The router supports this feature only for IPv4 or IPv6 ingress ACLs. 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.

```
Router#configure
Router(config)#ipv4 access-list test
Router(config-ipv4-acl)#10 permit udp 10.85.1.0 255.255.255.0 log
Router(config-ipv4-acl)#exit
Router(config)# interface FortyGigE0/0/0/22
Router(config-if)# ipv4 access-group test ingress
Router(config-if)# commit
```



Note Set log-level to **informational** or higher with the **logging console** command, so that the router displays the ACL log-messages on the console.

```
Router#configure
Router(config)#logging console informational
Router(config)# commit
```

For more information on log-levels, see section *Syslog Message Severity Levels* in the *Implementing System Logging* chapter of the *System Monitoring Configuration Guide*.

The following snippet shows a sample log message:

```
Router: ipv4_acl_mgr[350]: %ACL-IPV4_ACL-6-IPACCESSLOGP : access-list test (10) permit udp
10.85.1.2(0) -> 10.0.0.1(0), 1 packet
```

Enable Ingress Interface Logging on ACE

Table 16: Feature History Table

Feature Name	Release Information	Feature Description
Enable Ingress Interface Logging on ACE	Release 7.6.1	Using the **log-input** keyword, you can now enable Access Control Entries (ACEs) 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 ACE in earlier releases. This feature introduces an optional keyword **log-input** for the following commands: • permit (IPv4) • permit (IPv6) • deny (IPv4) • deny (IPv6)

This section shows you how to configure the ACE of an ACL with the optional keyword **log-input**. This option provides the same functionality as the **log** keyword, as described in the previous section *Enable Logging on ACE*, 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 user has enabled 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
```

Understanding Prefix Lists

Prefix lists are used in route maps and route filtering operations and can be used as an alternative to access lists in many Border Gateway Protocol (BGP) route filtering commands. A prefix is a portion of an IP address, starting from the far left bit of the far left octet. By specifying exactly how many bits of an address belong to a prefix, you can then use prefixes to aggregate addresses and perform some function on them, such as redistribution (filter routing updates).

BGP Filtering Using Prefix Lists

Prefix lists can be used as an alternative to access lists in many BGP route filtering commands. It is configured under the Global configurations of the BGP protocol. The advantages of using prefix lists are as follows:

- Significant performance improvement in loading and route lookup of large lists.
- Incremental updates are supported.
- More user friendly CLI. The CLI for using access lists to filter BGP updates is difficult to understand and use because it uses the packet filtering format.
- Greater flexibility.

Before using a prefix list in a command, you must set up a prefix list, and you may want to assign sequence numbers to the entries in the prefix list.

How the System Filters Traffic by Prefix List

Filtering by prefix list involves matching the prefixes of routes with those listed in the prefix list. When there is a match, the route is used. More specifically, whether a prefix is permitted or denied is based upon the following rules:

- An empty prefix list permits all prefixes.
- An implicit deny is assumed if a given prefix does not match any entries of a prefix list.
- When multiple entries of a prefix list match a given prefix, the longest, most specific match is chosen.

Sequence numbers are generated automatically unless you disable this automatic generation. If you disable the automatic generation of sequence numbers, you must specify the sequence number for each entry using the *sequence-number* argument of the **permit** and **deny** commands in IPv4 prefix list configuration command. Use the **no** form of the **permit** or **deny** command with the *sequence-number* argument to remove a prefix-list entry.

The **show** commands include the sequence numbers in their output.

Configuring Prefix Lists

Configuration Example

Creates a prefix-list "pfx_2" with a remark "Deny all routes with a prefix of 10/8". This prefix-list denies all prefixes matching /24 in 128.0.0.0/8.

```
Router#configure
Router(config)#ipv4 prefix-list pfx_2
```

```
Router(config-ipv4_pfx)#10 remark Deny all routes with a prefix of 10/8
Router(config-ipv4_pfx)#20 deny 128.0.0.0/8 eq 24
/* Repeat the above step as necessary. Use the no sequence-number command to delete an
entry. */

Router(config-ipv4_pfx)#commit
```

Running Configuration

```
Router#show running-config ipv4 prefix-list pfx_2
ipv4 prefix-list pfx_2
 10 remark Deny all routes with a prefix of 10/8
 20 deny 128.0.0.0/8 eq 24
!
```

Verification

Verify that the permit and remark settings are according to the set configuration.

```
Router# show prefix-list pfx_2
ipv4 prefix-list pfx_2
 10 remark Deny all routes with a prefix of 10/8
 20 deny 128.0.0.0/8 eq 24
RP/0/RP0/CPU0:ios#
```

Associated Commands

Sequencing Prefix List Entries and Revising the Prefix List

Configuration Example

Assigns sequence numbers to entries in a named prefix list and how to add or delete an entry to or from a prefix list. It is assumed a user wants to revise a prefix list. Resequencing a prefix list is optional.

Note It is possible to resequence ACLs for prefix-list but not for security ACLs.

```
Router#config
Router(config)#ipv4 prefix-list cl_1


Router(config)#10 permit 172.16.0.0 0.0.255.255
/* Repeat the above step as necessary adding statements by sequence number where you planned;
 use the no sequence-number command to delete an entry */

Router(config)#commit
end
Router#resequence prefix-list ipv4 cl_1 20 15
```

Running Configuration

```
/*Before resequencing/*
Router#show running-config ipv4 prefix-list cl_1
ipv4 prefix-list cl_1
```

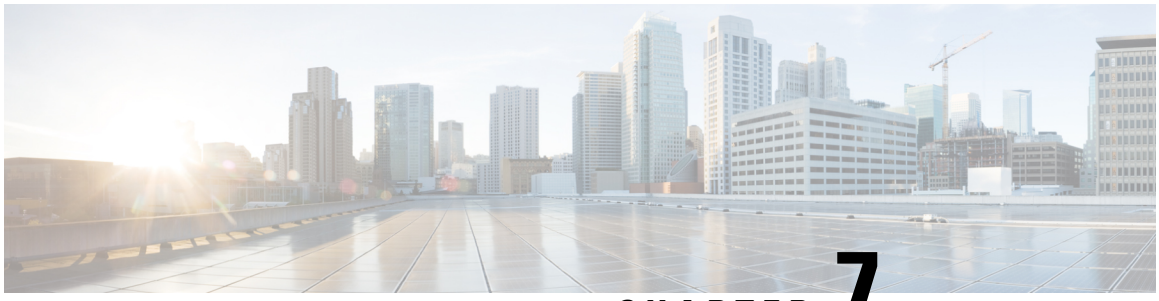
```
 10 permit 172.16.0.0/16
!
/* After resequencing using the resequence prefix-list ipv4 cl_1 20 15 command: */
Router#show running-config ipv4 prefix-list cl_1
ipv4 prefix-list cl_1
 20 permit 172.16.0.0/16
!
```

Verification

Verify that the prefix list has been resequenced:

