IP Addresses and Services Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.0.x

First Published: 2020-08-01

Last Modified: 2020-03-13

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

CONTENTS

Preface

Note This release has reached end-of-life status. For more information, see the End-of-Life and End-of-Sale Notices.

This preface contains these sections:

Changes to This Document

Table 1: Changes to This Document

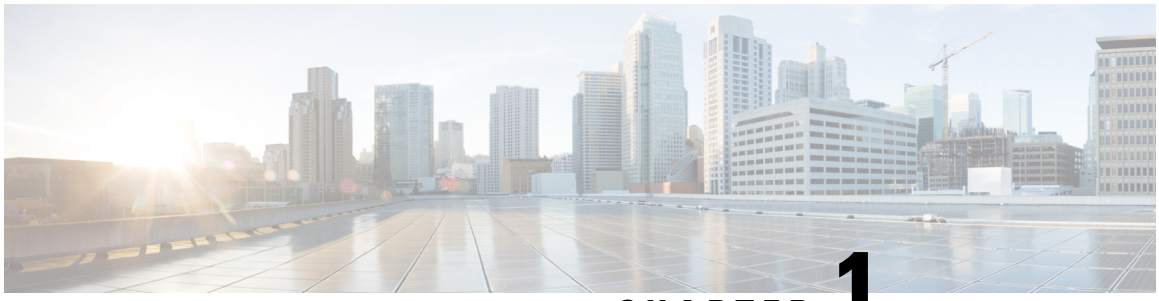
Date	Change Summary
March 2020	Initial release of this document.
August 2020	Republished for Release Release 7.0.14

Communications, Services, and Additional Information

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

Cisco Bug Search Tool

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



CHAPTER 1

New and Changed IP Addresses and Services Features

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

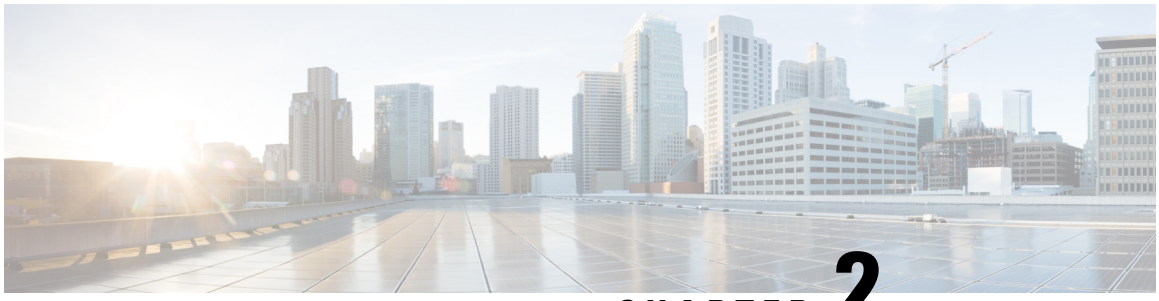
New and Changed IP Addresses and Services Features

IP Addresses Features Added or Modified in IOS XR Release 7.0.x

Table 2: New and Changed Features

Feature	Description	Changed in Release	Where Documented
Support for ACL-based forwarding	Support is added for ACL-based forwarding to filter packets.	Release 7.0.14	Configuring ACL-based Forwarding, on page 76

New and Changed IP Addresses and Services Features



CHAPTER 2

Implementing Network Stack IPv4 and IPv6

Implementing Network Stack IPv4 and IPv6

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

Restrictions

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

Network Stack IPv4 and IPv6 Exceptions

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

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

IPv4 and IPv6 Functionality

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

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

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

IPv6 for Cisco IOS XR Software

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

Prerequisites for Implementing Network Stack IPv4 and IPv6

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

How to Implement Network Stack IPv4 and IPv6

This section contains the following procedures:

Configuring IPv4 Addressing

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

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

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

Configuration Example

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

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

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

Running Configuration

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

Verification

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

```
Router# show ipv4 interface HundredGigE 0/0/0/24
Fri Sep 27 09:36:36.244 UTC
HundredGigE0/0/0/24 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.0.2.27/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
```

IPv4 Virtual Addresses

Configuring an IPv4 virtual address enables you to access the router from a single virtual address with a management network, without the prior knowledge of which route processor (RP) is active. An IPv4 virtual

address persists across RP failover situations. For this to happen, the virtual IPv4 address must share a common IPv4 subnet with a Management Ethernet interface on both RPs.

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

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

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

Configuration Example

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

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

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

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

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

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

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

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

IPv4 ICMP Rate Limiting

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

Configuration Example

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

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

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

Running Configuration

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

Verification

```
Router#show ipv4 interface HundredGigE0/0/0/2
HundredGigE0/0/0/2 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.85.1.1/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Multicast reserved groups joined: 224.0.0.2 224.0.0.1 224.0.0.2
      224.0.0.5 224.0.0.6
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
```

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

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

Enabling IPv4 Processing on an Unnumbered Interface

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

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

Configuration Example

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

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

Running Configuration

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

Verification

```
Router#show interface HundredGigE 0/0/0/25
HundredGigE0/0/0/25 is up, line protocol is up
  Interface state transitions: 5
```

```
  Hardware is HundredGigE, address is 00e2.2a33.445b (bia 00e2.2a33.445b)
  Layer 1 Transport Mode is LAN
  Internet address is 192.0.2.1/24
  MTU 1514 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
     reliability 255/255, txload 194/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 10000Mb/s, link type is force-up
  output flow control is off, input flow control is off
  Carrier delay (up) is 10 msec
  loopback not set,
  Last link flapped 01:38:49
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters 02:34:16
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 7647051000 bits/sec, 12254894 packets/sec
     1061401410 packets input, 82789675614 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 5 broadcast packets, 19429 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     76895885948 packets output, 6192569128048 bytes, 0 total output drops
     Output 7 broadcast packets, 18916 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     2 carrier transitions

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

Assigning Multiple IP Addresses to Network Interfaces

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

- There might not be enough host addresses for a particular network segment. For example, suppose your subnetting allows up to 254 hosts per logical subnet, but on one physical subnet you must have 300 host addresses. Using secondary IP addresses on the routers or access servers allows you to have two logical subnets using one physical subnet.
- Many older networks were built using Level 2 bridges, and were not subnetted. The judicious use of secondary addresses can aid in the transition to a subnetted, router-based network. Routers on an older, bridged segment can easily be made aware that many subnets are on that segment.
- Two subnets of a single network might otherwise be separated by another network. You can create a single network from subnets that are physically separated by another network by using a secondary address. In these instances, the first network is *extended*, or layered on top of the second network. Note that a subnet cannot appear on more than one active interface of the router at a time.

Note If any router on a network segment uses a secondary IPv4 address, all other routers on that same segment must also use a secondary address from the same network or subnet.

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

Configuration Example

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

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

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

Running Configuration

```
Router#show running-config interface HundredGigE 0/0/0/24
Mon Sep 30 09:37:12.096 UTC
interface HundredGigE0/0/0/24
 ipv4 address 192.0.2.27 255.255.255.0
 ipv4 address 192.0.2.1 255.255.255.0 secondary
!
```

Verification