```
Router#show prefix-list cl_1
ipv4 prefix-list cl_1
 20 permit 172.16.0.0/16
```

Associated Commands



CHAPTER 7

Implementing 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.)

Feature History for Implementing HSRP

- Prerequisites for Implementing HSRP , on page 133
- Guidelines and Restrictions for Implementing HSRP , on page 133
- Information About Implementing HSRP, on page 133
- How to Implement HSRP, on page 137
- BFD for HSRP, on page 152
- Enhanced Object Tracking for HSRP and IP Static, on page 154
- Hot Restartability for HSRP, on page 155
- Configuration Examples for HSRP Implementation on Software, on page 155

Prerequisites for Implementing HSRP

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.

Guidelines and Restrictions for Implementing HSRP

HSRP is supported on Ethernet interfaces, Ethernet sub-interfaces, Ethernet link bundles, and Bridge Virtual Interfaces (BVIs).

Information About Implementing HSRP

To implement HSRP on Cisco IOS XR software 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.

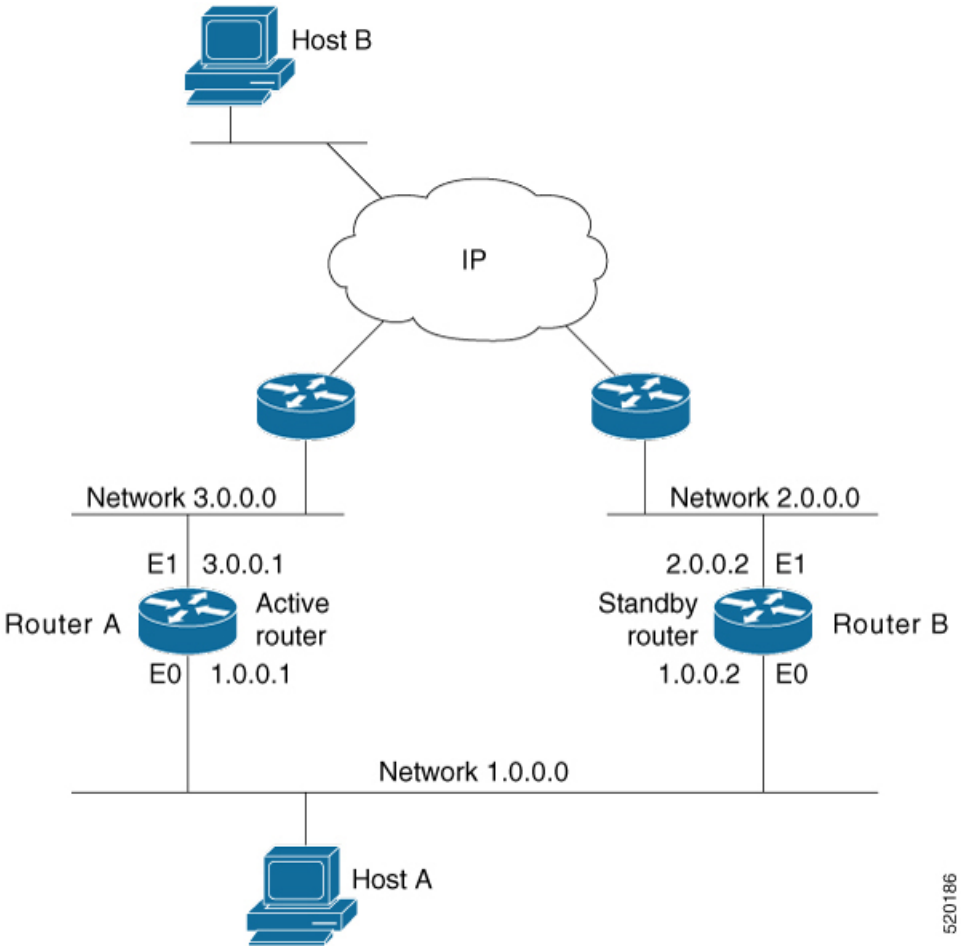
HSRP Groups

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.

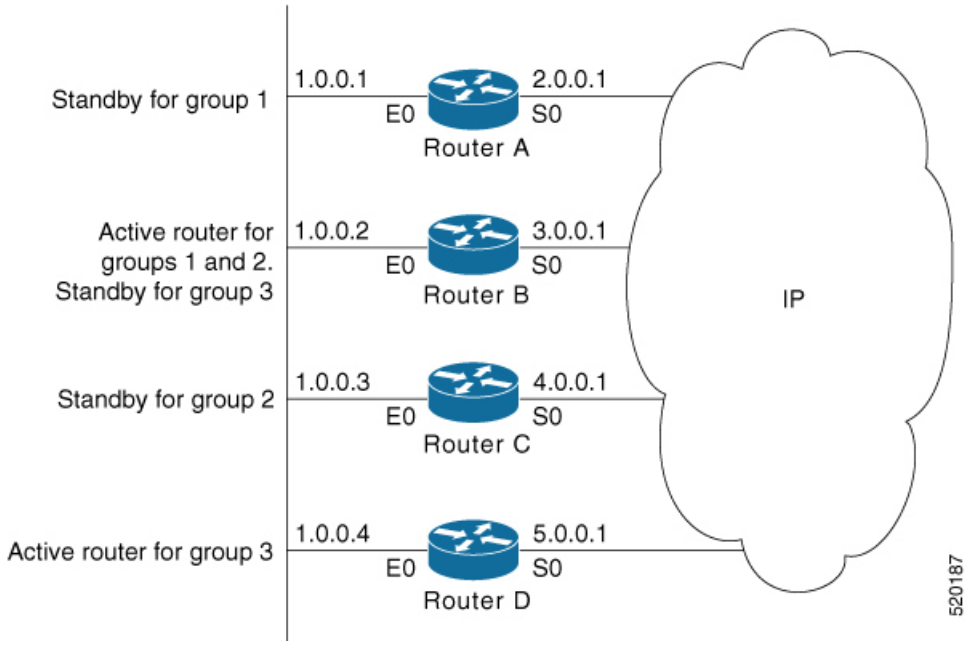
Figure 12: Routers Configured as an HSRP Group



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 13: 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.

How to Implement HSRP

This section contains instructions for the following tasks:

Enabling HSRP

The **hsrp ipv4** command activates HSRP on the configured interface. If an IP address is specified, that address is used as the designated address for the Hot Standby group. If no IP address is specified, the virtual address is learned from the active router. For HSRP to elect a designated router, at least one router in the Hot Standby group must have been configured with, or learned, the designated address. Configuring the designated address on the active router always overrides a designated address that is currently in use.

Configuration Steps

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP group submode.

Note The **version** keyword is available only if IPv4 address-family is selected. By default, version is set to 2 for IPv6 address families.

4. Activate HSRP on the configured interface.

Note
- If an IP address is specified, that address is used as the designated address for the Hot Standby group. If no IP address is specified, the virtual address is learned from the active router.
- If you configure HSRP for IPv6, you must configure a link local IPv6 address or enable it using the **autoconfig** keyword. If you do not configure a linklocal IPv6 address, the router does not accept the configuration when you commit your changes using the **commit** keyword.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP group submode. */
Router(config-hsrp-ipv4)# hsrp <group-number> version <version-no>

/* Activate HSRP on the configured interface. */
Router(config-hsrp-gp)# address {learn|address[secondary]}
Router(config-hsrp-gp)# commit
```

Running Configuration

```
Router# show running-configuration
 router hsrp
 interface GigabitEthernet0/2/0/1
  address-family ipv4
   hsrp 1 version 1
    address learn
   !
```

Running Configuration for Extended Group Range

```
Router# show running-configuration
router hsrp
interface TenGigE 0/0/0/2
 address-family ipv4
  hsrp <1-4095> version <1-2>
  address 10.20.30.1
  address 10.20.30.2 secondary
!
```

Enabling HSRP for IPv6

Use the following steps to enable HSRP for IPv6.

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP group submode.

Note The **version** keyword is available only if IPv4 address-family is selected. By default, version is set to 2 for IPv6 address families.

4. Activate HSRP on the configured interface and assigns a linklocal IPv6 address..

Note
- The virtual linklocal address must not match any other virtual linklocal address that is already configured for a different group.
- The virtual linklocal address must not match the interface linklocal IPv6 address.
- If you use the **autoconfig** keyword, the linklocal address is calculated using the EUI-64 format.
- Use the **legacy-compatible** keyword to be compatible with Cisco IOS and other legacy Cisco devices.

5. Activate HSRP on the configured interface and assigns a global IPv6 address.

Note If you configure HSRP for IPv6, you must configure a link local IPv6 address or enable it using the **autoconfig** keyword. If you do not configure a linklocal IPv6 address, the router does not accept the configuration when you commit your changes using the **commit** keyword.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP group submode. */
Router(config-hsrp-ipv4)# hsrp <group-number>

/* Activate HSRP on the configured interface. */
Router(config-hsrp-gp)# address global <ipv6-address>
Router(config-hsrp-gp)# commit
```

Running Configuration

```
Router# show running-configuration
configure
 router hsrp
 interface GigabitEthernet0/2/0/1
  address-family ipv4
   hsrp 1
    address linklocal autoconfig
    address global 2001:DB8:A:B::1
   !
```

Running Configuration for Expanded Group Range

```
Router# show running-configuration
router hsrp
interface TenGigE 0/0/0/2
 address-family ipv6
  hsrp <1-4095>
  address global 1:1::1
  address linklocal autoconfig
!
```

Configuring HSRP Group Attributes

To configure other Hot Standby group attributes that affect how the local router participates in HSRP, use the following procedure in interface configuration mode as needed:

Configuration Example

1. Enable HSRP configuration mode.
2. Enable HSRP interface configuration mode on a specific interface.
3. Enable HSRP address-family configuration mode on a specific interface.
4. Enable HSRP group submode.



Note The **version** keyword is available only if IPv4 address-family is selected. By default, version is set to 2 for IPv6 address families.

5. (Optional) Configure HSRP priority.
   - The assigned priority is used to help select the active and standby routers. Assuming that preemption is enabled, the router with the highest priority becomes the designated active router. In case of ties, the primary IP addresses are compared, and the higher IP address has priority.
   - The priority of the device can change dynamically if an interface is configured with the **track** command and another interface on the device goes down.
   - If preemption is not enabled using the **preempt** command, the router may not become active even though it might have a higher priority than other HSRP routers.
   - To restore the default HSRP priority values, use the **no priority** command.
6. (Optional) Configure an interface so that the Hot Standby priority changes on the basis of the availability of other interfaces.

- When a tracked interface goes down, the Hot Standby priority decreases by 10. If an interface is not tracked, its state changes do not affect the Hot Standby priority. For each interface configured for Hot Standby, you can configure a separate list of interfaces to be tracked.
- The optional *priority-decrement* argument specifies by how much to decrement the Hot Standby priority when a tracked interface goes down. When the tracked interface comes back up, the priority is incrementally increased by the same amount.
- When multiple tracked interfaces are down and the *priority-decrement* argument has been configured, these configured priority decrements are cumulative. If tracked interfaces are down, but none of them were configured with priority decrements, the default decrement is 10 and it is cumulative.
- The **preempt** command must be used in conjunction with this command on all routers in the group whenever the best available router should be used to forward packets. If the **preempt** command is not used, the active router stays active, regardless of the current priorities of the other HSRP routers.
- To remove the tracking, use the **no preempt** command.

7. (Optional) Configure HSRP preemption and preemption delay.

- When you configure preemption and preemption delay with the **preempt** command, the local router attempts to assume control as the active router when the local router has a Hot Standby priority higher than the current active router. If the **preempt** command is not configured, the local router assumes control as the active router only if it receives information indicating that no router is currently in the active state (acting as the designated router).
- When a router first comes up, it does not have a complete routing table. If it is configured to preempt, it becomes the active router, yet it is unable to provide adequate routing services. This problem can be solved by configuring a delay before the preempting router actually preempts the currently active router.
- The preempt *delay seconds* value does not apply if there is no router currently in the active state. In this case, the local router becomes active after the appropriate timeouts (see the **timers** command), regardless of the preempt delay seconds value.
- To restore the default HSRP preemption and preemption delay values, use the **no preempt** command.

8. (Optional) Configure an authentication string for the Hot Standby Router Protocol (HSRP).

- The authentication string is sent unencrypted in all HSRP messages. The same authentication string must be configured on all routers and access servers on a LAN to ensure interoperation.
- Authentication mismatch prevents a device from learning the designated Hot Standby IP address and the Hot Standby timer values from other routers configured with HSRP.
- Authentication mismatch does not prevent protocol events such as one router taking over as the designated router.
- To delete an authentication string, use the **no authentication** command.

Configuration