```
Router#show ipv4 interface HundredGigE 0/0/0/24
Mon Sep 30 09:38:18.262 UTC
HundredGigE0/0/0/24 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.0.2.27/24
  Secondary address 192.0.2.1/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
```

Configuring IPv6 Addressing

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

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

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

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

Larger IPv6 Address Space

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

IPv6 Address Formats

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

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

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

It is common for IPv6 addresses to contain successive hexadecimal fields of zeros. To make IPv6 addresses less cumbersome, two colons (::) can be used to compress successive hexadecimal fields of zeros at the beginning, middle, or end of an IPv6 address. (The colons represent successive hexadecimal fields of zeros.) Table 3: Compressed IPv6 Address Formats, on page 11 lists compressed IPv6 address formats.

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

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

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

Table 3: Compressed IPv6 Address Formats

IPv6 Address Type	Preferred Format	Compressed Format
Unicast	2001:0:0:0:0DB8:800:200C:417A	1080::0DB8:800:200C:417A
Multicast	FF01:0:0:0:0:0:0:101	FF01::101
Loopback	0:0:0:0:0:0:0:1	::1
Unspecified	0:0:0:0:0:0:0:0	::

The loopback address listed in Table 3: Compressed IPv6 Address Formats, on page 11 may be used by a node to send an IPv6 packet to itself. The loopback address in IPv6 functions the same as the loopback address in IPv4 (127.0.0.1).

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

The unspecified address listed in Table 3: Compressed IPv6 Address Formats, on page 11 indicates the absence of an IPv6 address. For example, a newly initialized node on an IPv6 network may use the unspecified address as the source address in its packets until it receives its IPv6 address.

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

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

IPv6 Address Type: Unicast

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

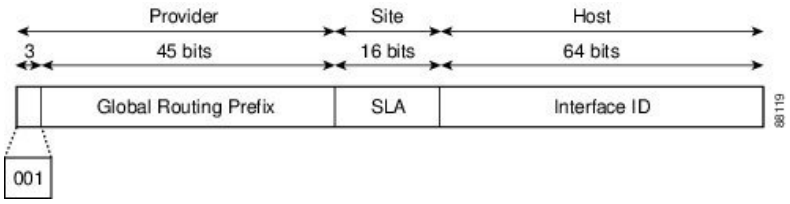
- Global aggregatable address
- Site-local address (proposal to remove by IETF)
- Link-local address
- IPv4-compatible IPv6 address

Aggregatable Global Address

An aggregatable global address is an IPv6 address from the aggregatable global unicast prefix. The structure of aggregatable global unicast addresses enables strict aggregation of routing prefixes that limits the number of routing table entries in the global routing table. Aggregatable global addresses are used on links that are aggregated upward through organizations, and eventually to the Internet service providers (ISPs).

Aggregatable global IPv6 addresses are defined by a global routing prefix, a subnet ID, and an interface ID. Except for addresses that start with binary 000, all global unicast addresses have a 64-bit interface ID. The current global unicast address allocation uses the range of addresses that start with binary value 001 (2000::/3). This figure below shows the structure of an aggregatable global address.

Figure 1: Aggregatable Global Address Format



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

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

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

An interface ID is used to identify interfaces on a link. The interface ID must be unique to the link. It may also be unique over a broader scope. In many cases, an interface ID is the same as or based on the link-layer address of an interface. Interface IDs used in aggregatable global unicast and other IPv6 address types must be 64 bits long and constructed in the modified EUI-64 format.

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

- For all IEEE 802 interface types (for example, Ethernet interfaces and FDDI interfaces), the first three octets (24 bits) are taken from the Organizationally Unique Identifier (OUI) of the 48-bit link-layer address (MAC address) of the interface, the fourth and fifth octets (16 bits) are a fixed hexadecimal value of FFFE, and the last three octets (24 bits) are taken from the last three octets of the MAC address. The construction of the interface ID is completed by setting the Universal/Local (U/L) bit—the seventh bit of the first octet—to a value of 0 or 1. A value of 0 indicates a locally administered identifier; a value of 1 indicates a globally unique IPv6 interface identifier.
- For tunnel interface types that are used with IPv6 overlay tunnels, the interface ID is the IPv4 address assigned to the tunnel interface with all zeros in the high-order 32 bits of the identifier.

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

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

1. The router is queried for MAC addresses (from the pool of MAC addresses in the router).

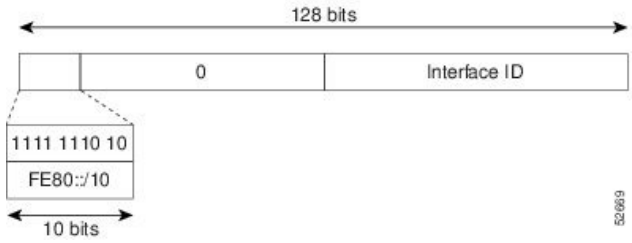
2. If no MAC address is available, the serial number of the Route Processor (RP) or line card (LC) is used to form the link-local address.

Link-Local Address

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

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

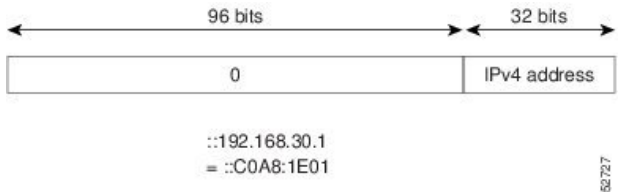
Figure 2: Link-Local Address Format



IPv4-Compatible IPv6 Address

An IPv4-compatible IPv6 address is an IPv6 unicast address that has zeros in the high-order 96 bits of the address and an IPv4 address in the low-order 32 bits of the address. The format of an IPv4-compatible IPv6 address is 0:0:0:0:0:0:A.B.C.D or ::A.B.C.D. The entire 128-bit IPv4-compatible IPv6 address is used as the IPv6 address of a node and the IPv4 address embedded in the low-order 32 bits is used as the IPv4 address of the node. IPv4-compatible IPv6 addresses are assigned to nodes that support both the IPv4 and IPv6 protocol stacks and are used in automatic tunnels. This figure below shows the structure of an IPv4-compatible IPv6 address and a few acceptable formats for the address.

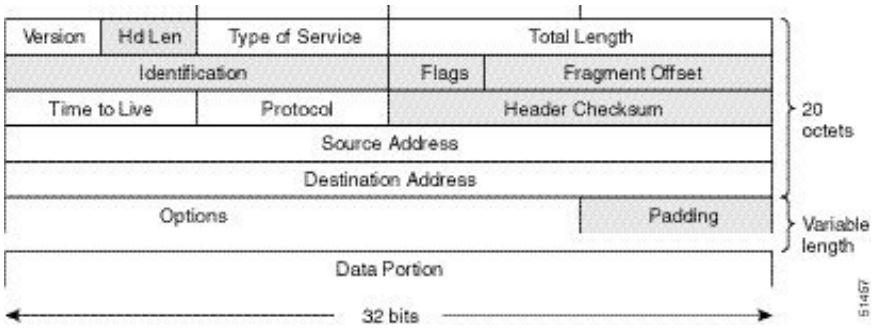
Figure 3: IPv4-Compatible IPv6 Address Format



Simplified IPv6 Packet Header

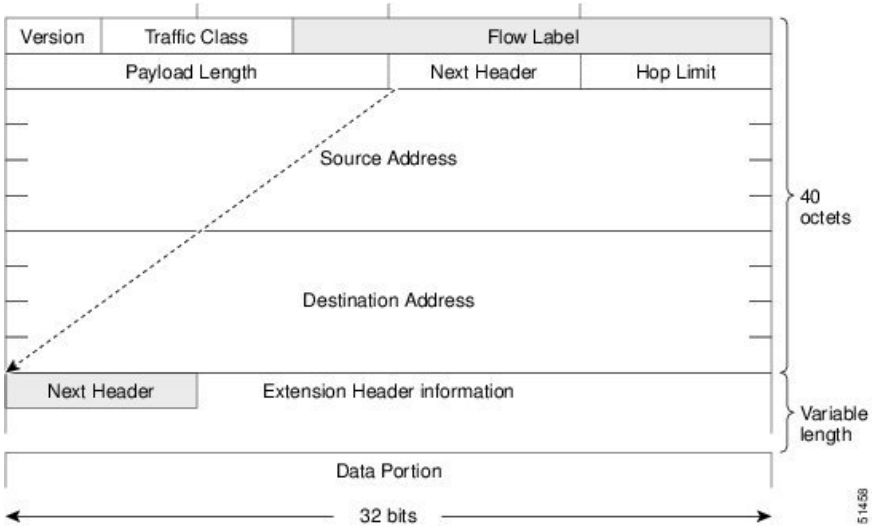
The basic IPv4 packet header has 12 fields with a total size of 20 octets (160 bits). The 12 fields may be followed by an Options field, which is followed by a data portion that is usually the transport-layer packet. The variable length of the Options field adds to the total size of the IPv4 packet header.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

Figure 4: IPv4 Packet Header Format



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

Figure 5: IPv6 Packet Header Format



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

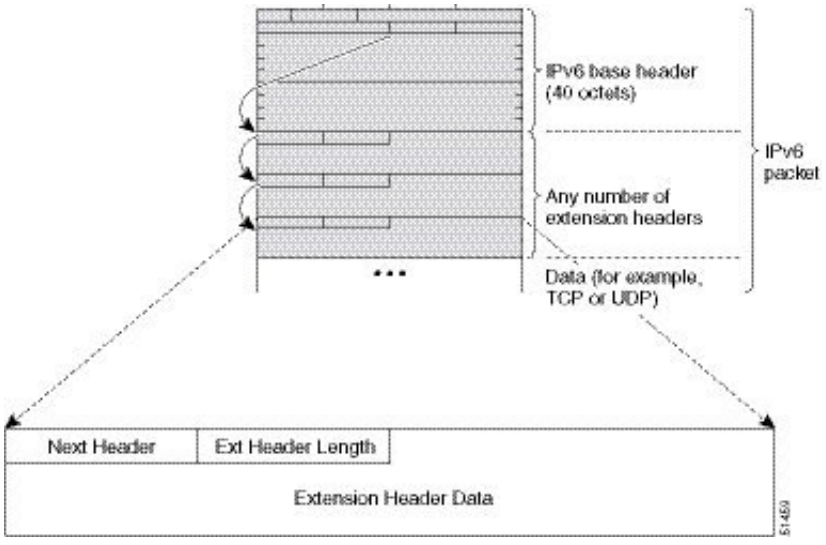
Table 4: Basic IPv6 Packet Header Fields

Field	Description
Version	Similar to the Version field in the IPv4 packet header, except that the field lists number 6 for IPv6 instead of number 4 for IPv4.
Traffic Class	Similar to the Type of Service field in the IPv4 packet header. The Traffic Class field tags packets with a traffic class that is used in differentiated services.

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

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

Figure 6: IPv6 Extension Header Format



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

Table 5: IPv6 Extension Header Types

Header Type	Next Header Value	Description
Hop-by-hop options header	0	This header is processed by all hops in the path of a packet. When present, the hop-by-hop options header always follows immediately after the basic IPv6 packet header.
Destination options header	60	The destination options header can follow any hop-by-hop options header, in which case the destination options header is processed at the final destination and also at each visited address specified by a routing header. Alternatively, the destination options header can follow any Encapsulating Security Payload (ESP) header, in which case the destination options header is processed only at the final destination.
Routing header	43	The routing header is used for source routing.
Fragment header	44	The fragment header is used when a source must fragment a packet that is larger than the maximum transmission unit (MTU) for the path between itself and a destination. The Fragment header is used in each fragmented packet.
Authentication header and ESP header	51 50	The Authentication header and the ESP header are used within IP Security Protocol (IPSec) to provide authentication, integrity, and confidentiality of a packet. These headers are identical for both IPv4 and IPv6.
Upper-layer header	6 (TCP) 17 (UDP)	The upper-layer (transport) headers are the typical headers used inside a packet to transport the data. The two main transport protocols are TCP and UDP.
Mobility header	To be done by IANA	Extension headers used by mobile nodes, correspondent nodes, and home agents in all messaging related to the creation and management of bindings.

IPv6 Multicast Groups

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

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

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

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

Configuration Example

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

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

Running Configuration

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

Verification

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

```
Router#show ipv6 interface HundredGigE 0/0/0/25
HundredGigE0/0/0/25 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::7ae7:abff:febd:d4c4
  Global unicast address(es):
    2001:db8:0:1::1, subnet is 2001:db8:0:1::/64
    2001:db8:0:1:7ae7:abff:febd:d4c4, subnet is 2001:db8:0:1::/64
  Joined group address(es): ff02::1:ff00:1 ff02::1:ffbd:d4c4 ff02::2
      ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 160 to 240 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Configuration Example Using eui-64Keyword

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

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

Running Configuration

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

Verification

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

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

Mon Sep 30 09:06:12.638 UTC
HundredGigE0/0/0/35 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::7ae7:abff:febd:d4ec
  Global unicast address(es):
    2001:db8:0:1:7ae7:abff:febd:d4ec, subnet is 2001:db8:0:1::/64
  Joined group address(es): ff02::1:ffbd:d4ec ff02::2 ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 160 to 240 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Configuration Example Using link-local Address

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

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

Running Configuration

```
Router# show running-config interface HundredGigE 0/0/0/34
Mon Sep 30 09:11:09.742 UTC
```

```
interface HundredGigE0/0/0/34
 ipv6 address fe80::260:3eff:fe11:6770 link-local
!
```

Verification

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

```
Router#show ipv6 interface HundredGigE 0/0/0/34
 HundredGigE0/0/0/34 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::260:3eff:fe11:6770
  No global unicast address is configured
  Joined group address(es): ff02::1:ff11:6770 ff02::2 ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 160 to 240 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Enable IPv6

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

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

Running Configuration

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

Verification

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

```
Router#show ipv6 interface HundredGigE 0/0/0/33
Mon Sep 30 09:14:48.430 UTC
HundredGigE0/0/0/33 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::7ae7:abff:febd:d4e4
  No global unicast address is configured
  Joined group address(es): ff02::1:ffbd:d4e4 ff02::2 ff02::1
```

```
MTU is 1514 (1500 is available to IPv6)
ICMP redirects are disabled
ICMP unreachables are enabled
ND DAD is enabled, number of DAD attempts 1
ND reachable time is 0 milliseconds
ND cache entry limit is 1000000000
ND advertised retransmit interval is 0 milliseconds
ND router advertisements are sent every 160 to 240 seconds
ND router advertisements live for 1800 seconds
Hosts use stateless autoconfig for addresses.
Outgoing access list is not set
Inbound  common access list is not set, access list is not set
Table Id is 0xe0800000
Complete protocol adjacency: 0
Complete glean adjacency: 0
Incomplete protocol adjacency: 0
Incomplete glean adjacency: 0
Dropped protocol request: 0
Dropped glean request: 0
```

IPv6 Virtual Addresses

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

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

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

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

Configuration Example

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

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

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

```
Router#configure terminal
```

```
Router(config)#ipv6 virtual address 2001:db8:0:1::1/64
Router(config)#interface mgmtEth 0/RP0/CPU0/0
Router(config-if)#ipv6 address 2001:0DB8:0:1::1/64
Router(config-if)#no shutdown
Router(config-if)#commit
Router(config-if)#exit
```

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

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

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

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

IPv6 ICMP Rate Limiting

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

Configuration Example

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

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

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

Running Configuration

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

Associated Commands

- ipv6 icmp error-interval

Configuring IPv4 and IPv6 Protocol Stacks

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

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

Configuration Example

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

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

Running Configuration

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

Verification

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

```
Router#show ipv4 interface HundredGigE 0/0/0/31
Mon Sep 30 09:49:10.383 UTC
HundredGigE0/0/0/31 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.0.2.1/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000

Router#show ipv6 interface HundredGigE 0/0/0/31
 Mon Sep 30 09:51:01.472 UTC
HundredGigE0/0/0/31 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::7ae7:abff:febd:d4dc
  Global unicast address(es):
    2001:db8:c18:1::3, subnet is 2001:db8:c18:1::/64
  Joined group address(es): ff02::1:ff00:3 ff02::1:ffbd:d4dc ff02::2
      ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 160 to 240 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Selecting Flexible Source IP

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

Configuration Example

Enables RFC compliance for source address selection.

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

Running Configuration

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

Associated Commands

- icmp ipv4 source vrf

Path MTU Discovery for IPv6

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

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

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

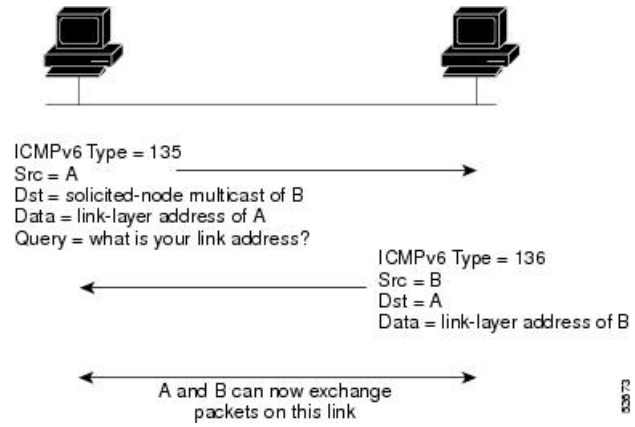
IPv6 Neighbor Discovery

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

IPv6 Neighbor Solicitation Message

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

Figure 7: IPv6 Neighbor Discovery—Neighbor Solicitation Message



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

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

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

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

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

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

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

source to the destination node; solicited neighbor advertisement messages indicate that a path is working in both directions.