```
/* Enable HSRP configuration mode. */
Router# configure
Router(config)# router hsrp

/* Enable HSRP interface configuration mode on a specific interface. */
```

```
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP group submode. */
Router(config-hsrp-ipv4)# hsrp <group-number> version <version-no>

/* (Optional) Configure HSRP priority. */
Router(config-hsrp-gp)# priority <priority>

/* (Optional) Configure an interface so that the Hot Standby priority changes on the basis
 of the availability of other interfaces. */
Router(config-hsrp-gp)# track <type> instance <priority-decrement>

/* (Optional) Configure an authentication string for the Hot Standby Router Protocol (HSRP).
 */
Router(config-hsrp-gp)# authentication <string>
```

Running Configuration

```
Router# show running-configuration
configure
router hsrp
 interface TenGigE0/2/0/1
  address-family ipv4
   hsrp 1 version 1
    priority 100
    track TenGigE0/3/0/1
    preempt
    authentication company1
   !
  !
 !
!
```

Configuring the HSRP Activation Delay

The activation delay for HSRP is designed to delay the startup of the state machine when an interface comes up. This give the network time to settle and avoids unnecessary state changes early after the link comes up.

Configuration Example

1. Enable HSRP configuration mode.
2. Enable HSRP interface configuration mode on a specific interface.
3. Configure the delay of startup of the state machine.

Note The reload delay is the delay applied after the first interface up event. The minimum delay is the delay that is applied after any subsequent interface up event (if the interface flaps).

4. Enable HSRP address-family configuration mode on a specific interface.
5. Enable HSRP group submode.
6. Activate HSRP on the configured interface.

Note

- If an IP address is specified, that address is used as the designated address for the Hot Standby group. If no IP address is specified, the virtual address is learned from the active router.
- If you configure HSRP for IPv6, you must configure a link local IPv6 address or enable it using the **autoconfig** keyword. If you do not configure a linklocal IPv6 address, the router does not accept the configuration when you commit your changes using the **commit** keyword.

Configuration

```
/* Enable HSRP configuration mode. */
Router# configure
Router(config)# router hsrp

/* Enable HSRP interface configuration mode on a specific interface. */
Router(config-hsrp)# interface <type> <interface-path-id>

/* Configure the delay of startup of the state machine. */
Router(config-hsrp-if)# hsrp delay minimum <seconds> reload <seconds>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP group submode. */
Router(config-hsrp-ipv4)# hsrp <group-number> version <version-no>

/* Activate HSRP on the configured interface. */
Router(config-hsrp-gp)# address { learn | address [secondary] }
Router(config-hsrp-gp)# commit
```

Running Configuration

```
Router# show running-configuration
configure
router hsrp
 interface TenGigE0/2/0/1
  hsrp delay minimum 2 reload 10
  address-family ipv4
   hsrp 1
    address learn
   !
```

Disabling HSRP Support for ICMP Redirect Messages

By default, HSRP filtering of ICMP redirect messages is enabled on routers running HSRP.

To configure the reenabling of this feature on your router if it is disabled, use the **hsrp redirects** command in interface configuration mode.

Configuration Example

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP group submode.

Note The **version** keyword is available only if IPv4 address-family is selected. By default, version is set to 2 for IPv6 address families.

4. Activate HSRP on the configured interface.

Note
- If an IP address is specified, that address is used as the designated address for the Hot Standby group. If no IP address is specified, the virtual address is learned from the active router.
- If you configure HSRP for IPv6, you must configure a link local IPv6 address or enable it using the **autoconfig** keyword. If you do not configure a linklocal IPv6 address, the router does not accept the configuration when you commit your changes using the **commit** keyword.

5. Configure Internet Control Message Protocol (ICMP) redirect messages to be sent when the Hot Standby Router Protocol (HSRP) is configured on an interface.

Note
- The **hsrp redirects** command can be configured on a per-interface basis. When HSRP is first configured on an interface, the setting for that interface inherits the global value. If ICMP redirects have been explicitly disabled on an interface, then the global command cannot reenable the functionality.
- With the **hsrp redirects** command enabled, ICMP redirect messages are filtered by replacing the real IP address in the next-hop address of the redirect packet with a virtual IP address, if it is known to HSRP.
- To revert to the default, which is that ICMP messages are enabled, use the **no hsrp redirects** command.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Configure Internet Control Message Protocol (ICMP) redirect messages to be sent when the
 Hot Standby Router Protocol (HSRP) is configured on an interface. */
Router(config-hsrp-gp)# hsrp redirects disable

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP group submode. */
Router(config-hsrp-ipv4)# hsrp <group-number> version <version-no>

/* Activate HSRP on the configured interface. */
Router(config-hsrp-gp)# address {learn|address[secondary]}
```

Running Configuration

```
Router# show running-configuration
router hsrp
```

```
interface TenGigE 0/2/0/1
 address-family ipv4
  hsrp 1 version 1
   address learn
  !
 !
 hsrp redirects disable
!
!
```

Multiple Group Optimization (MGO) for HSRP

Multiple Group Optimization provides a solution for reducing control traffic in a deployment consisting of many subinterfaces. By running the HSRP control traffic for just one of the sessions, the control traffic is reduced for the subinterfaces with identical redundancy requirements. All other sessions are subordinates of this primary session, and inherit their states from it.

Customizing HSRP

Customizing the behavior of HSRP is optional. Be aware that as soon as you enable a HSRP group, that group is in operation.

Configuration Example

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP group submode.



Note The **version** keyword is available only if IPv4 address-family is selected. By default, version is set to 2 for IPv6 address families.

4. Configure an HSRP session name.
5. Enables hot standby protocol for IP.



Note If an IP address is specified, that address is used as the designated address for the Hot Standby group. If no IP address is specified, the virtual address is learned from the active router.

6. Configure the secondary virtual IPv4 address for a router.
7. Configure an authentication string for the Hot Standby Router Protocol (HSRP).
8. Enable HSRP slave configuration mode on a specific interface.
9. Configure the subordinate group to inherit its state from a specified group.
10. Configure the primary virtual IPv4 address for the subordinate group.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP group configuration mode on a specific interface. */
Router(config-hsrp-ipv4)# hsrp <group-no> version <version-no>

/* Configure an HSRP session name. */
Router(config-hsrp-gp)# name <name>

/* Enable hot standby protocol for IP. */
Router(config-hsrp-gp)# address { learn | address}

/* Configure the secondary virtual IPv4 address for a router. */
Router(config-hsrp-gp)# address <address> secondary

/* Configures an authentication string for the Hot Standby Router Protocol (HSRP). */
Router(config-hsrp-gp)# authentication <string>

/* Enables HSRP slave configuration mode on a specific interface. */
Router(config-hsrp-gp)# hsrp <group-no> slave

/* Configure the subordinate group to inherit its state from a specified group. */
Router(config-hsrp-slave)# follow mgo-session-name

/* Configure the primary virtual IPv4 address for the subordinate group.
Router(config-hsrp-slave)# address <ip-address>
```

Running Configuration

```
Router# show running-configuration
router hsrp
 interface TenGigE0/2/0/1
  address-family ipv4
   hsrp 1 version 1
    name s1
     address learn
     address 1198.51.100.1 secondary
     authentication company1
     hsrp 2 slave
      follow s1
      address 192.0.2.1
     !
   !
  !
 !
!
```

Configuring a Primary Virtual IPv4 Address

To enable hot standby protocol for IP, use the **address (hsrp)** command in the HSRP group submode.

Configuration Example

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.

3. Enable HSRP group configuration mode on a specific interface.

Note

- The **version** keyword is available only if IPv4 address-family is selected. By default, version is set to 2 for IPv6 address families.
- The HSRP version 2 extended group range must be restricted to 0-255, even though the configuration up to 0-4095 is accepted.

4. Enable hot standby protocol for IP.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP group configuration mode on a specific interface. */
Router(config-hsrp-ipv4)# hsrp <group-number> version <version-no>

/* Enable hot standby protocol for IP. */
Router(config-hsrp-ipv4)# address { learn | address }
```

Running Configuration

```
Router# show running-configuration
router hsrp
 interface TenGigE 0/2/0/1
  address-family ipv4
   hsrp 1 version 1
    address learn
   !
  !
 !
!
```

Configuring a Secondary Virtual IPv4 Address

To configure the secondary virtual IPv4 address for a router, use the **address secondary** command in the Hot Standby Router Protocol (HSRP) virtual router submode.

Configuration Example

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP group configuration mode on a specific interface.

Note

- The **version** keyword is available only if IPv4 address-family is selected. By default, version is set to 2 for IPv6 address families.
- The HSRP version 2 extended group range must be restricted to 0-255, even though the configuration up to 0-4095 is accepted.