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

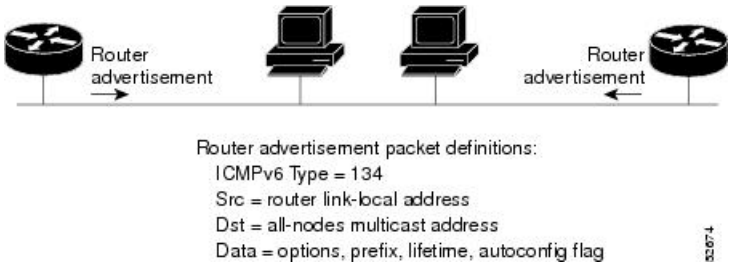
Neighbor solicitation messages are also used in the stateless autoconfiguration process to verify the uniqueness of unicast IPv6 addresses before the addresses are assigned to an interface. Duplicate address detection is performed first on a new, link-local IPv6 address before the address is assigned to an interface. (The new address remains in a tentative state while duplicate address detection is performed.) Specifically, a node sends a neighbor solicitation message with an unspecified source address and a tentative link-local address in the body of the message. If another node is already using that address, the node returns a neighbor advertisement message that contains the tentative link-local address. If another node is simultaneously verifying the uniqueness of the same address, that node also returns a neighbor solicitation message. If no neighbor advertisement messages are received in response to the neighbor solicitation message and no neighbor solicitation messages are received from other nodes that are attempting to verify the same tentative address, the node that sent the original neighbor solicitation message considers the tentative link-local address to be unique and assigns the address to the interface.

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

IPv6 Router Advertisement Message

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

Figure 8: IPv6 Neighbor Discovery—Router Advertisement Message



Router advertisement messages typically include the following information:

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

- Additional information for hosts, such as the hop limit and MTU a host should use in packets that it originates

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

The following router advertisement message parameters can be configured:

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

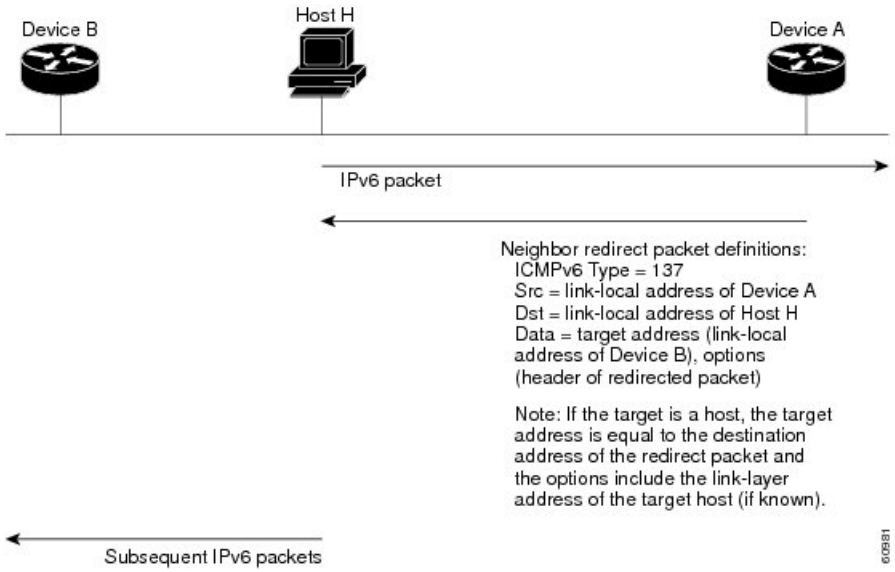
The configured parameters are specific to an interface. The sending of router advertisement messages (with default values) is automatically enabled on Ethernet and FDDI interfaces. For other interface types, the sending of router advertisement messages must be manually configured by using the **no ipv6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

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

IPv6 Neighbor Redirect Message

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

Figure 9: IPv6 Neighbor Discovery—Neighbor Redirect Message



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

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

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

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

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

Configuring IPARM Conflict Resolution

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

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

Static Policy Resolution

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

Configuration Example

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

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

Running Configuration

```
Router#show running-config | in ipv4 config
Building configuration...
!! IOS XR Configuration version = 6.0.0.26I
!! Last configuration change at Mon Dec 14 21:57:27 2015 by root
!
hostname sample-83
logging console debugging
username root
 group root-lr
 group test
 secret 5 $1$d2NC$RbAdqdU7kw/eKJpMo/GJI1
!
cdp
ipv4 conflict-policy static
interface Loopback0
 ipv4 address 192.0.2.1 255.255.255.0
!
…..
```

Verification

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

Longest Prefix Address Conflict Resolution

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

Configuration Example

Configures longest prefix address conflict resolution.

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

Running Configuration

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

Verification

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

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

Highest IP Address Conflict Resolution

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

Configuration

Configures highest IP address conflict resolution.

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

Running Configuration

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

Verification

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

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

Forced down interface         Up interface                   VRF
```

Route-Tag Support for Connected Routes

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

Configuration Example

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

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

Running Configuration

```
Router#show running-config interface HundredGigE 0/0/0/24
interface HundredGigE0/0/0/24
 ipv4 address 192.0.2.27 255.255.255.0 route-tag 20
 shutdown
!
```

Verification

Verify the parameters of the route.

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

Address Repository Manager

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

Address Conflict Resolution

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

Conflict Database

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

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

address 1: 10.1.1.1/16

address 2: 10.2.1.1/16

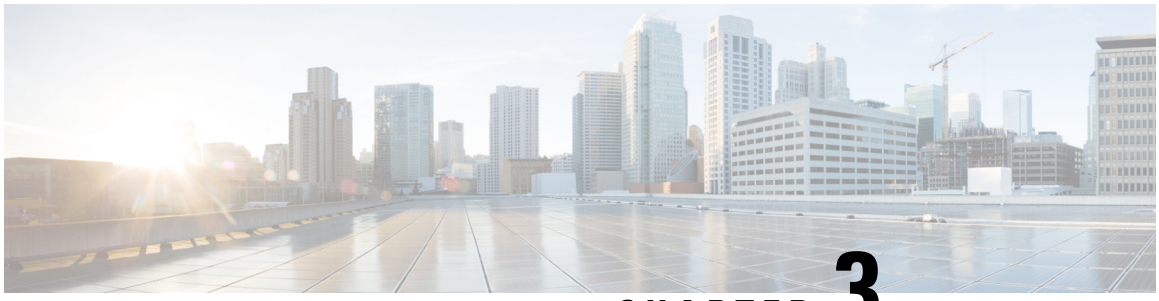
address 3: 10.3.1.1/16

address 4: 10.4.1.1/8

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

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

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



CHAPTER 3

Configuring ARP

Configuring ARP

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

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

ARP and Proxy ARP

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

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

Address Resolution Overview

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

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

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

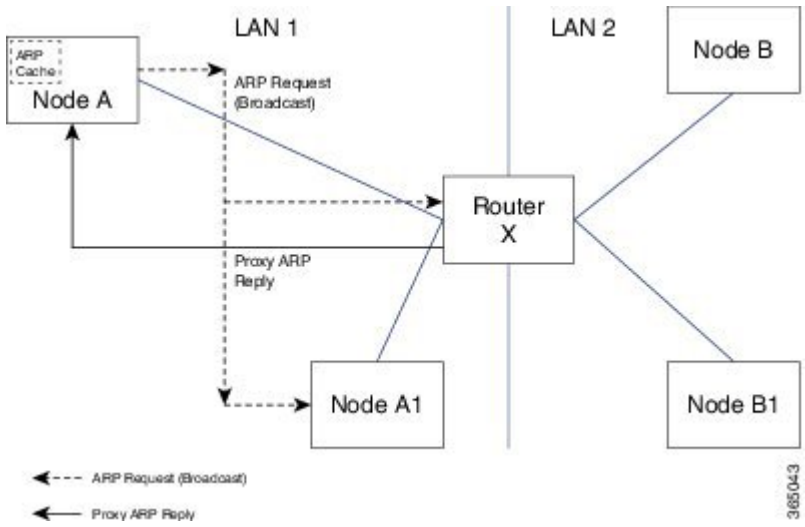
Address Resolution on a Single LAN

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

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

Address Resolution When Interconnected by a Router

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



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

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

ARP Cache Entries

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

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

Defining a Static ARP Cache Entry

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

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

Configuration Example

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

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

Running Configuration

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

Verification

Verify that the State is static for proper functioning:

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

Proxy ARP and Local Proxy ARP

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

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

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

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

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

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

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

Enabling Proxy ARP

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

Configuration Example

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

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

Running Configuration

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

```
!
!
```

Verification

Verify that proxy ARP is configured and enabled:

```
Router#show arp idb HundredGigE 0/0/0/0 location 0/0/CPU0
  interface HundredGigE0/0/0/0 (0x08000038):
  IPv4 address 192.0.2.1, Vrf ID 0x60000000
  VRF Name default
  Dynamic learning: Enable
  Dynamic entry timeout: 14400 secs
  Purge delay: off
  IPv4 caps added (state up)
  MPLS caps not added
  Interface not virtual, not client fwd ref,
  Proxy arp is configured, is enabled
  Local Proxy arp not configured
  Packet IO layer is NetIO
  Srg Role : DEFAULT
  Idb Flag : 262332
  IDB is Complete
```

Enabling Local Proxy ARP

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

Configuration Example

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

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

Running Configuration

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

Verification

Verify that local proxy ARP is configured:

```
Router#show arp idb HundredGigE0/0/0/0 location 0/0/CPU0
HundredGigE0/0/0/0(0x08000038):
  IPv4 address 192.0.2.1, Vrf ID 0x60000000
  VRF Name default
  Dynamic learning: Enable
  Dynamic entry timeout: 14400 secs
  Purge delay: off
  IPv4 caps added (state up)
  MPLS caps not added
  Interface not virtual, not client fwd ref,
```

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

Associated Commands

- local-proxy-arp
- show arp idb

Configuring ARP purge-delay

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

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

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

Configuring Example

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

Running Configuration

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

Verification

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

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

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

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

Configuring ARP Timeout

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

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

Configuring Example

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

Running Configuration

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

Verification

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

HundredGigE0/0/0/35 (0x0f000200):
  IDB Client: default
  IPv4 address 192.0.2.1, Vrf ID 0x00000000
  VRF Name unknown
  Dynamic learning: Enable
  Dynamic entry timeout: 100 secs
  Drop adjacency timeout: Disable
  Purge delay: off
  IPv4 caps not added
  MPLS caps not added
  Interface not virtual, not client fwd ref,
  Proxy arp not configured, not enabled
  Local Proxy arp not configured
  Packet IO layer is NetIO
  Srg Role : DEFAULT
  Idb Flag : 0
  IDB is not Complete
```

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

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

Direct Attached Gateway Redundancy

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

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

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

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

Restrictions

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

- IPv6 is not supported.
- Ethernet bundles are not supported.
- Non-Ethernet interfaces are not supported.
- Hitless ARP Process Restart is not supported.
- Hitless RSP Failover is not supported.