4. Configure the secondary virtual IPv4 address for a router.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP group configuration mode on a specific interface. */
Router(config-hsrp-ipv4)# hsrp <group-number> version <version-no>

/* Configure the secondary virtual IPv4 address for a router. */
Router(config-hsrp-ipv4)# address <address> secondary
```

Running Configuration

```
Router# show running-configuration
router hsrp
 interface TenGigE 0/2/0/1
  address-family ipv4
   hsrp 1 version 1
    192.0.2.1
   !
  !
 !
!
```

Configuring the Subordinate Group to Inherit its State from a Specified Group

To instruct the subordinate group to inherit its state from a specified group, use the **hsrp slave follow** command in HSRP slave submode mode.

Configuration Example

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP slave configuration mode on a specific interface.
4. Configure the subordinate group to inherit its state from a specified group.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>
```

```
/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP slave configuration mode on a specific interface. */
Router(config-hsrp-ipv4)# hsrp <group-no> slave

/* Configure the subordinate group to inherit its state from a specified group. */
Router(config-hsrp-slave)# address <ip-address>
```

Running Configuration

```
Router# show running-configuration
router hsrp
 interface TenGigE 0/2/0/1
  address-family ipv4
   hsrp 1 slave
    address 192.0.2.1
   !
  !
 !
!
```

Configuring a Subordinate Primary Virtual IPv4 Address

To configure the primary virtual IPv4 address for the subordinate group, use the **subordinate primary virtual IPv4 address** command in the HSRP slave submode.

Configuration Example

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP slave configuration mode on a specific interface.
4. Configure the primary virtual IPv4 address for the subordinate group.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP slave configuration mode on a specific interface. */
Router(config-hsrp-ipv4)# hsrp <group-no> slave

/* Configure the primary virtual IPv4 address for the subordinate group. */
Router(config-hsrp-slave)# address <ip-address>
```

Running Configuration

```
Router# show running-configuration
router hsrp
 interface TenGigE 0/2/0/1
  address-family ipv4
   hsrp 1 slave
    address 192.0.2.1
   !
  !
```

```
 !
!
```

Configuring a Secondary Virtual IPv4 address for the Subordinate Group

Perform this task to configure the secondary virtual IPv4 address for the subordinate group.

Configuration Example

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP slave configuration mode on a specific interface.
4. Configure the secondary virtual IPv4 address for a router.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP slave configuration mode on a specific interface. */
Router(config-hsrp-ipv4)# hsrp <group-no> slave

/* Configure the secondary virtual IPv4 address for the subordinate group. */
Router(config-hsrp-slave)# address <ip-address> secondary
```

Running Configuration

```
Router# show running-configuration
router hsrp
 interface TenGigE 0/2/0/1
  address-family ipv4
   hsrp 1 slave
    address 192.0.2.1 secondary
   !
  !
 !
!
```

Configuring a Subordinate Virtual MAC Address

To configure the virtual MAC address for the subordinate group, use the **subordinate virtual mac address** command in the HSRP slave submode.

Configuration Example

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP slave configuration mode on a specific interface.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP slave configuration mode on a specific interface. */
Router(config-hsrp-ipv4)# hsrp <group-no> slave
```

Running Configuration

```
Router# show running-configuration
router hsrp
 interface TenGigE 0/2/0/1
  address-family ipv4
   hsrp 1 slave
   !
  !
 !
!
```

Configuring an HSRP Session Name

To configure an HSRP session name, use the **session name** command in the HSRP group submode.

Configuration Example

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP group configuration mode on a specific interface.

Note

- The **version** keyword is available only if IPv4 address-family is selected. By default, version is set to 2 for IPv6 address families.
- The HSRP version 2 extended group range must be restricted to 0-255, even though the configuration up to 0-4095 is accepted.

4. Configure the subordinate group to inherit its state from a specified group.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP group configuration mode on a specific interface. */
Router(config-hsrp-ipv4)# hsrp <group-no> hsrp <version-no>

/* Configure the subordinate group to inherit its state from a specified group. */
Router(config-hsrp-ipv4)# name <name>
```

Running Configuration

```
Router# show running-configuration
router hsrp
 interface TenGigE 0/2/0/1
  address-family ipv4
   hsrp 1 version 2
    name s1   !
  !
 !
!
```

BFD for HSRP

BFD for HSRP

HSRP uses BFD to detect link failure and facilitate fast failover times without excessive control packet overhead.The HSRP process creates BFD sessions as required. When a BFD session goes down, each Standby group monitoring the session transitions to Active state. HSRP doesn't participate in any state elections for 10 seconds, after a transition to Active state that a BFD session going down triggers.

Advantages of BFD

- BFD provides failure detection in less than one second.
- BFD supports most of the encapsulation types.
- BFD is not tied to any particular routing protocol, supports almost all routing protocols.

Guidelines and Restrictions for BFD for HSRP (Concept)

- The routers support up to 255 sessions each for both IPv4 and IPv6, allowing for a total of up to 510 sessions in NCS 5700 fixed port routers and line cards in Native mode, in compatible mode only 255 sessions for both IPv4 and IPv6 together supported. While using maximum BFD sessions, we recommend to use a minimum BFD timmer of 100ms with a multiplier of 3.
- Minimum interval determines the frequency of sending BFD packets to BFD peers (in milliseconds). You can configure the minimum interval using the **bfd minimum-interval** command.
- BFD multiplier is the number of times a packet is missed before BFD declares the neighbor down. You can configure the BFD multiplier using the **bfd multiplier** command.

Enabling BFD

The following procedure describes how to enable BFD for HSRP.

1. Enable HSRP interface configuration mode on a specific interface.

```
Router# configure
Router(config)#  router hsrp
Router(config-hsrp)# interface TenGigE 0/2/0/1
```

2. Enable HSRP address-family configuration mode on a specific interface.

```
Router(config-hsrp-if)# address-family ipv4
```

3. Enable HSRP group sub-mode on a specific interface and configure BFD.

```
Router(config-hsrp-ipv4)# hsrp 1 version 1 bfd fast-detect
Router(config-hsrp-ipv4)# commit
Router(config-hsrp-ipv4)# exit
```

4. (Optional) Configure the BFD timer minimum interval.

```
Router(config-hsrp)# hsrp bfd minimum-interval 15 ms
```

5. (Optional) Configure the BFD timer multiplier.

```
Router(config-hsrp)# hsrp bfd multiplier 3
```

6. Verification.

```
Router# show running-configuration router hsrp
router hsrp
 interface TenGigE 0/2/0/1
  address-family ipv4
   hsrp 1 version 1
     bfd fast-detect
     !
    minimum-interval 15 ms
    multiplier 3
    !
   !
  !
```

Modifying BFD timers (minimum interval)

Minimum interval determines the frequency of sending BFD packets to BFD peers (in milliseconds). The default minimum interval is 15ms.

```
Router# configure
Router(config)#  router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>
Router(config-hsrp-if)# hsrp bfd minimum-interval <interval>
router(config-hsrp-if)# address-family ipv4
commit
```

Modifying BFD timers (multiplier)

```
Router# configure
Router(config)#  router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>
Router(config-hsrp-if)# hsrp bfd multiplier <multiplier>
```

```
Router(config-hsrp-if)# address-family ipv4
commit
```

Enhanced Object Tracking for HSRP and IP Static

A failure between the active router and the core network cannot be detected using standard HSRP failure detection mechanisms. Object tracking is used to detect such failures. When such a failure occurs, the active router applies a priority decrement to its HSRP session. If this causes its priority to fall below that of the standby router, it will detect this from the HSRP control traffic, and then use this as a trigger to preempt and take over the active role.

Cisco IOS XR software supports up to 512 tracked objects.

The enhanced object tracking for HSRP and IP Static feature provides first-hop redundancy as well as default gateway selection based on IP Service Level Agreement (IPSLA).

See the *Routing Configuration Guide*, for more information about enhanced object tracking for static routes.

Configuring object tracking for HSRP

To enable tracking of the named object with the specified decrement, use the following configuration in the HSRP group sub mode.

Configuration Example

1. Enable HSRP interface configuration mode on a specific interface.
2. Enable HSRP address-family configuration mode on a specific interface.
3. Enable HSRP group sub-mode on a specific interface.

Note The **version** keyword is available only if IPv4 address-family is selected. By default, version is set to 2 for IPv6 address families.

4. Enable tracking of the named object with the specified decrement.

Configuration

```
/* Enable HSRP interface configuration mode on a specific interface. */
Router# configure
Router(config)# router hsrp
Router(config-hsrp)# interface <type> <interface-path-id>

/* Enable HSRP address-family configuration mode on a specific interface. */
Router(config-hsrp-if)# address-family ipv4

/* Enable HSRP group sub-mode on a specific interface. */
Router(config-hsrp-ipv4)# hsrp <group-no> hsrp <version-no>

/* Enable tracking of the named object with the specified decrement. */
Router(config-hsrp-gp)# track object <name> [priority-decrement]
```

Running Configuration

```
Router# show running-configuration
router hsrp
 interface TenGigE 0/2/0/1
  address-family ipv4
   hsrp 1 version 1
    track object t1 2
   !
  !
 !
!
```

Hot Restartability for HSRP

In the event of failure of a HSRP process in one active group, forced failovers in peer HSRP active router groups should be prevented. Hot restartability supports warm RP failover without incurring forced failovers to peer HSRP routers for active groups.