Configuring DAGR

Configuration Example

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

```
Router(config-if-dagr)# peer ipv4 192.0.2.1
Router(config-if-dagr-peer)# route distance normal 140 priority 3
Router(config-if-dagr-peer)# route metric normal 84 priority 80
Router(config-if-dagr-peer)# timers query 2 standby 19
Router(config-if-dagr-peer)# priority-timeout 25
Router(config-if-dagr)# commit
```

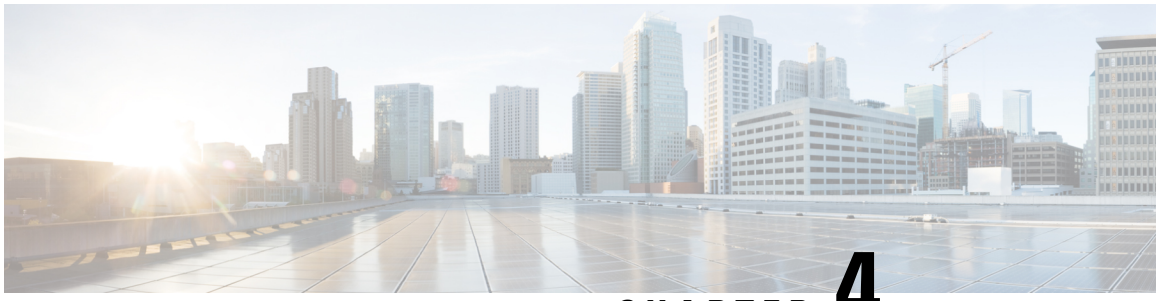
Running Configuration

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

Verification

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

```
Router# show arp dagr
------------------------------------------------------------------------------
0/1/CPU0
------------------------------------------------------------------------------
Interface                 Virtual IP      State     Query-pd Dist Metr
HundredGigE0/0/0/1     192.0.2.1    Active    None     150  100
HundredGigE0/0/0/1     192.0.2.45     Query     1        None None
```



CHAPTER 4

Implementing Host Services and Applications

Implementing Host Services and Applications

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

Prerequisites for implementing Host Services and Applications

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

Network Connectivity Tools

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

Ping

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

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

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

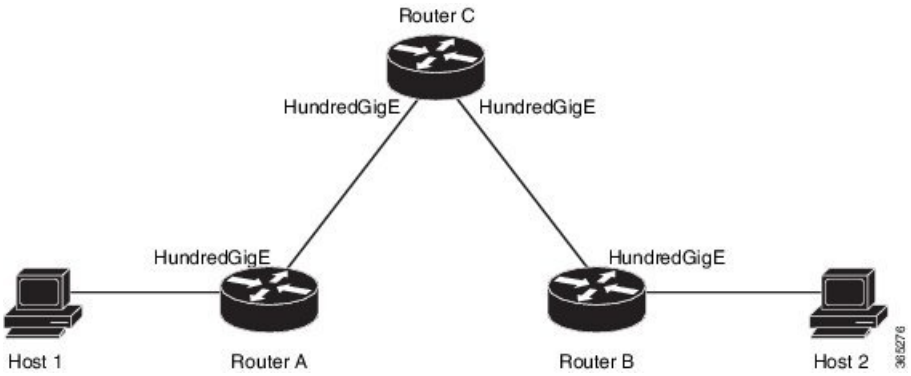
Checking Network Connectivity

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

Configuration for Checking Network Connectivity

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

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



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

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

Configuration Example

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

```
Router# ping 192.0.2.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.0.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5)
Router#!!!!!

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

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

Router# ping
Tue Sep 24 02:41:45.739 UTC
Protocol [ipv4]: ipv4
Target IP address: 192.0.2.1
Repeat count [5]: 5
Datagram size [100]: 1
% A decimal number between 36 and 18024.
Datagram size [100]: 36
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 1
Extended commands? [no]: y
Source address or interface: 12.12.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes? [no]:
Type escape sequence to abort.
Sending 5, 36-byte ICMP Echos to 192.0.2.1, timeout is 1 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/7 ms
```

Associated Commands

- ping

Checking Network Connectivity for Multiple Destinations

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

Configuration Example

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

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

```
Router# ping bulk ipv4 input cli batch
*/You must hit the Enter button and then specify one destination address per line*/
Please enter input via CLI with one destination per line and when done Ctrl-D/(exit) to
initiate pings:
1: 192.0.2.1
2: 198.51.100.1
```

```
3: 203.0.113.1
4:
Starting pings...
Target IP address: 192.0.2.1
Repeat count [5]: 5
Datagram size [100]: 1
% A decimal number between 36 and 18024.
Datagram size [100]: 1
% A decimal number between 36 and 18024.
Datagram size [100]: 1000
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 10
Extended commands? [no]: no
Sweep range of sizes? [no]: q
% Please answer 'yes' or 'no'.
Sweep range of sizes? [no]: q
% Please answer 'yes' or 'no'.
Sweep range of sizes? [no]:
Type escape sequence to abort.
Sending 5, 1000-byte ICMP Echos to 192.0.2.1, vrf is default, timeout is 1 seconds:
!!!!!
Success rate is 100 percent (5/5),
Target IP address: 198.51.100.1
Repeat count [5]:
Datagram size [100]: q
% A decimal number between 36 and 18024.
Datagram size [100]:
Timeout in seconds [2]:
Interval in milliseconds [10]:
Extended commands? [no]:
Sweep range of sizes? [no]:
Sending 5, 100-byte ICMP Echos to 192.0.2.1, vrf is default, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5),
Target IP address:  203.0.113.1
Repeat count [5]: 4
Datagram size [100]: 100
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 10
Extended commands? [no]: no
Sweep range of sizes? [no]: no
Sending 4, 100-byte ICMP Echos to 192.0.2.1, vrf is default, timeout is 1 seconds:
!!!!!
Success rate is 100 percent (4/5),
```

Associated Commands

- ping bulk ipv4

Traceroute

Where the **ping** command can be used to verify connectivity between devices, the **traceroute** command can be used to discover the paths packets take to a remote destination and where routing breaks down.

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

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

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

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

Checking Packet Routes

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

Configuration Example

Trace the route from 192.0.2.1 to 198.51.100.1:

```
Router# traceroute 198.51.100.1
Type escape sequence to abort.
Tracing the route to 198.51.100.1
 1  192.0.2.1 39 msec  *  3 msec

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

Router #traceroute
Protocol [ipv4]:
Target IP address: 198.51.100.1
Source address: 192.0.2.1
Numeric display? [no]:
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:

Type escape sequence to abort.
Tracing the route to 198.51.100.1
 1  192.0.2.1.1 3 msec  *  3 msec
```

Associated Commands

- traceroute

Domain Services

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

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

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

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

Configuring Domain Services

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

Configuration Example

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

```
Defining the Domain Host
================================
Router# configure
Router(config)#domain ipv4 host host1 192.168.7.18
Router(config)#domain ipv4 host host2 10.2.0.2 192.168.7.33
Router(config)#commit

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

Specifying the Addresses of the Name Servers
============================================
*/Specify host 192.168.1.111 as the primary name server
and host 192.168.1.2 as the secondary server/*
Router#configure
Router(config)#domain name-server 192.168.1.111
```

```
Router(config)#domain name-server 192.168.1.2
Router(config)#commit
```

Verification

```
Router#show hosts
Default domain is cisco.com
Name/address lookup uses domain service
Name servers: 192.168.1.111, 192.168.1.2

Host                      Flags     Age(hr) Type Address(es)
host2                   (perm, OK)    0      IP  10.2.0.2
                                                 192.168.7.33
host1                   (perm, OK)    0      IP  192.168.7.18
```

Associated Commands

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

File Transfer Services

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

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

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

FTP

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

Configuring a Router to Use FTP Connections

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

- Passive-mode FTP
- Password

- IP address

Configuration Example

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

```
Router#configure
Router(config)#ftp client passive

Router(config)#ftp client anonymous-password xxxx
Router(config)#ftp client source-interface HundredGigE 0/0/0/0
Router(config)#commit
```

Running Configuration

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

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

Associated Commands

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

TFTP

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

Configuring a Router to Use TFTP Connections

Configuration Example

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

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

Running Configuration

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

Verification

```
Router#show cinetd services
Vrf Name Family  Service  Proto Port ACL max_cnt  curr_cnt  wait   Program Client Option

default  v4    tftp udp    69      unlimited  0       wait   tftpd    sysdb disk0:

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

Associated Commands

- tftp client source-interface type
- show cinetd services

SCP

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

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

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

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

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

Transferring Files Using SCP

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

Configuration Example

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

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

Verification

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

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

Associated Commands

- scp

Cisco inetd

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

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

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

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

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

Telnet

Enabling Telnet allows inbound Telnet connections into a networking device.

Prerequisites

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

Configuration Example

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

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

Verification

```
Router# show cinetd services
Vrf Name  Family    Service   Proto Port ACL max_cnt  curr_cnt  wait   Program Client Option

default   v4         tftp      udp  69       unlimited  0        wait   tftpd   sysdb
disk0:
default   v4     telnet  tcp  23        10  0        nowait telnetd sysdb
```

Syslog source-interface

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

Configuration Example

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

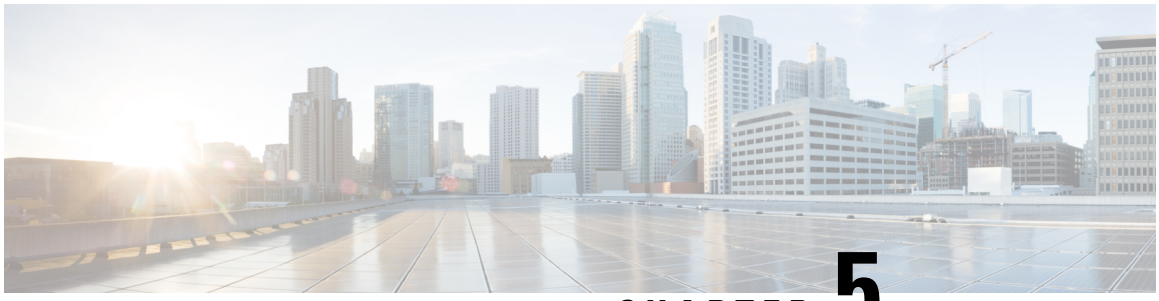
```
Router#configure
Router(config)#logging source-interface Loopback2