Configuration Examples for HSRP Implementation on Software

This section provides the following HSRP configuration examples:

Configuring an HSRP Group: Example

The following is an example of enabling HSRP on an interface and configuring HSRP group attributes:

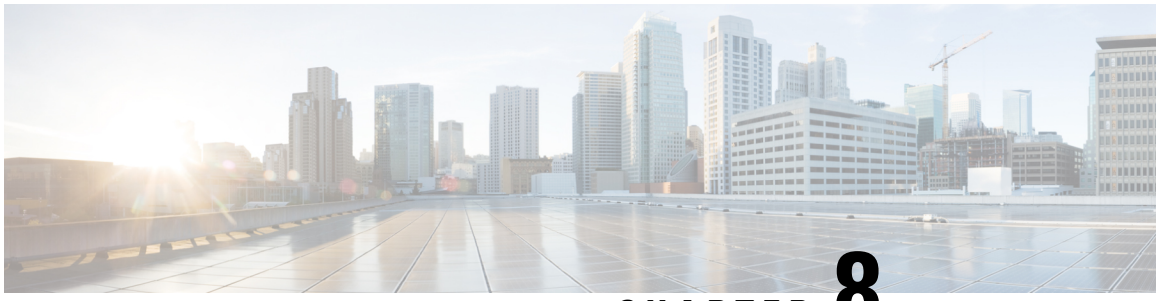
```
configure
router hsrp
interface 0/2/0/1
hsrp 1 ipv4 1.0.0.5
commit
hsrp 1 timers 100 200
hsrp 1 preempt delay 500
hsrp priority 20
hsrp track  0/2/0/2
hsrp 1 authentication company0
hsrp use-bia
commit
```

Configuring a Router for Multiple HSRP Groups: Example

The following is an example of configuring a router for multiple HSRP groups:

```
configure
router hsrp
interface  0/2/0/3
hsrp 1 ipv4 1.0.0.5
hsrp 1 priority 20
hsrp 1 preempt
hsrp 1 authentication sclara
hsrp 2 ipv4 1.0.0.6
hsrp 2 priority 110
hsrp 2 preempt
hsrp 2 authentication mtview
```

```
hsrp 3 ipv4 1.0.0.7
hsrp 3 preempt
hsrp 3 authentication svale
commit
```



CHAPTER 8

Implementing LPTS

- LPTS Overview, on page 157
- LPTS Policers, on page 157
- LPTS Domain Based Policers, on page 162

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

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 are able to configure the rate per policer per node.

Note

- You can get the default policer values and the current rates of the flow types from the output of the following show command:

  ```
  show lpts pifib hardware police
  ```

- For quick file transfer through a data port, you can configure LPTS policer rate for SSH flow.

  Verify that the LPTS drops using the command, **show lpts pifib hardware entry brief location** *node-id* **|inc SSH**. If there are any LPTS drops, increase the rate up to a maximum of 50000 pps.

  Increase the value to the maximum only if required, as the CPU cycles usage increases with higher PPS.

  For example,

  ```
  Router#configure
  Router(config)#lpts pifib hardware police location 0/0/CPU0
  Router(config-pifib-policer-per-node)# flow ssh known rate 50000
  Router(config-pifib-policer-per-node)#commit
  ```

-

Verification

This show **show lpts pifib hardware entry brief location** command is updated to display the statistics of the flow types. The counters are printed under the OOS field description. The * indicates the statistics of the resources are exhausted. Note, that the LPTS functionality is not impacted.

```
RP/0/RP0/CPU0:Router# show lpts pifib hardware entry brief location 0/3/CPU0
Tue Dec 22 10:57:08.322 UTC

-------------------------------------------------------------------
             Node: 0/RP0/CPU0
-------------------------------------------------------------------
     G  - Global flowtype counters
    (*) - stats resources exhausted,
          stats are shared per flow type
-------------------------------------------------------------------

Type DestIP           SrcIP            Interface        vrf   L4     LPort/Type     RPort
 npu  Flowtype         DestNode   PuntPrio Accept Drop   Domain           OOS
---- ---------------- ---------------- ---------------- ----- ------ -------------- ------
 ---- ---------------- ---------- -------- ------ ------ ---------------- ------
IPV4 any              any              any              0     0      any            0
 0    Fragment         Local LC   LOW      0      0      0-default
IPV4 224.0.0.5        any              BE105.201        0     89     any            0
 0    OSPF-mc-known    Dlvr RP0   HIGH     1      0      0-default        *
IPV4 224.0.0.5        any              BE105.202        0     89     any            0
 0    OSPF-mc-known    Dlvr RP0   HIGH     1      0      0-default        *
IPV4 224.0.0.5        any              BE105.203        0     89     any            0
 0    OSPF-mc-known    Dlvr RP0   HIGH     1      0      0-default        *
IPV4 224.0.0.5        any              BE105.204        0     89     any            0
 0    OSPF-mc-known    Dlvr RP0   HIGH     1      0      0-default        *
IPV4 224.0.0.5        any              BE105.205        0     89     any            0
 0    OSPF-mc-known    Dlvr RP0   HIGH     1      0      0-default        *
IPV4 224.0.0.5        any              BE105.206        0     89     any            0
 0    OSPF-mc-known    Dlvr RP0   HIGH     1      0      0-default        *
```

Configuration Example

Configure the LPTS policer for the OSPF and BGP flow types 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-pifib-policer-global)#flow ospf unicast default rate 3000
Router(config-pifib-policer-global)#flow bgp default rate 4000
Router (config-pifib-policer-global)#commit
```

Running Configuration

```
lpts pifib hardware police
flow ospf unicast default rate 3000
flow bgp default rate 4000
!
```

Verification

```
Router#show run lpts pifib hardware police
lpts pifib hardware police
flow ospf unicast default rate 3000
flow bgp default rate 4000
```

Configuration Example

Configure the LPTS policer for the OSPF and BGP flow types with the following values on an individual node - 0/RP0/CPU0:

- ospf unicast default rate 3000
- flow bgp default rate 4000

```
Router#configure
Router(config)#lpts pifib hardware police location 0/RP0/CPU0
Router(config-pifib-policer-per-node)#flow ospf unicast default rate 3000
Router(config-pifib-policer-per-node)#flow bgp default rate 4000
Router(config-pifib-policer-per-node)#commit
```

Running Configuration

```
lpts pifib hardware police location 0/RP0/CPU0
flow ospf unicast default rate 3000
flow bgp default rate 4000
```

Verification

The **show lpts pifib hardware police location 0/RP0/CPU0** command displays pre-Internal Forwarding Information Base (IFIB) information for the designated node.

```
Router#show lpts pifib hardware police location 0/RP0/CPU0
-------------------------------------------------------------
               Node 0/RP0/CPU0:
-------------------------------------------------------------
 Burst = 100ms for all flow types
-------------------------------------------------------------
FlowType               Policer Type    Cur. Rate Burst     npu
```

```
---------------------- ------- ------- --------- --------- ---------
OSPF-uc-default        32106   np      3000      1000      0
BGP-default            32118   np      4000      1250      0
```

Verification

The **show controllers npu stats traps-all instance all location 0/RP0/CPU0** command displays packets that are locally processed and packets that are dropped by the CPU.

```
Router# show controllers npu stats traps-all instance all location 0/RP0/CPU0

Trap Type                                       NPU  Trap TrapStats   Policer Packet    Packet

                                                ID    ID      ID              Accepted  Dropped
=============================================================================================
RxTrapMimSaMove(CFM_DOWM_MEP_DMM)                 0    6    0x6          32037   0         0

RxTrapMimSaUnknown(RCY_CFM_DOWN_MEP_DMM)          0    7    0x7          32037   0         0

RxTrapAuthSaLookupFail (IPMC default)             0    8    0x8          32033   0         0

RxTrapSaMulticast                                 0    11   0xb          32018   0         0

RxTrapArpMyIp                                     0    13   0xd          32001   0         0

RxTrapArp                                         0    14   0xe          32001   11        0

RxTrapDhcpv4Server                                0    18   0x12         32022   0         0

RxTrapDhcpv4Client                                0    19   0x13         32022   0         0

RxTrapDhcpv6Server                                0    20   0x14         32022   0         0

RxTrapDhcpv6Client                                0    21   0x15         32022   0         0

RxTrapL2Cache_LACP                                0    23   0x17         32003   0         0

RxTrapL2Cache_LLDP1                               0    24   0x18         32004   0         0

RxTrapL2Cache_LLDP2                               0    25   0x19         32004   1205548   0

RxTrapL2Cache_LLDP3                               0    26   0x1a         32004   0         0

RxTrapL2Cache_ELMI                                0    27   0x1b         32005   0         0

RxTrapL2Cache_BPDU                                0    28   0x1c         32027   0         0

RxTrapL2Cache_BUNDLE_BPDU                         0    29   0x1d         32027   0         0

RxTrapL2Cache_CDP                                 0    30   0x1e         32002   0         0

RxTrapHeaderSizeErr                               0    32   0x20         32018   0         0

RxTrapIpCompMcInvalidIp                           0    35   0x23         32018   0         0

RxTrapMyMacAndIpDisabled                          0    36   0x24         32018   0         0

RxTrapMyMacAndMplsDisable                         0    37   0x25         32018   0         0

RxTrapArpReply                                    0    38   0x26         32001   2693      0

RxTrapFibDrop                                     0    41   0x29         32018   0         0