Router(config)#commit
```

Running Configuration

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

Associated Commands

- logging source-interface
- show running-configuration logging



CHAPTER 5

Implementing Access Lists

Understanding Access Lists

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

An access control list (ACL) consists of one or more access control entries (ACE) that collectively define the network traffic profile. Access control entries (ACE) are entries in an ACL that describe the access rights related to a particular security identifier or user. This profile can then be referenced by Cisco IOS XR software features such as traffic filtering, route filtering, QoS classification, and access control. There are 2 types of ACLs:

- Standard ACLs- Verifies only the source IP address of the packets. Traffic is controlled by the comparison of the address or prefix configured in the ACL, with the source address found in the packet.
- Extended ACLs- Verifies more than just the source address of the packets. Attributes such as destination address, specific IP protocols, User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) port numbers, Differentiated Services Code Point (DSCP), and so on are validated. Traffic is controlled by a comparison of the attributes stated in the ACL with those in the incoming or outgoing packets.

Cisco IOS XR does not differentiate between standard and extended access lists. Standard access list support is provided for backward compatibility.

Purpose of IP Access Lists

- Filter incoming or outgoing packets on an interface.
- Filter packets for mirroring.
- Redirect traffic as required.
- Restrict the contents of routing updates.
- Limit debug output based on an address or protocol.
- Control vty access.
- Identify or classify traffic for advanced features, such as congestion avoidance, congestion management, and priority and custom queueing.

How an IP Access List Works

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

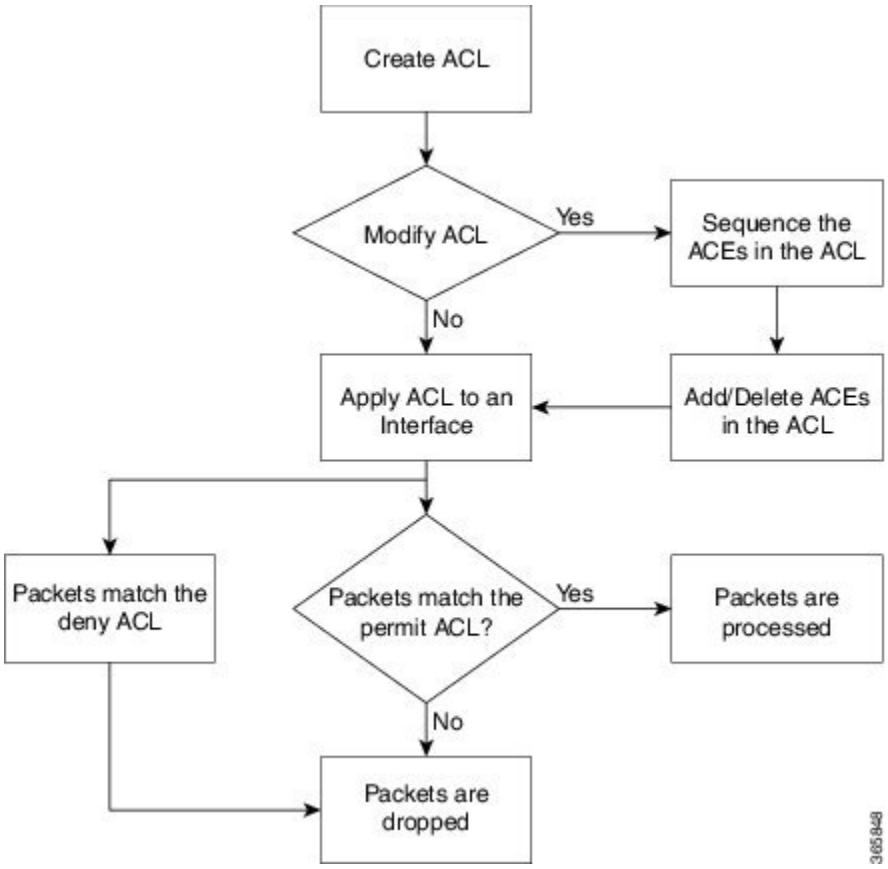
An access list can be configured and named, but it is not in effect until the access list is referenced by a command that accepts an access list. Multiple commands can reference the same access list. An access list can control traffic arriving at the router or leaving the router, but not traffic originating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

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

You can also filter packets on the basis of transport layer information, such as whether the packet is a TCP, UDP, Internet Control Message Protocol (ICMP), or Internet Group Management Protocol (IGMP) packet.

ACL Workflow

The following image illustrates the workflow of an ACL.



IP Access List Process and Rules

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

- The software tests the source or destination address or the protocol of each packet being filtered against the conditions in the access list, one condition (permit or deny statement) at a time.
- If a packet does not match an access list statement, the packet is then tested against the next statement in the list.
- If a packet and an access list statement match, the remaining statements in the list are skipped and the packet is permitted or denied as specified in the matched statement. The first entry that the packet matches determines whether the software permits or denies the packet. That is, after the first match, no subsequent entries are considered.
- If the access list denies the address or protocol, the software discards the packet and returns an ICMP Host Unreachable message. ICMP is configurable in the Cisco IOS XR software.

  ICMP type and code such as ECHO, ECHO-REPLY, MASK-REPLY, MASK-REQUEST, and so on are supported.
- If no conditions match, the software drops the packet because each access list ends with an unwritten or implicit deny statement. That is, if the packet has not been permitted or denied by the time it was tested against each statement, it is denied.
- The access list should contain at least one permit statement or else all packets are denied.

- Because the software stops testing conditions after the first match, the order of the conditions is critical. The same permit or deny statements specified in a different order could result in a packet being passed under one circumstance and denied in another circumstance.
- Only one access list per interface, per protocol, per direction is allowed.
- Inbound access lists process packets arriving at the router. Incoming packets are processed before being routed to an outbound interface. An inbound access list is efficient because it saves the overhead of routing lookups if the packet is to be discarded because it is denied by the filtering tests. If the packet is permitted by the tests, it is then processed for routing. For inbound lists, permit means continue to process the packet after receiving it on an inbound interface; **deny** means discard the packet.
- Outbound access lists process packets before they leave the router. Incoming packets are routed to the outbound interface and then processed through the outbound access list. For outbound lists, permit means send it to the output buffer; deny means discard the packet.
- An access list can not be removed if that access list is being applied by an access group in use. To remove an access list, remove the access group that is referencing the access list and then remove the access list.
- Before removing an interface, which is configured with an ACL that denies certain traffic, you must remove the ACL and commit your configuration. If this is not done, then some packets are leaked through the interface as soon as the **no interface <interface-name>** command is configured and committed.
- An access list must exist before you can use the **ipv4 | ipv6 access-group** command.

Helpful Hints for Creating IP Access Lists

Consider the following when creating an IP access list:

- Create the access list before applying it to an interface.
- Organize your access list so that more specific references in a network or subnet appear before more general ones.
- To make the purpose of individual statements more easily understood at a glance, you can write a helpful remark before or after any statement.

Source and Destination Addresses

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

ACL Filtering by Wildcard Mask and Implicit Wildcard Mask

Address filtering uses wildcard masking to indicate whether the software checks or ignores corresponding IP address bits when comparing the address bits in an access-list entry to a packet being submitted to the access list. By carefully setting wildcard masks, an administrator can select a single or several IP addresses for permit or deny tests.

Wildcard masking for IP address bits uses the number 1 and the number 0 to specify how the software treats the corresponding IP address bits. A wildcard mask is sometimes referred to as an *inverted mask*, because a 1 and 0 mean the opposite of what they mean in a subnet (network) mask.

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

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

You do not have to supply a wildcard mask with a source or destination address in an access list statement. If you use the **host** keyword, the software assumes a wildcard mask of 0.0.0.0.

Unlike subnet masks, which require contiguous bits indicating network and subnet to be ones, wildcard masks allow noncontiguous bits in the mask.

You can also use Classless Inter-Domain Routing (CIDR) format (/x) in place of wildcard bits. For example, the IPv4 address 1.2.3.4 0.255.255.255 corresponds to 1.2.3.4/8 and for IPv6 address 2001:db8:abcd:0012:0000:0000:0000:0000 corresponds to 2001:db8:abcd:0012::0/64.

Transport Layer Information

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

Restrictions for Configuring Access Lists

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

- IPv4 and IPv6 ACLs are not supported for loopback and interflex interfaces.
- If the Ternary content-addressable memory (TCAM) utilization is high and large ACLs are modified, then an error may occur. During such instances, remove the ACL from the interface and reconfigure the ACL. Later, reapply the ACL to the interface.
- Filtering of Multiprotocol Label Switching (MPLS) packets through interface ACL is not supported.
- Modifying an ACL when it is attached to the interface is supported.
- You can configure an ACL name with a maximum of 64 characters.
- You can configure an ACL name to comprise of only letters and numbers.

Including Comments in Access Lists

You can include comments (remarks) about entries in any named IP access list using the remark access list configuration command. The remarks make the access list easier for the network administrator to understand and scan. Each remark line is limited to 255 characters.

The remark can go before or after a **permit** or **deny** statement. You should be consistent about where you put the remark so it is clear which remark describes which **permit** or **deny** statement. For example, it would be confusing to have some remarks before the associated **permit** or **deny** statements and some remarks after the associated statements. Remarks can be sequenced.

Remember to apply the access list to an interface or terminal line after the access list is created.

IP Access List Entry Sequence Numbering

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

then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

The IP Access List Entry Sequence Numbering feature allows users to add sequence numbers to access-list entries and resequence them. When you add a new entry, you choose the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

Sequence Numbering Behavior

The following details the sequence numbering behavior:

- If entries with no sequence numbers are applied, the first entry is assigned a sequence number of 10, and successive entries are incremented by 10. The maximum configurable sequence number is 2147483643 for IPv4 and IPv6 entries. For other entries, the maximum configurable sequence number is 2147483646. If the generated sequence number exceeds this maximum number, the following message displays:

```
Exceeded maximum sequence number.
```

- If you provide an entry without a sequence number, it is assigned a sequence number that is 10 greater than the last sequence number in that access list and is placed at the end of the list.
- ACL entries can be added without affecting traffic flow and hardware performance.
- If a new access list is entered from global configuration mode, then sequence numbers for that access list are generated automatically.
- Distributed support is provided so that the sequence numbers of entries in the route processor (RP) and line card (LC) are synchronized at all times.
- This feature works with named standard and extended IP access lists. Because the name of an access list can be designated as a number, numbers are acceptable.

Adding Entries with Sequence Numbers: Example

In the following example, an new entry is added to IPv4 access list acl_5.

```
ipv4 access-list acl_5
 2 permit ipv4 host 192.0.2.1 any
 5 permit ipv4 host 198.51.100.44 any
 10 permit ipv4 host 198.51.100.1 any
 20 permit ipv4 host 198.51.100.2 any
configure
ipv4 access-list acl_5
 15 permit 203.0.113.1 255.255.255.0
end
ipv4 access-list acl_5
 2 permit ipv4 host 192.0.2.1 any
 5 permit ipv4 host 198.51.100.44 any
 10 permit ipv4 host 198.51.100.1 any
 15 permit ipv4 203.0.113.1 255.255.255.0 any
 20 permit ipv4 host 198.51.100.2 any
```

Adding Entries Without Sequence Numbers: Example

The following example shows how an entry with no specified sequence number is added to the end of an access list. When an entry is added without a sequence number, it is automatically given a sequence number that puts it at the end of the access list. Because the default increment is 10, the entry will have a sequence number 10 higher than the last entry in the existing access list.

```
configure
ipv4 access-list acl_10
permit 192.0.2.1 255.255.255.0
permit 198.51.100.1 255.255.255.0
permit 203.0.113.1 255.255.255.0
end

ipv4 access-list acl_10
 10 permit ip  192.0.2.1 255.255.255.0 any
 20 permit ip  198.51.100.1 255.255.255.0 any
 30 permit ip  203.0.113.1 255.255.255.0 any

configure
ipv4 access-list acl_10
permit  203.0.113.5 255.255.255.0
end

ipv4 access-list acl_10
 10 permit ip  192.0.2.1 255.255.255.0 any
 20 permit ip  198.51.100.1 255.255.255.0 any
 30 permit ip  203.0.113.1 255.255.255.0 any
 40 permit ip  203.0.113.5 255.255.255.0 any
```

Applying Access Lists

After you create an access list, you must reference the access list to make it work. Access lists can be applied on *either* outbound or inbound interfaces. This section describes guidelines on how to accomplish this task for both terminal lines and network interfaces.

Set identical restrictions on all the virtual terminal lines, because a user can attempt to connect to any of them.

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

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

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

Configuring IPv4 ACLs

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

Notes and Restrictions for Configuring IPv4 Ingress ACLs

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

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

Notes and Restrictions for Configuring IPv4 Egress ACLs

IPv4 egress ACLs are characterized by the following behavior.

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

Configuring an Ingress IPv4 ACL on a HundredGigE Interface

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

```
/* Configure a HundredGigE interface with an IPv4 address */
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)# no shut
Router(config-if)# commit
Thu Jul 11 08:46:51.930 UTC
Router(config-if)# exit

/* Verify if the interface is up */
Router(config)# do show ipv4 interface brief

Thu Jul 11 08:46:51.930 UTC
Interface                  IP-Address      Status          Protocol Vrf-Name
HundredGigE 0/0/0/0         192.0.2.1        Up              Up       default

/* Configure an IPv4 ingress ACL */
Router(config)# ipv4 access-list V4-ACL-INGRESS
Router(config-ipv4-acl)# 10 permit tcp 192.0.2.2 255.255.255.0 any
Router(config-ipv4-acl)# 20 deny udp any any
Router(config-ipv4-acl)# 30 permit ipv4 192.0.2.64 255.255.255.0 any
Router(config-ipv4-acl)# commit
Thu Jul 11 08:55:12.806 UTC

/* Verify the ingress ACL creation */
Router(config)# do show access-lists ipv4
Thu Jul 11 08:55:44.824 UTC
...
ipv4 access-list V4-ACL-INGRESS
```

```
 10 permit tcp 192.0.2.2 255.255.255.0 any
 20 deny udp any any
 30 permit ipv4 192.0.2.64 255.255.255.0 any


/* Apply the ingress ACL to the HundredGigE interface */
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv4 access-group V4-ACL-INGRESS ingress
Router(config-if)# commit
Thu Jul 11 09:01:26.744 UTC
Router(config-if)# exit

/* Verify if the ingress ACL has been successfully applied to the interface */
Router(config)# do show ipv4 interface
Thu Jul 11 09:01:50.445 UTC
HundredGigE 0/0/0/0 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.0.2.1/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  common access list is not set, access list is V4-ACL-INGRESS
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
```

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

Configuring an Egress IPv4 ACL on a HundredGigE Interface

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

```
/* Configure a HundredGigE interface with an IPv4 address */
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv4 address 198.51.100.1 255.255.255.0
Router(config-if)# no shut
Router(config-if)# commit
Thu Jul 11 08:55:12.806 UTC
Router(config-if)# exit

/* Verify if the interface is up */
Router(config)# do show ipv4 interface brief
Thu Jul 11 08:55:44.824 UTC

Interface                      IP-Address      Status          Protocol Vrf-Name
HundredGigE 0/0/0/0         192.0.2.1          Up              Up       default
HundredGigE 0/0/0/1         198.51.100.1       Up              Up       default

/* Configure an IPv4 egress ACL */
Router(config)# ipv4 access-list V4-ACL-EGRESS
Router(config-ipv4-acl)# 10 permit ipv4 203.0.113.1 255.255.255.0 192.0.2.1 0.255.255.255
Router(config-ipv4-acl)# 20 deny ipv4 any any
Router(config-ipv4-acl)# commit
Thu Jul 11 08:59:10.093 UTC

/* Verify the egress ACL creation */
Router(config)# do show access-lists ipv4
Thu Jan 25 10:25:19.896 IST
ipv4 access-list V4-ACL-EGRESS
```

```
 10 permit ipv4 203.0.113.1 255.255.255.0 192.0.2.1 255.255.255.0
 20 deny ipv4 any any
...