RxTrapMTU                                         0    42   0x2a         32020   0         0
```

RxTrapMiscDrop	0	43	0x2b	32018	0	0
RxTrapL2AclDeny	0	44	0x2c	32034	0	0
Rx_UNKNOWN_PACKET	0	46	0x2e	32018	0	0
RxTrapL3AclDeny	0	47	0x2f	32034	0	0
RxTrapOamY1731MplsTp(OAM_SWOFF_DN_CCM)	0	57	0x39	32029	0	0
RxTrapOamY1731Pwe(OAM_SWOFF_DN_CCM)	0	58	0x3a	32030	0	0
RxTrapOamLevel	0	64	0x40	32023	0	0
RxTrapRedirectToCpuOamPacket	0	65	0x41	32025	0	0
RxTrapOamPassive	0	66	0x42	32024	0	0
RxTrap1588	0	67	0x43	32038	0	0
RxTrapExternalLookupError	0	72	0x48	32018	0	0
RxTrapArplookupFail	0	73	0x49	32001	0	0
RxTrapUcLooseRpfFail	0	84	0x54	32035	0	0
RxTrapMplsControlWordTrap	0	88	0x58	32015	0	0
RxTrapMplsControlWordDrop	0	89	0x59	32015	0	0
RxTrapMplsUnknownLabel	0	90	0x5a	32018	0	0
RxTrapIpv4VersionError	0	98	0x62	32018	0	0
RxTrapIpv4ChecksumError	0	99	0x63	32018	0	0
RxTrapIpv4HeaderLengthError	0	100	0x64	32018	0	0
RxTrapIpv4TotalLengthError	0	101	0x65	32018	0	0
RxTrapIpv4Ttl0	0	102	0x66	32008	0	0
RxTrapIpv4Ttl1	0	104	0x68	32008	0	0
RxTrapIpv4DipZero	0	106	0x6a	32018	0	0
RxTrapIpv4SipIsMc	0	107	0x6b	32018	0	0
RxTrapIpv6VersionError	0	109	0x6d	32018	0	0
RxTrapIpv6HopCount0	0	110	0x6e	32011	0	0
RxTrapIpv6LoopbackAddress	0	113	0x71	32018	0	0
RxTrapIpv6MulticastSource	0	114	0x72	32018	0	0
RxTrapIpv6NextHeaderNull	0	115	0x73	32010	0	0
RxTrapIpv6Ipv4CompatibleDestination	0	121	0x79	32018	0	0
RxTrapMplsTtl1	0	125	0x7d	32012	316278	2249
RxTrapUcStrictRpfFail	0	137	0x89	32035	0	0

RxTrapMcExplicitRpfFail	0	138	0x8a	32033	0	0
RxTrapOamp(OAM_BDL_DN_NON_CCM)	0	141	0x8d	32031	0	0
RxTrapOamEthUpAccelerated(OAM_BDL_UP_NON_CCM)	0	145	0x91	32032	0	0
RxTrapReceive	0	150	0x96	32017	125266112	0
RxTrapUserDefine_FIB_IPV4_NULL0	0	151	0x97	32018	0	0
RxTrapUserDefine_FIB_IPV6_NULL0	0	152	0x98	32018	0	0
RxTrapUserDefine_FIB_IPV4_GLEAN	0	153	0x99	32016	0	0
RxTrapUserDefine_FIB_IPV6_GLEAN	0	154	0x9a	32016	0	0
RxTrapUserDefine_IPV4_OPTIONS	0	155	0x9b	32006	0	0
RxTrapUserDefine_IPV4_RSVP_OPTIONS	0	156	0x9c	32007	0	0
RxTrapUserDefine	0	157	0x9d	32026	0	0
RxTrapUserDefine_BFD	0	163	0xa3	32028	0	0
RxTrapMC	0	181	0xb5	32033	0	0
RxNetflowSnoopTrap0	0	182	0xb6	32018	0	0
RxNetflowSnoopTrap1	0	183	0xb7	32018	0	0
RxTrapMimSaMove(CFM_DOWM_MEP_DMM)	1	6	0x6	32037	0	0
RxTrapMimSaUnknown(RCY_CFM_DOWN_MEP_DMM)	1	7	0x7	32037	0	0
RxTrapAuthSaLookupFail (IPMC default)	1	8	0x8	32033	0	0
RxTrapSaMulticast	1	11	0xb	32018	0	0
RxTrapArpMyIp	1	13	0xd	32001	0	0

Associated Commands

- lpts pifib hardware police
- flow ospf
- flow bgp
- show lpts pifib hardware police

LPTS Domain Based Policers

You can configure a particular port, a group of ports, or a line card of a router with LPTS policers of a single domain. Configuration of port-based policers that belong to a particular domain enables better categorisation and control of different types of ingress traffic. For example, since iBGP traffic has a higher rate of traffic flow, the ports that handle iBGP traffic can be configured with higher policer rates compared to the ports that handle eBGP traffic.

Restrictions

- The policer rates that are configured for ports or line cards are carried forwards as policer rates of the domain after configuring the ports or line cards as part of a domain. For example, if port hundredGigE 0/0/0/1 and port hundredGigE 0/0/0/2 have policer rate of 3000 for ospf unicast known flow and if the ports are configured as part of domain CORE, then the policer rate of domain CORE for ospf unicast known flow is 3000 unless it is configured otherwise.
- You can configure only one domain per router.
- A Domain name can be any word but can have up to a maximum of 32 characters.

Configuration Example

To configure LPTS domain based policers, use the following steps:

1. Enter the LPTS hardware configuration mode and create a domain.
2. Configure the interfaces for the domain.
3. Enter the LPTS hardware configuration mode for the domain CORE, and then configure the ingress policer rates for the domain CORE at the global level.
4. Enter the LPTS hardware configuration mode for the domain CORE, and then configure the ingress policer rates for the domain CORE at the line card level.

Configuration

```
/* Enter the LPTS hardware ingress policer configuration mode and create a domain named
CORE. */
Router# config
Router(config)# lpts pifib hardware domain CORE

/* Configure the interfaces for the domain CORE. */
Router(config-lpts-domains-CORE)# interface hundredGigE 0/0/0/1
Router(config-lpts-domains-CORE)# interface hundredGigE 0/0/0/2
Router(config-lpts-domains-CORE)# commit
Router(config-lpts-domains-CORE)# exit

/* Enter the LPTS hardware configuration mode for the domain CORE, and then configure the
ingress policer rates for the domain CORE at the global level. */
Router(config)# lpts pifib hardware police domain CORE
Router(config-lpts-policer-global-CORE)# flow ospf unicast known rate 6000
Router(config-lpts-policer-global-CORE)# flow ospf unicast default rate 7000
Router(config-lpts-policer-global-CORE)# commit
Router(config-lpts-policer-global-CORE)# exit
Router(config-lpts-policer-global)# exit

/* Enter the LPTS hardware configuration mode for the domain CORE, and then configure the
ingress policer rates for the domain CORE at the line card level. */
Router(config)# lpts pifib hardware police location 0/0/CPU0 domain CORE
Router(config-lpts-policer-global-CORE)# flow ospf unicast known rate 7000
Router(config-lpts-policer-global-CORE)# flow ospf unicast default rate 8000
Router(config-lpts-policer-global-CORE)# commit
```

Running Configuration

```
lpts pifib hardware domain CORE
 interface HundredGigE0/0/0/1
```

```
 interface HundredGigE0/0/0/2
!
lpts pifib hardware police
 domain CORE
  flow ospf unicast known rate 6000
  flow ospf unicast default rate 7000
 !

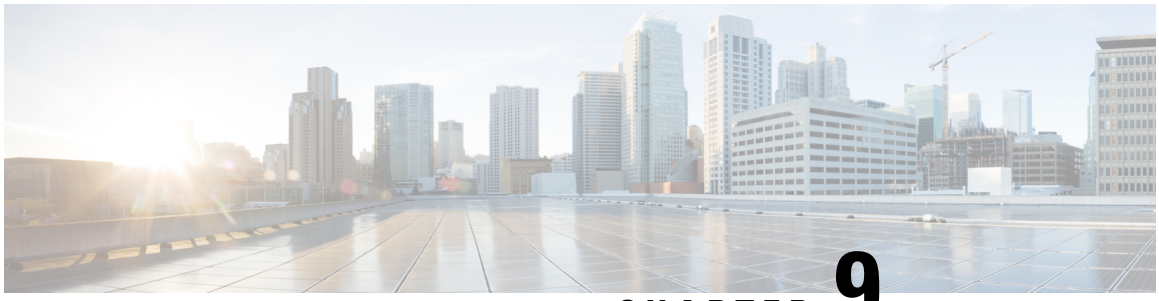
lpts pifib hardware police location 0/0/CPU0 domain CORE
 flow ospf unicast known rate 7000
 flow ospf unicast default rate 8000
 !
```

Verification

Use the following command to verify information about the LPTS domains configured:

```
Router# show lpts pifib domains
Thu Nov 21 15:49:31.334 IST

 Domains Information: 1 Configured
 ---------------------------------
   Domain: [1] CORE
   ------------------------
   interface [----------] HundredGigE0/0/0/1
   interface [----------] HundredGigE0/0/0/2
               0 local of total 2 interfaces
```



CHAPTER 9

Implementing VRRP

- Configuring VRRP, on page 165
- Enabling Multiple Group Optimization (MGO) for VRRP, on page 175
- Configuring SNMP Server Notifications for VRRP Events, on page 177

Configuring 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 Understanding VRRP, on page 165

Restrictions for Configuring VRRP

- ICMP redirects are not supported.
- The maximum VRRP supported is only 16, but this scale number may vary or reduce further based on your BFD, BVI V4 and V6 configuration. For example, if BFD is configured then the value becomes 16-1=15. If BVI v4 and BVI v6 are also configured along with BFD, then the value is 16-3=13 only.

Understanding 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.

Note VRRP is supported over VRF.

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.

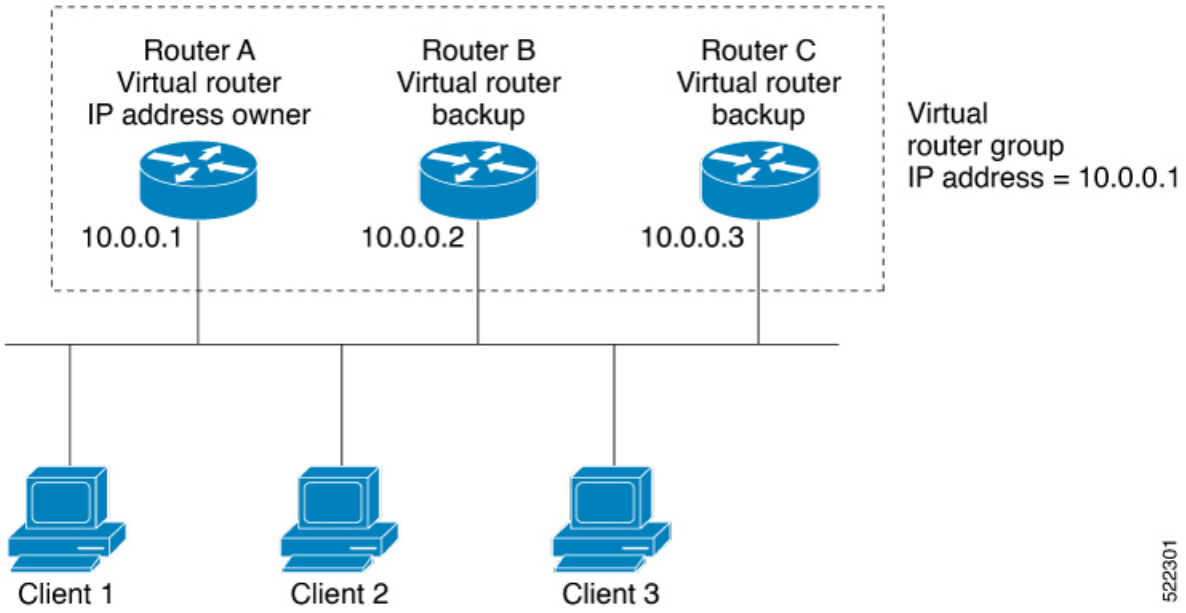
The Virtual Router Redundancy Protocol (VRRP) feature can solve the static configuration problem. VRRP is an IP routing redundancy protocol designed to allow for transparent failover at the first-hop IP router. VRRP enables a group of routers to form a single *virtual router* . The LAN clients can then 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, Figure 14: Basic VRRP Topology, on page 167 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 14: 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.

Customizing VRRP

Configuration Example

Customizing the behavior of VRRP is optional. Be aware that as soon as you enable a VRRP group, that group is operating. It is possible that if you first enable a VRRP group before customizing VRRP, the router could take over control of the group and become the master virtual router before you have finished customizing the feature. Therefore, if you plan to customize VRRP, it is a good idea to do so before enabling VRRP.

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2

router(config-vrrp)#delay minimum 2 reload 10
/* (Optional) Delays the startup of the state machine when an interface comes up. */

router(config-vrrp-if)#address-family ipv6
router(config-vrrp-address-family)#vrrp 3
/* The version keyword is available only if IPv4 address-family is selected. */



router(config-vrrp-virtual-router)#accept-mode disable
/* Disables the installation of routes for the VRRP virtual addresses. */

router(config-vrrp-virtual-router)#priority 254
/* (Optional) Sets the priority of the virtual router. */

router(config-vrrp-virtual-router)#preempt delay 15
/* (Optional) Controls which router becomes the master router. */

router(config-vrrp-virtual-router)#timer 4
/* (Optional) Configures the interval between successive advertisements by the master router
 in a VRRP virtual router. */

router(config-vrrp-virtual-router)#track interface TenGigE 0/0/0/2 30
/* (Optional) Configures the VRRP to track an interface. */

router(config-vrrp-virtual-router)#commit
```

Running Configuration

```
Router#show running-config router vrrp
router vrrp
interface TenGigE 0/0/0/2
delay minimum 2 reload 10
address-family ipv6
```

```
vrrp 3
text-authentication
accept-mode disable
priority 254
preempt delay 15
timer 4
track interface TenGigE 0/0/0/2 30
!
```

Verification

```
Router#show vrrp detail

TenGigE0/0/0/2 - IPv4 vrID 3
  State is Master, IP address owner
    1 state changes, last state change 00:01:00
    State change history:
    May 19 12:28:59.825 UTC  Init     -> Master   Virtual IP configured
  Last resign sent:     Never
  Last resign received: Never
  Virtual IP address is 10.0.0.1
  Virtual MAC address is 0000.5E00.0103, state is active
  Master router is local
  Version is 2
  Advertise time 4 secs
    Master Down Timer 12.015 (3 x 4 + (1 x 4/256))
  Minimum delay 2 sec, reload delay 10 sec
  Current priority 255
    Configured priority 254, may preempt
      minimum delay 15 secs
  Authentication enabled, string "text1"
    Tracked items: 1/1 up: 30 decrement
      Object name                State     Decrement
      TenGigE0/0/0/2              Up            30
```

Enabling VRRP

Configuration Example

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2
router(config-vrrp-if)#address-family ipv4
router(config-vrrp-address-family)#vrrp 3 version 3
/* The version keyword is available only if IPv4 address-family is selected. */

router(config-vrrp-virtual-router)#address 10.20.30.1
/* Enables VRRP on an interface and specifies the IP address of the virtual router. */

router(config-vrrp-virtual-router)#commit
```

Running Configuration

```
Router#show running-config router vrrp
router vrrp
interface TenGigE 0/0/0/2
address-family ipv4
vrrp 3 version 3
```

```
address 10.20.30.1
!
```

Verification

```
Router#show vrrp detail

TenGigE0/0/0/2 - IPv4 vrID 3
  State is Master, IP address owner
    1 state changes, last state change 00:01:00
    State change history:
    May 19 12:28:59.825 UTC  Init     -> Master   Virtual IP configured
  Last resign sent:     Never
  Last resign received: Never
  Virtual IP address is 10.20.30.1
  Virtual MAC address is 0000.5E00.0103, state is active
  Master router is local
  Version is 2
  Advertise time 4 secs
    Master Down Timer 12.015 (3 x 4 + (1 x 4/256))
  Current priority 255
```

Clearing VRRP Statistics

Clears all the software counters for the specified virtual router.

```
Router#clear vrrp statistics
/* If no interface is specified, statistics of all virtual routers are removed. */
```

Configuring a Global Virtual IPv6 Address

Configuration Example

Configures the global virtual IPv6 address for a virtual router.

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2
router(config-vrrp-if)#address-family ipv6
router(config-vrrp-address-family)#vrrp 3
/* The version keyword is available only if IPv4 address-family is selected. */

router(config-vrrp-if-virtual-router)#address global 2001:db8::
router(config-vrrp-virtual-router)#commit
```

Running Configuration

```
Router#show running-config router vrrp
router vrrp
interface TenGigE 0/0/0/2
address-family ipv6
vrrp 3
address global 2001:db8::
!
```

Configuring the Primary and Secondary Virtual IPv4 Addresses

Configuration Example

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2
router(config-vrrp-if)#address-family ipv4
router(config-vrrp-address-family)#vrrp 3 version 3
/* The version keyword is available only if IPv4 address-family is selected. */

router(config-vrrp-if-virtual-router)#address 10.20.30.1
/* Configures primary virtual IPv4 address for a virtual router. */

router(config-vrrp-if-virtual-router)#address 10.20.30.2 secondary
/* Configures secondary virtual IPv4 address for a virtual router. */

router(config-vrrp-virtual-router)#commit
```

Running Configuration

```
Router#show running-config router vrrp
router vrrp
interface TenGigE 0/0/0/2
address-family ipv4
vrrp 3 version 3
address 10.20.30.1
address 10.20.30.2 secondary
!
```

Verification

```
Router#show vrrp detail

TenGigE0/0/0/2 - IPv4 vrID 3
  State is Master, IP address owner
    1 state changes, last state change 00:01:00
    State change history:
    May 19 12:28:59.825 UTC  Init     -> Master   Virtual IP configured
  Last resign sent:     Never
  Last resign received: Never
  Virtual IP address is 10.20.30.1
  Virtual MAC address is 0000.5E00.0103, state is active
  Master router is local
  Virtual secondary IP address is 10.20.30.2
  Version is 2
  Advertise time 4 secs
    Master Down Timer 12.015 (3 x 4 + (1 x 4/256))
  Current priority 255
```

Configuring a Virtual Link-Local IPv6 Address

Configures either the virtual link-local IPv6 address for a virtual router or specifies that the virtual link-local IPv6 address be enabled and calculated automatically from the virtual router virtual MAC address.

The IPv6 address space is structured differently compared to IPv4. Link-local addresses are used to identify each interface on the local network. These addresses may either be configured or determined automatically in a standard way using the link-layer (hardware) address of the interface (MAC address for Ethernet interfaces). Link-local addresses have a standard format and are valid only on the local network (they cannot be routed to, from multiple hops away).

Global unicast IPv6 addresses occupy a disjoint subset of the IPv6 address space from link-local addresses. They can be routed to, from multiple hops away and have an associated prefix length (between 0 and 128 bits).

Each VRRP virtual router has an associated virtual link-local address. This may be configured or determined automatically from the virtual router's virtual MAC address. The virtual MAC address must be unique on the local network. The virtual link-local address is analogous to an IPv4 virtual router's primary virtual IPv4 address, except that its virtual IP (VIP) state is always considered to be up, since duplicate address detection is not required for addresses whose scope is local.

Configuration Example

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2
router(config-vrrp-if)#address-family ipv6

/* Use one of the following address linklocal commands: */
router(config-vrrp-address-family)#vrrp 1 address linklocal FE80::260:3EFF:FE11:6770
/* Configures the virtual link-local IPv6 address for the virtual router. */

router(config-vrrp-address-family)#vrrp 1 address linklocal autoconfigure
/* Specifies that the virtual link-local IPv6 address should be enabled and calculated
automatically
from the virtual router virtual MAC address. */

router(config-vrrp-virtual-router)#commit
```

Running Configuration

```
Router#show running-config router vrrp
router vrrp
interface TenGigE 0/0/0/2
address-family ipv6
vrrp 1 address linklocal FE80::260:3EFF:FE11:6770
!
```

BFD for VRRP

Bidirectional Forwarding Detection (BFD) is a network protocol used to detect faults between two forwarding engines. BFD sessions operate in asynchronous mode. In asynchronous mode, both endpoints periodically send hello packets to each other. If a number of those packets are not received, the session is considered down.