/* Apply the egress ACL to the HundredGigE interface */
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# ipv4 access-group V4-ACL-EGRESS egress
Router(config-if)# commit
Thu Jul 11 09:19:49.569 UTC
Router(config-if)# exit

/* Verify if the egress ACL has been successfully applied to the interface */
Router(config)# do show ipv4 interface
Thu Jul 11 09:01:50.445 UTC
HundredGigE 0/0/0/1 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 198.51.100/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is V4-ACL-EGRESS
  Inbound  common access list is not set, access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
...
```

You have successfully configured an IPv4 egress ACL on a HundredGigE interface. For more information on logging messages, see Understanding IP Access List Logging Messages, on page 83.

Configuring IPv6 ACLs

You can filter IP version 6 (IPv6) traffic by creating IPv6 access control lists (ACLs) and applying them to interfaces similar to the way that you create and apply IP version 4(IPv4) named ACLs.

Restrictions and Guidelines

The following restrictions and guidelines apply while configuring IPv6 ACLs:

- Ingress IPv6 ACLs are supported on all interfaces.
- From Release 7.3.1 onwards, the maximum number of ACLs allowed per router is 126.

  In earlier releases, for the Cisco 8100 and 8200 Series fixed chassis, the maximum number of ACLs allowed per router is 45. For the Cisco 8800 modular chassis, the maximum number of ACLs allowed per router is 30.
- ACL logging with input interface (using the **log-input** keyword) is not supported.
- Packet Length (using the **pkt-length** keyword) is not supported.
- TCP flags are not supported on ingress and egress IPv6 ACLs.

Configuring an Ingress IPv6 ACL on a HundredGigE Interface

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

```
/* Configure a HundredGigE interface with an IPv6 address */
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv6 address 2001::1/64
Router(config-if)# no shut
Router(config-if)# commit
Thu Jul 11 09:28:07.759 UTC
Router(config-if)# exit

/* Verify if the interface is up */
Router(config)# do show ipv6 interface brief
Thu Jul 11 09:28:43.657 UTC
HundredGigE 0/0/0/0 [Up/Up]
    fe80::bd:b9ff:fea9:5606
    2001::1
…

/* Configure an IPv6 ingress ACL */
Router(config)# ipv6 access-list V6-INGRESS-ACL
Router(config-ipv6-acl)# 10 permit ipv6 any any
Router(config-ipv6-acl)# 20 deny udp any any
Router(config-ipv6-acl)# commit
Thu Jul 11 09:41:02.625 UTC
Router(config-ipv6-acl)# exit


/* Verify the ingress ACL creation */
Router(config)# do show access-lists ipv6
Thu Jul 11 09:41:37.260 UTC
ipv6 access-list V6-INGRESS-ACL
 10 permit ipv6 any any
 20 deny udp any any


/* Apply the ingress ACL to the HundredGigE interface */
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv6 access-group V6-INGRESS-ACL ingress
Router(config-if)# commit
Thu Jul 11 09:43:59.733 UTC
Router(config-if)# exit

/* Verify if the ingress ACL has been successfully applied to the interface */
Router(config)# do show ipv6 interface
Thu Jan 25 11:34:08.028 IST
HundredGigE 0/0/0/0 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::bd:b9ff:fea9:5606
  Global unicast address(es):
    2001::1, subnet is 1001::/64
  Joined group address(es): ff02::1:ff00:1 ff02::1:ffa9:5606 ff02::2
      ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  common access list is not set, access list is V6-INGRESS-ACL
  Table Id is 0xe0800000
```

```
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
…
```

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

Configuring an Egress IPv6 ACL on a HundredGigE Interface

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

```
/* Configure a HundredGigE interface with an IPv6 address */
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# ipv6 address 2001::1/64
Router(config-if)# no shut
Router(config-if)# commit
Thu Jan 25 11:41:25.778 IST
Router(config-if)# exit

/* Verify if the interface is up */
Router(config)# do show ipv6 interface brief
Thu Jul 11 09:47:50.812 UTC
HundredGigE 0/0/0/0 [Up/Up]
    fe80::bd:b9ff:fea9:5606
    1001::1
HundredGigE 0/0/0/1 [Up/Up]
    fe80::23:e9ff:fea8:a44e
    2001::1

/* Configure an IPv6 egress ACL */
Router(config)# ipv6 access-list V6-EGRESS-ACL
Router(config-ipv6-acl)# 10 permit ipv6 any any
Router(config-ipv6-acl)# 20 deny udp any any
Router(config-ipv6-acl)# commit
Thu Jul 11 09:50:40.566 UTC
Router(config-ipv6-acl)# exit

/* Verify the egress ACL creation */
Router(config)# do show access-lists ipv6
Thu Jul 11 09:51:16.687 UTC
ipv6 access-list V6-EGRESS-ACL
 10 permit ipv6 any any
 20 deny udp any any
…

/* Apply the egress ACL to the HundredGigE interface */
Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# ipv6 access-group V6-EGRESS-ACL egress
Router(config-if)# commit
Thu Jul 11 09:52:57.751 UTC
Router(config-if)# exit

/* Verify if the egress ACL has been successfully applied to the interface */
Router(config)# do show ipv6 interface
Thu Jul 11 09:53:41.365 UTC
…
HundredGigE is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::23:e9ff:fea8:a44e
  Global unicast address(es):
```

```
    2001::1, subnet is 2001::/64
  Joined group address(es): ff02::1:ff00:1 ff02::1:ffa8:a44e ff02::2
      ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
   Hosts use stateless autoconfig for addresses.
  Outgoing access list is V6-EGRESS-ACL
  Inbound  common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
...
```

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

Configuring Ingress and Egress IPv6 ACLs on Bundle Interfaces

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

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

/* Configure an IPv6 egress ACL */
Router(config)# ipv6 access-list V6-EGRESS-ACL-bundle interface
Router(config-ipv6-acl)# 10 permit tcp any any range 3000 4000
Router(config-ipv6-acl)# 20 permit ipv6 any any
Router(config-ipv6-acl)# commit
Thu Jul 11 10:02:34.568 UTC
Router(config-ipv6-acl)# exit

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

/* Verify the egress and ingress ACL creation */
Router(config)# do show access-lists ipv6
Thu Jul 11 10:04:35.798 UTC
ipv6 access-list V6-DENY-INGRESS-ACL
 10 deny ipv6 any any
ipv6 access-list V6-EGRESS-ACL-BI
 10 permit tcp any any range 3000 4000
 20 permit ipv6 any any
...

/* Apply the egress and ingress ACLs to the bundle interface */
```

```
Router(config)# interface Bundle-Ether 1
Router(config-if)# ipv6 access-group V6-EGRESS-ACL-BI egress
Router(config-if)# ipv6 access-group V6-DENY-INGRESS-ACL ingress
Router(config-if)# commit
Thu Jul 11 10:06:06.452 UTC
Router(config-if)# exit

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

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

Configuring Extended Access Lists

Use Extended Access Lists to verify more than just the source address of the packets. Attributes such as destination address, specific IP protocols, UDP or TCP port numbers, DSCP, and so on are validated. Traffic is controlled by a comparison of the attributes stated in the ACL with those in the incoming or outgoing packets.

Configuration Example

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

```
/* Enter the global configuration mode and create the access list*/
Router# configure
Router(config)# ipv4 access-list acl_1
Router(config-ipv4-acl)# 10 remark Do not allow user1 to telnet out
/*Specify the condition to allow or deny the network traffic.*/
Router(config-ipv4-acl)# 10 permit 172.16.0.0 0.0.255.255
Router(config-ipv4-acl)# 20 deny 192.168.34.0 0.0.0.255
Router(config-ipv4-acl)commit
```

Running Configuration

```
Router#show running-config
Mon Jul 29 05:56:14.315 UTC
Building configuration...
!! IOS XR Configuration

!
ipv4 access-list acl_1
 10 permit ipv4 172.16.0.0 0.0.255.255 any
 20 deny ipv4 192.168.34.0 0.0.0.255 any
!
```

Verification

```
Router#show access-lists ipv4 acl_1 hardware ingress location 0/0/CPU0
Tue Jul 2 08:03:29.495 UTC
ipv4 access-list acl_1
66 deny igmp 30.0.20.0 0.0.0.255 30.0.10.0 0.0.0.255 v3-report (11604 matches)
67 deny igmp host 30.0.20.1 host 30.0.10.1 v2-report
```

IPv4 and IPv6 ACL in Class Map

Quality of Service (Qos) features are enhanced to support these:

- Support on L3 interface, sub-interface, bundle interface and bundle sub-interface
- Support for only ingress direction
- IPv6-supported match fields:
  - Destination Port
  - Fragment bit
  - ICMP type and code
  - IGMP type and code
  - IPv6 Destination Address
  - IPv6 Source Address
  - IPv6 Protocol
  - Precedence/DSCP
  - Source Port

Configuring IPv6 ACL QoS - An Example

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

```
ipv6 access-list aclv6
10 permit ipv6 1111:6666::2/64 1111:7777::2/64 authen
30 permit tcp host 1111:4444::2 eq 100 host 1111:5555::2
```

```
!

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

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

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

User-Defined TCAM Keys for IPv4 and IPv6

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

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

- Direction of attachment only for ingress
- Protocol type (IPv4/IPv6)

Because the default key definitions are constrained (do not include all qualifier/action fields), User-Defined Key (UDK) definitions are supported for the following types:

- Traditional Ingress IPv4 ACL (uncompressed)
- Traditional Ingress IPv6 ACL (uncompressed)

The User-Defined TCAM Key (UDK) functionality provides the flexibility to define your own TCAM key for ingress, traditional, or IPv4 and IPv6 ACL only:

To include the well-known fields in the default TCAM key, see IPv4 and IPv6 Key Formats, on page 74.

A User-Defined TCAM Key (UDK) can be defined globally or locally per line card. If both global and local UDK is available for a line card, then global UDK is be ignored for the line card.

A UDK can be configured using the following command:

```
hw-module profile tcam format access-list [ipv4 | ipv6] field1 field2[location rack/slot/cpu0]
```

To define UDK globally, you can ignore the location option.

```
hw-module profile tcam format access-list [ipv4 | ipv6] field1field2
```

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

User-Defined Fields

TCAM key consists of several qualifiers. Use the sets of qualifiers 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] [l3 | l4]* **offset** *byte-offset* **length** *no of bytes*

You can add the UDF 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 You can define up to 8 UDFs systemwide. Currently, you can define UDFs globally.

Restrictions

- Cisco 8000 Series Routers support UDF only on L3 interfaces and not on L2 ports.
- Deep Packet Inspection is available only up to 128 Bytes inside the packet initiating from the L2 header.
- Cisco 8000 Series Routers support UDF only in ingress direction.
- Cisco 8000 Series Routers support only L3 and L4 base offsets for both inner and outer headers.
- Cisco 8000 Series Routers support only four bytes of match length.
- Modification for UDF configuration is allowed, but you must reload the related line card to be effective using the **reload location** *node Id* command.
- Modification for UDK is not supported using the **hw-module profile tcam format** command.

  First, remove the existing UDK using the **no hw-module profile tcam format** command, then add a new UDK definition.
- If you configure UDK, you cannot use the default keys. But you can explicitly define the default fields in UDK.

Configuration Example

Steps

1. Create an UDF and define UDK.
2. Manually, reload the node on the line card.
3. Configure ACL using fields defined in UDK.
4. Attach ACL to an interface in ingress direction.

Configuration

The following example shows how to deny packets with the following condition:

- The packets with the source address as 192.0.2.0 and destination address as 203.0.113.0
- The packets with the payload pattern of 0x4567 at an offset of 48 bytes from the L3 header

```
/* Create an UDF and define global UDK and UDF*\
Router#configure terminal
Router(config)#udf udf_outerl3_2b header outer l3 offset 48 length 2
Router(config)#hw-module profile tcam format access-list ipv4 ipv4-sip ipv4-dip udf1
udf_outerl3_2b

In order to activate/deactivate this ipv4 profile, you must manually reload line cards
Router(config)#exit

 /* Reload the node on the Cisco 8000 Series Routers line card*\
Router#reload location 0/8/CPU0
Wed Aug 21 14:12:40.123 UTC
Proceed with reload? [confirm]

Router#:Aug 21 14:12:44.120 UTC: fsdbagg[216]: %PKT_INFRA-FM-4-FAULT_MINOR : ALARM_MINOR
:FABRIC-PLANE-5 :DECLARE :: Fabric Plane-5 DOWN
Router:Aug 21 14:12:44.123 UTC: fsdbagg[216]: %PKT_INFRA-FM-4-FAULT_MINOR : ALARM_MINOR
:FABRIC-PLANE-6 :DECLARE :: Fabric Plane-6 DOWN

/*Configure ACL for User Defined Key*\
Router(config)#ipv4 access-list acl1
Router(config-ipv4-acl)10 deny ipv4 192.0.2.0 203.0.113.0 any udf udf_outerl3_2b 0x4567
0xffff
Router(config-ipv4-acl)#exit
Router(config)#interface HundredGigE 0/0/0/24
Router (config-if)#ipv4 access-group acl1 ingress
```

The following example shows how to deny the `IP in IP Tunneling` packet with the following condition:

- Packet with the inner IP header having the DSCP as AF42.
- Packet with the payload pattern of 0x12345678 at offset of 20 bytes from outer layer 4 header.

```
Router#configure terminal
Router(config)#udf udf_innerl3_1b header inner l3 offset  1  length 1
Router(config)#udf udf_outerl4_4b header outer l4 offset  20  length 4
Router(config)#hw-module profile tcam format access-list ipv4 protocol udf1 udf_innerl3_1b
  udf udf_outerl4_4b
```

IPv4 and IPv6 Key Formats

The following table shows the qualifier fields that are supported in the IPv4 and IPv6 key formats.