Advantages of BFD

- BFD provides failure detection in less than one second.
- BFD supports all types of encapsulation.
- BFD is not tied to any particular routing protocol, supports almost all routing protocols.

BFD Process

VRRP uses BFD to detect a link failure and facilitate fast failover times without excessive control packet overhead.

The VRRP process creates BFD sessions as required. When a BFD session goes down, each backup group monitoring the session transitions to the active state.

After a transition to active state triggered by a BFD session going down, VRRP does not participate in any state elections for 10 seconds.



Note IPv4 only supports BFD for VRRP.

Configuring BFD

Enabling BFD

```
Router# configure
Router(config)#  router vrrp
Router(config-vrrp)# interface <type> <interface-path-id>
Router(config-vrrp-if)# address-family ipv4
Router(config-vrrp-ipv4)# vrrp[group number] version <version-no> bfd
fast-detect [peer ipv4 <ipv4-address> <interface-type> <interface-path-id>]
commit
```

Verifying BFD on VRRP

```
router vrrp
interface TenGigE0/0/0/3.1
bfd minimum-interval 4
bfd multiplier 3
address-family ipv4
vrrp 1
priority 200
address 41.41.1.3
bfd fast-detect peer ipv4 41.41.1.2
```

Modifying BFD timers (minimum interval)

Minimum interval determines the frequency of sending BFD packets to BFD peers (in milliseconds). The default minimum interval is 15ms.

```
Router# configure
```

```
Router(config)#  router vrrp
Router(config-vrrp)# interface <type> <interface-path-id>
Router(config-vrrp-if)# bfd minimum-interval <interval>
Router(config-vrrp-if)# bfd multiplier <multiplier>
router(config-vrrp-if)# address-family ipv4
commit
```

Modifying BFD timers (multiplier)

Multiplier is the number of consecutive BFD packets which must be missed from a BFD peer before declaring that peer unavailable. The default multiplier is 3.

```
Router# configure
Router(config)#  router vrrp
Router(config-vrrp)# interface <type> <interface-path-id>
Router(config-vrrp-if)#  bfd multiplier <multiplier>
router(config-vrrp-if)# address-family ipv4
commit
```

Disabling State Change Logging

Configuration Example

Disables the task of logging the VRRP state change events via syslog.

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#message state disable
router(config-vrrp)#commit
```

Enabling Multiple Group Optimization (MGO) for VRRP

Configuration Examples

Multiple Group Optimization for Virtual Router Redundancy Protocol (VRRP) provides a solution for reducing control traffic in a deployment consisting of many subinterfaces. By running the VRRP control traffic for just one session, the control traffic is reduced for the subinterfaces with identical redundancy requirements. All other sessions are subordinates of this primary session, and inherit their states from it.

Configuring VRRP Session Name

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2
router(config-vrrp-if)#address-family ipv4
router(config-vrrp-address-family)#vrrp 1
/* Enables VRRP group configuration mode on a specific interface. */
```

```
router(config-vrrp-vritual-router)#name m1
/* Specifies the VRRP session name. */

router(config-vrrp-gp)#commit
```

Configuring the Subordinate Group to Inherit its State from a Specified Group

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2
router(config-vrrp-if)#address-family ipv4

router(config-vrrp-address-family)#vrrp 2 slave
/* Enables VRRP slave configuration mode on a specific interface. */

router(config-vrrp-slave)#follow m1
/* Instructs the subordinate group to inherit its state from the specified group, m1 (MGO
session name). */

router(config-vrrp-slave)#address 10.2.3.2
/* Specifies the primary virtual IPv4 address for subordinate group. */

router(config-vrrp-slave)#address 10.2.3.3 secondary
/* Specifies the secondary virtual IPv4 address for subordinate group. */

router(config-vrrp-gp)#commit
```

Primary and Secondary Virtual IPv4 Addresses for the Subordinate Group

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2
router(config-vrrp-if)#address-family ipv4

router(config-vrrp-address-family)#vrrp 2 slave
/* Enables VRRP slave configuration mode on a specific interface. */

router(config-vrrp-slave)#address 10.2.3.2
/* Specifies the primary virtual IPv4 address for subordinate group. */

router(config-vrrp-slave)#address 10.2.3.3 secondary
/* Specifies the secondary virtual IPv4 address for subordinate group. */

router(config-vrrp-slave)#commit
```

Running Configuration

```
Router#show running-config router vrrp 1
router vrrp
interface TenGigE 0/0/0/2
address-family ipv4
vrrp 1
name m1
!

/* Subordinate group */
Router#show running-config router vrrp 2
router vrrp
interface TenGigE 0/0/0/2
address-family ipv4
vrrp 2 slave
follow m1
address 10.2.3.2
address 10.2.3.3 secondary
```

!

Configuring SNMP Server Notifications for VRRP Events

MIB support for VRRP

VRRP enables one or more IP addresses to be assumed by a router when a failure occurs. For example, when IP traffic from a host reaches a failed router because the failed router is the default gateway, the traffic is transparently forwarded by the VRRP router that has assumed control. VRRP does not require configuration of dynamic routing or router discovery protocols on every end host. The VRRP router controlling the IP address(es) associated with a virtual router is called the IP address owner router, and forwards packets sent to these IP addresses. The election process provides dynamic fail over (standby) in the forwarding responsibility should the IP address owner router become unavailable. This allows any of the virtual router IP addresses on the LAN to be used as the default first hop router by end-hosts.

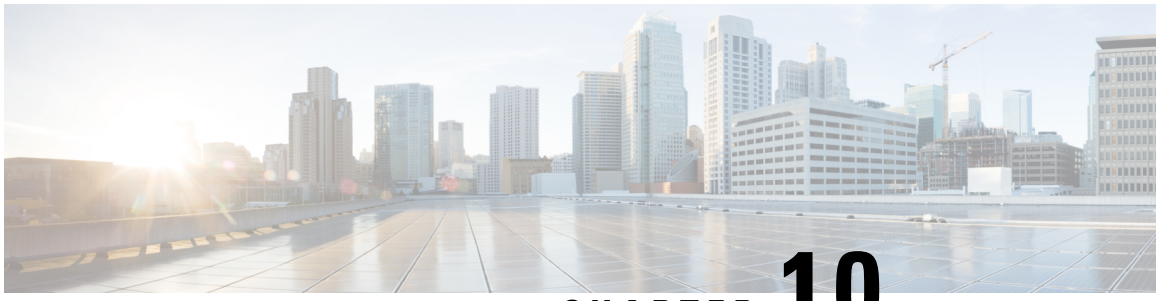
The advantage gained from using VRRP is a higher availability default path without requiring configuration of dynamic routing or router discovery protocols on every end-host. Simple Network Management Protocol (SNMP) traps provide information of the state changes, when the virtual routers (in standby) are moved to IP address owner router's state or if the standby router is made IP address owner router.

Configuration Example

Enables SNMP server notifications (traps) for VRRP.

```
Router#configure
Router(config)#snmp-server traps vrrp events
router(config)#commit
```

Use the **show snmp traps details** command to view details of SNMP server notifications.



CHAPTER 10

Information About Configuring TCP, UDP Transports

To configure TCP, UDP, and RAW transports, you must understand the following concepts:

- NSR Overview, on page 179
- Graceful Restart, on page 180
- TCP Overview, on page 180
- UDP Overview, on page 180
- Configuring Failover as a Recovery Action for NSR, on page 180

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 restart 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) restarts. If a standby NSR-enabled process restarts, 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 Implementing OSPF on Routing Configuration Guide .

Graceful Restart

You can use nonstop forwarding (NSF) for BGP to forward data packets along known routes in the Forward Information Base (FIB) while the BGP routing protocol information is being restored following a failover. With NSF, BGP peers do not experience routing flaps. During a failover, the data traffic is forwarded through intelligent modules while the standby supervisor becomes active.

If a Cisco router experiences a cold reboot, the network does not forward traffic to the router and removes the router from the network topology. In this scenario, BGP experiences a nongraceful restart and removes all routes. When Cisco operating system applies the startup configuration, BGP reestablishes peering sessions and relearns the routes.

A Cisco router that has dual supervisors can experience a stateful supervisor switchover. During the switchover, BGP uses nonstop forwarding to forward traffic based on the information in the FIB, and the system is not removed from the network topology. A router whose neighbor is restarting is referred to as a "helper." After the switchover, a graceful restart operation begins. When it is in progress, both routers reestablish their neighbor relationship and exchange their BGP routes. The helper continues to forward prefixes pointing to the restarting peer, and the restarting router continues to forward traffic to peers although those neighbor relationships are restarting. When the restarting router has all route updates from all BGP peers that are graceful restart capable, the graceful restart is complete, and BGP informs the neighbors that it is operational again.

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.

Configuring Failover as a Recovery Action for NSR

When the active TCP or the NSR client of the active TCP terminates or restarts, 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) restarts or terminates.

For information on how to configure MPLS Label Distribution Protocol (LDP) for NSR, refer to the *MPLS Configuration Guide for Cisco NCS 540 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 NCS 540 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