Table 6: Qualifier Fields Supported in IPv4 and IPv6 Key Formats

Parameter	Default TCAM Key	
	IPv4	IPv6
Destination Address	Supported	Supported

Parameter	Default TCAM Key	
	IPv4	IPv6
Destination Port	Supported	Supported
Fragment bit	Supported	Supported
ICMP type and code	Supported	Supported
IGMP type and code	Supported	Supported
Protocol/Next Header	Supported	Supported
Precedence/DSCP	Supported	Supported
Source Address	Supported	Supported
Source Port	Supported	Supported
TCP Flags	Supported	Not supported
Time to live (TTL) Match	Supported	Not supported
UDF 1-8	Not supported	Not supported

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

Table 7: Action Fields Supported in IPv4 and IPv6 Key Formats

Parameter	Default Action Field	
	IPv4	IPv6
Permit	Supported	Supported
Deny	Supported	Supported
Next Hop	Supported	Supported
Log	Supported for Ingress only	Supported for Ingress only
Capture	Supports only ingress Encapsulated Remote SPAN (ERSPAN)	Supports only ingress Encapsulated Remote Switch port Analyzer (ERSPAN)
Stats Counter	Deny stats is always enabled (Permit stats is not supported)	Deny stats is always enabled (Permit stats is not supported)

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#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*/
```

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 through packet-based destination address lookup. This feature of using next-hop in ACL configurations for forwarding is called ACL Based Forwarding (ABF).

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

Restrictions

- Traffic outages can occur during transitions from an existing nexthop to another nexthop.
- IPv4 and IPv6 ABF nexthops routed over GRE interfaces are not supported.
- VRF-select (where only the VRF is configured for the nexthop) is not supported in ABF for IPv4 and IPv6 addresses.
- Logging of permit statistics for ABF is not supported.

Feature Highlights

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

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

- As ABF is ACL-based, packets that do not match an existing rule (ACE) in the ACL are subjected to the default ACL rule (drop all). If the ACL is being used for ABF-redirect only (not for security), then include an explicit ACE rule at the end of the ACL (lowest user priority) to match and "permit" all traffic. This ensures that all traffic that does not match an ABF rule is permitted and forwarded as normal.
- ABF is supported on permit rules only.
- ABF default route is not supported.
- Packets punted in the ingress direction from the NPU to the linecard CPU are not subjected to ABF treatment due to lack of ABF support in the slow path. These packets will be forwarded normally based on destination-address lookup by the software dataplane. Some examples of these types of packets are (but are not limited to) packets with IPv4 options, IPv6 extension headers, and packets destined for glean (unresolved/incomplete) adjacencies.

Configuration Example

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

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

Configuration

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

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

/* Set the conditions for the ACL and configure ABF: */
/* The next hop for this entry is specified. */
Router(config-ipv4-acl)# 10 permit ipv4 192.168.18.0 0.255.255.255 any nexthop1 ipv4
192.168.20.2
Router(config-ipv4-acl)# 15 permit ipv4 192.168.21.0 0.0.0.255 any
Router(config-ipv4-acl)# 20 permit ipv4 192.168.22.0 0.0.255.255 any nexthop1 ipv4
192.168.23.2
/* More than two nexthops */
Router(config-ipv4-acl)# 25 permit tcp any range 2000 3000 any range 4000 5000 nexthop1
ipv4 192.168.23.1 nexthop2 ipv4 192.168.24.1 nexthop3 ipv4 192.168.25.1

/* VRF support on ABF */
Router(config-ipv4-acl)# 30 permit tcp any eq www host 192.168.12.2 precedence immediate
nexthop1 vrf vrf1_ipv4 ipv4 192.168.13.2 nexthop2 vrf vrf1_ipv4 ipv4 192.168.14.2

Router(config-ipv4-acl)# 35 permit ipv4 any any

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

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

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

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

/* More than two nexthops */
Router(config-ipv6-acl)# 25 permit tcp any range 2000 3000 any range 4000 5000 nexthop1
ipv6 2001:db8::3 nexthop2 ipv6 2001:db8::4 nexthop3 ipv6 2001:db8::5

/* VRF support on ABF */
Router(config-ipv6-acl)# 30 permit tcp any eq www host 2001:db8::8 precedence immediate
nexthop1 vrf vrf1_ipv6 ipv6 2001:db8::7 nexthop2 vrf vrf1_ipv6 ipv6 2001:db8::6

Router(config-ipv6-acl)# 35 permit ipv6 any any

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

Running Configuration

```
Router# show access-lists ipv4
ipv4 access-list abf-acl
10 permit ipv4 192.168.18.0 0.255.255.255 any nexthop1 192.168.20.2
15 permit ipv4 192.168.21.0 0.0.0.255 any
20 permit ipv4 192.168.22.0 0.0.255.255 any nexthop1 192.168.23.2
25 permit tcp any range 2000 3000 any range 4000 5000 nexthop1 ipv4 192.168.23.1 nexthop2
ipv4 192.168.24.1 nexthop3 ipv4 192.168.25.1
30 permit tcp any eq www host 192.168.12.2 precedence immediate nexthop1 vrf vrf1_ipv4 ipv4
 192.168.13.2 nexthop2 vrf vrf1_ipv4 ipv4 192.168.14.2
35 permit ipv4 any any
!
Router# show access-lists ipv6
ipv6 access-list abf-acl-ipv6
 10 permit ipv6 2001:db8::/32 any nexthop1 ipv6 2001:db8::2
 25 permit tcp any range 2000 3000 any range 4000 5000 nexthop1 ipv6 2001:db8::3 nexthop2
ipv6 2001:db8::4 nexthop3 ipv6 2001:db8::5
 30 permit tcp any eq www host 2001:db8::8 precedence immediate nexthop1 vrf vrf1_ipv6 ipv6
```

```
 2001:db8::7 nexthop2 vrf vrf1_ipv6 ipv6 2001:db8::6
 35 permit ipv6 any any
```

Verification

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

```
Router# show access-lists ipv4 abf nexthops client pfilter_ea location 0/3/CPU0
Tue May 17 22:25:05.940 UTC

 ACL name : abf-acl
  ACE seq.           NH-1                  NH-2                  NH-3
 ---------  ---------------------  ---------------------  ---------------------
        20    Global 192.168.23.2            Not present            Not present
    status                     UP            Not present            Not present
     exist                     No            Not present            Not present
    pd ctx                Present            Not present            Not present
              Track not present      Track not present                     --
        25    Global 192.168.23.1    Global 192.168.24.1        Global 192.168.25.1
    status                     UP                     UP                     UP
     exist                    Yes                    Yes                    Yes
    pd ctx                Present                Present                Present
              Track not present      Track not present            Track not present
```

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

```
Router# show access-lists usage pfilter location all
sh access-lists ipv4 abf nexthops client pfilter_ea loc 0/RP0/CPU0
Wed Jul 29 20:48:18.559 UTC

ACL name : abf-1
  ACE seq.         NH-1             NH-2             NH-3
 ---------  ---------------  ---------------  ---------------
        10    27.138.216.32         28.0.0.2      Not present
    status               UP               UP      Not present
at status       Not Present      Not Present      Not present
     exist                No              Yes      Not present
       vrf           default          default      Not present
     track       Not present      Not present      Not present
    pd ctx           Present          Present      Not present
```

Associated Commands

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

Associated Topics

- Configuring IPv4 ACLs
- Configuring IPv6 ACLs

Access Control List Counters

In Cisco IOS XR software, ACL counters are maintained both in hardware and software. Hardware counters are used for packet filtering applications. Software counters are used by all the applications mainly involving software packet processing.

Software counters are updated for the packets processed in software, for example, exception packets punted to the LC CPU for processing, or ACL used by routing protocols, and so on. The counters that are maintained are an aggregate of all the software applications using that ACL. To display software-only ACL counters, use the **show access-lists ipv4** *access-list-name* [**sequence** *number*] command in EXEC mode.

Configuring ACLs with Fragment Control

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

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

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

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

If the Access-List Entry has...	Then...

...no **fragments** keyword and all of the access-list entry information matches	For an access-list entry containing only Layer 3 information:
	• The entry is applied to non-fragmented packets, initial fragments, and non-initial fragments.
	For an access-list entry containing Layer 3 and Layer 4 information:
	• The entry is applied to non-fragmented packets and initial fragments.
	  • If the entry matches and is a **permit** statement, the packet or fragment is permitted.
	  • If the entry matches and is a **deny** statement, the packet or fragment is denied.
	• The entry is also applied to non-initial fragments in the following manner. Because non-initial fragments contain only Layer 3 information, only the Layer 3 portion of an access-list entry can be applied. If the Layer 3 portion of the access-list entry matches, and
	  • If the entry is a **permit** statement, the non-initial fragment is permitted.
	  • If the entry is a **deny** statement, the next access-list entry is processed.
	Note The deny statements are handled differently for non-initial fragments versus non-fragmented or initial fragments.

...the **fragments** keyword and all of the access-list entry information matches	The access-list entry is applied only to non-initial fragments. **Note** If the **fragments** keyword is configured for an access-list entry, the Layer 4 information will be ignored for the non-initial fragments.

You should not add the **fragments** keyword to every access-list entry, because the first fragment of the IP packet is considered a non-fragment and is treated independently of the subsequent fragments. Because an initial fragment will not match an access list permit or deny entry that contains the **fragments** keyword, the packet is compared to the next access list entry until it is either permitted or denied by an access list entry that does not contain the **fragments** keyword. Therefore, you may need two access list entries for every deny entry. The first deny entry of the pair will not include the **fragments** keyword, and applies to the initial fragment. The second deny entry of the pair will include the **fragments** keyword and applies to the subsequent fragments. In the cases where there are multiple **deny** access list entries for the same host but with different Layer 4 ports, a single deny access-list entry with the **fragments** keyword for that host is all that has to be added. Thus all the fragments of a packet are handled in the same manner by the access list.

Packet fragments of IP datagrams are considered individual packets and each fragment counts individually as a packet in access-list accounting and access-list violation counts.

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

Note Within the scope of ACL processing, Layer 3 information refers to fields located within the IPv4 header; for example, source, destination, protocol. Layer 4 information refers to other data contained beyond the IPv4 header; for example, source and destination ports for TCP or UDP, flags for TCP, type and code for ICMP.

Configuring TTL Matching

You can configure ACLs to match on the TTL value specified in the IPv4 header. You can specify the TTL match condition to be based on a single value, or multiple values.

TTL matching is supported for both ingress and egress ACLs.

Configuration

Use the following steps to configure TTL matching.

```
/* Configure an IPv4 ACL with the TTL parameters */
Router(config)# ipv4 access-list acl-v4
Router(config-ipv4-acl)# 10 deny tcp any any ttl eq 100
Router(config-ipv4-acl)# 20 permit tcp any any ttl range 1 50
Router(config-ipv4-acl)# 30 permit tcp any any ttl neq 100
Router(config-ipv4-acl)# commit
Thu Nov  2 12:22:58.948 IST

/* Attach the IPv4 ACL to the HundredGigE interface */
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# ipv4 address 15.1.1.1 255.255.255.0
Router(config-if)# ipv4 access-group acl-v4 ingress
Router(config-if)# commit
```

Running Configuration

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

```
Router(config)# show run
Thu Nov  2 14:01:53.376 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Thu Nov  2 12:22:59 2017 by annseque
!
ipv4 access-list acl-v4
 10 deny tcp any any ttl eq 100
 20 permit tcp any any ttl range 1 50
 30 permit tcp any any ttl neq 100
!
interface HundredGigE 0/0/0/0
 ipv4 address 15.1.1.1 255.255.255.0
 ipv4 access-group acl-v4 ingress
!
```

You have successfully configured TTL matching for IPv4 ACLs.

Understanding IP Access List Logging Messages

Cisco IOS XR software can provide logging messages about packets permitted or denied by a standard IP access list. That is, any packet that matches the access list causes an informational logging message about the packet to be sent to the console. The level of messages logged to the console is controlled by the **logging console** command in global configuration mode.

The first packet that triggers the access list causes an immediate logging message, and subsequent packets are collected over 5-minute intervals before they are displayed or logged. The logging message includes the access list number, whether the packet was permitted or denied, the source IP address of the packet, and the number of packets from that source permitted or denied in the prior 5-minute interval.

However, you can use the { **ipv4 | ipv6** } **access-list log-update threshold** command to set the number of packets that, when they match an access list (and are permitted or denied), cause the system to generate a log message. You might do this to receive log messages more frequently than at 5-minute intervals.

Caution If you set the *update-number* argument to 1, a log message is sent right away, rather than caching it; every packet that matches an access list causes a log message. A setting of 1 is not recommended because the volume of log messages could overwhelm the system.

Even if you use the { **ipv4 | ipv6} access-list log-update threshold** command, the 5-minute timer remains in effect, so each cache is emptied at the end of 5 minutes, regardless of the number of messages in each cache. Regardless of when the log message is sent, the cache is flushed and the count reset to 0 for that message the same way it is when a threshold is not specified.

Note The logging facility might drop some logging message packets if there are too many to be handled or if more than one logging message is handled in 1 second. This behavior prevents the router from using excessive CPU cycles because of too many logging packets. Therefore, the logging facility should not be used as a billing tool or as an accurate source of the number of matches to an access list.

Per Interface Statistics

When binding ACL to interfaces, you can configure ACE drop counts by using the **interface-statistics** keyword in the per-interface mode.

In Cisco 8000 Series Routers, you can allocate up to 8 stats counters per NPU.

This also limits the number of the interface that can support the same ACL in per-interface-stats mode to 8. Additional binding of the same ACL in per-interface-stats mode will be rejected.

Note For a specific direction, all interfaces on a line card using the same ACL must be configured in the same stats mode, if not the subsequent binding will be rejected.

This is true for the bundle interface too. If you add any member to the existing bundle interface with a different stats mode, the binding will be rejected.

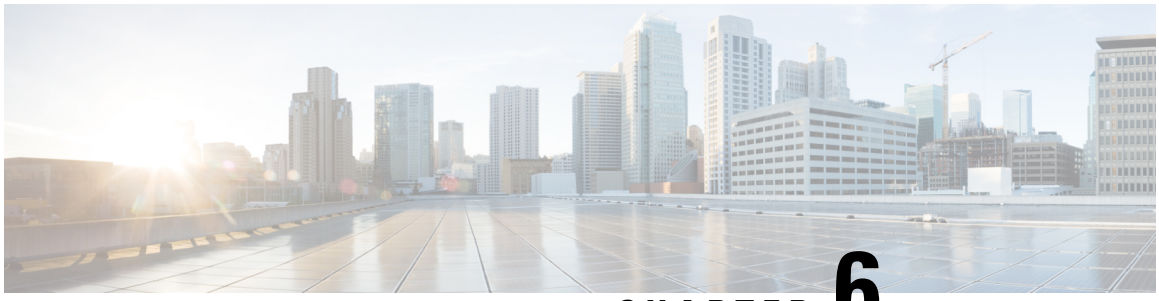
Configuration Example

```
Router# interface HundredGigE 0/0/0/0
Router(config-if)#ipv4 address 1.1.1.1 255.255.0.0
Router(config-if)#ipv6 address 2001::1/64
Router(config-if)#ipv4 access-group test ingress interface-statistics
Router(config-if)#commit
```

Verification

To display deny stats on the interface:

```
Router#show access-lists ipv4 test hardware ingress interface
 FourHundredGigE 0/1/0/0 location 0/1/CPU0

ipv6 access-list test
10 permit ipv6 any 200:23::/64
20 deny udp any any (2356 matches)
```



CHAPTER 6

Implementing Cisco Express Forwarding

Implementing Cisco Express Forwarding

Cisco Express Forwarding (CEF) is an advanced, Layer 3 IP switching technology. CEF optimizes network performance and scalability for networks with large and dynamic traffic patterns, such as the Internet, on networks characterized by intensive web-based applications, or interactive sessions. CEF is an inherent feature and the users need not perform any configuration to enable it. If required, the users can change the default route purge delay and static routes. Cisco 8000 Series Routers supports only single stage forwarding.

Components

Cisco IOS XR software CEF always operates in CEF mode with two distinct components:

- Forwarding Information Base (FIB) database: The protocol-dependent FIB process maintains the forwarding tables for IPv4 and IPv6 unicast in the route processor and line card (LC). The FIB on each node processes Routing Information Base (RIB) updates, performing route resolution and maintaining FIB tables independently in the route processor and line card (LC). FIB tables on each node can be slightly different.
- Adjacency table—a protocol-independent adjacency information base (AIB)

CEF is a primary IP packet-forwarding database for Cisco IOS XR software. CEF is responsible for the following functions:

- Software switching path
- Maintaining forwarding table and adjacency tables (which are maintained by the AIB) for software and hardware forwarding engines

The following features are supported for CEF on Cisco IOS XR software:

- Bundle interface support

- Multipath support
- Route consistency
- High availability features such as packaging, restartability, and Out of Resource (OOR) handling
- OSPFv2 SPF prefix prioritization
- BGP attributes download

CEF Benefits

- Improved performance—CEF is less CPU-intensive than fast-switching route caching. More CPU processing power can be dedicated to Layer 3 services such as quality of service (QoS) and encryption.
- Scalability—CEF offers full switching capacity at each line card.
- Resilience—CEF offers an unprecedented level of switching consistency and stability in large dynamic networks. In dynamic networks, fast-switched cache entries are frequently invalidated due to routing changes. These changes can cause traffic to be process switched using the routing table, rather than fast switched using the route cache. Because the Forwarding Information Base (FIB) lookup table contains all known routes that exist in the routing table, it eliminates route cache maintenance and the fast-switch or process-switch forwarding scenario. CEF can switch traffic more efficiently than typical demand caching schemes.

The following CEF forwarding tables are maintained in Cisco IOS XR software:

- IPv4 CEF database—Stores IPv4 Unicast routes for forwarding IPv4 unicast packets
- IPv6 CEF database—Stores IPv6 Unicast routes for forwarding IPv6 unicast packets

Prerequisites for Implementing Cisco Express Forwarding

The following prerequisites are required to implement Cisco Express Forwarding:

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

Verifying CEF

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

- show cef {ipv4 | ipv6} summary

Displays a summary of the IPv4 or IPv6 CEF table.

```
Router#show cef ipv4 summary
Fri Nov 20 13:50:45.239 UTC

Router ID is 216.1.1.1

IP CEF with switching (Table Version 0) for node0_RP0_CPU0
```

```
 Load balancing: L4
 Tableid 0xe0000000 (0x8cf5b368), Vrfid 0x60000000, Vrid 0x20000000, Flags 0x1019
 Vrfname default, Refcount 4129
 56 routes, 0 protected, 0 reresolve, 0 unresolved (0 old, 0 new), 7616 bytes
 13 rib, 0 lsd, 0:27 aib, 1 internal, 10 interface, 4 special, 1 default routes
 56 load sharing elements, 24304 bytes, 1 references
 1 shared load sharing elements, 432 bytes
 55 exclusive load sharing elements, 23872 bytes
 0 route delete cache elements
 13 local route bufs received, 1 remote route bufs received,  0 mix bufs received
 13 local routes, 0 remote routes
 13 total local route updates processed
 0 total remote route updates processed
 0 pkts pre-routed to cust card
 0 pkts pre-routed to rp card
 0 pkts received from core card
 0 CEF route update drops, 0 revisions of existing leaves
 0 CEF route update drops due to version mis-match
 Resolution Timer: 15s
 0 prefixes modified in place
 0 deleted stale prefixes
 0 prefixes with label imposition, 0 prefixes with label information
 0 LISP EID prefixes, 0 merged, via 0 rlocs
28 next hops
 1 incomplete next hop

 0 PD backwalks on LDIs with backup path
```

- show cef { ipv4 address | ipv6 address } detail

Displays the details of the IPv4 or IPv6 CEF table.

```
Router#show cef 203.0.1.2 detail
203.0.1.2/32, version 102239, internal 0x1000001 0x0 (ptr 0xa932b408) [1], 0x0 (0xaf4a6ad8),
 0xa20 (0xc22c6da8)
 Updated Jul  3 21:40:17.827
 local adjacency 203.1.104.2
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
  gateway array (0xb9061e70) reference count 1982, flags 0x8068, source lsd (5), 1 backups
                [1983 type 4 flags 0x108401 (0x943df068) ext 0x0 (0x0)]
  LW-LDI[type=1, refc=1, ptr=0xaf4a6ad8, sh-ldi=0x943df068]
  gateway array update type-time 1 Jul  3 20:23:36.957
 LDI Update time Jul  3 20:23:36.964
 LW-LDI-TS Jul  3 21:40:17.834
   via 203.1.104.2/32, Bundle-Ether104, 11 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0xa446b0a8 0x0]
    next hop 203.1.104.2/32
    local adjacency
   via 203.1.114.2/32, Bundle-Ether114, 9 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 1 NHID 0x0 [0xa446ac18 0x0]
    next hop 203.1.114.2/32
    local adjacency

    Load distribution: 0 1 (refcount 1983)

    Hash  OK  Interface                 Address
    0     Y   Bundle-Ether104           203.1.104.2
    1     Y   Bundle-Ether114           203.1.114.2
```

Configuring Static Route

Routers forward packets using either route information from route table entries that you manually configure or the route information that is calculated using dynamic routing algorithms. Static routes, which define explicit paths between two routers, cannot be automatically updated; you must manually reconfigure static routes when network changes occur. Static routes use less bandwidth than dynamic routes. Use static routes where network traffic is predictable and where the network design is simple. You should not use static routes in large, constantly changing networks because static routes cannot react to network changes. Most networks use dynamic routes to communicate between routers but might have one or two static routes configured for special cases. Static routes are also useful for specifying a gateway of last resort (a default router to which all unroutable packets are sent).

Configuration Example

Create a static route between Router A and B over a HundredGigE interface. The destination IP address is 203.0.113.0/24 and the next hop address is 192.0.2.1.


```
Router(config)#router static address-family ipv4 unicast
Router(config-static-afi)#203.0.113.0/24 HundredGigE0/0/0/0 192.0.2.1
Router(config-static-afi)#commit
```

Running Configuration

```
Router#show running-config router static address-family ipv4 unicast
router static
 address-family ipv4 unicast
 203.0.113.0/24 HundredGigE0/0/0/0 192.0.2.1
 !
!
```

Associated Commands

- router static
- show cef

BGP Attributes Download

The BGP Attributes Download feature enables you to display the installed BGP attributes in CEF.

- The **show cef bgp-attribute** command displays the installed BGP attributes in CEF.
- The **show cef bgp-attribute** *attribute-id* command and the **show cef bgp-attribute** *local-attribute-id* command are used to view the specific BGP attributes by attribute ID and local attribute ID.

Verification

```
Router# show cef bgp-attribute
Wed Aug 21 14:05:51.772 UTC

VRF: default

_____________
Table ID: 0xe0000000. Total number of entries: 1
OOR state: GREEN. Number of OOR attributes: 0

BGP Attribute ID: 0x6, Local Attribute ID: 0x1
    Aspath      :    2
    Community   :
    Origin AS   :    2
    Next Hop AS :    2
```

Proactive Address Resolution Protocol and Neighbor Discovery

When CEF installs a route for which there is no layer 2 adjacency information, CEF creates an incomplete layer 3 next-hop and programs it on the hardware. Because of this incomplete programming, the first packet will be forwarded to the software forwarding path. The software forwarding in turn strips off the layer 2 header from the packet and forwards it to ARP (Address Resolution Protocol) or ND (Neighbor Discovery) in order to resolve the layer 2 adjacency information. In such a packet, if there is feature specific information present in the layer 2 header, the software forwarding path fails to strip off the layer 2 header completely and thus ARP or ND is unable to resolve the missing layer 2 adjacency information and thereby this results in traffic being dropped.

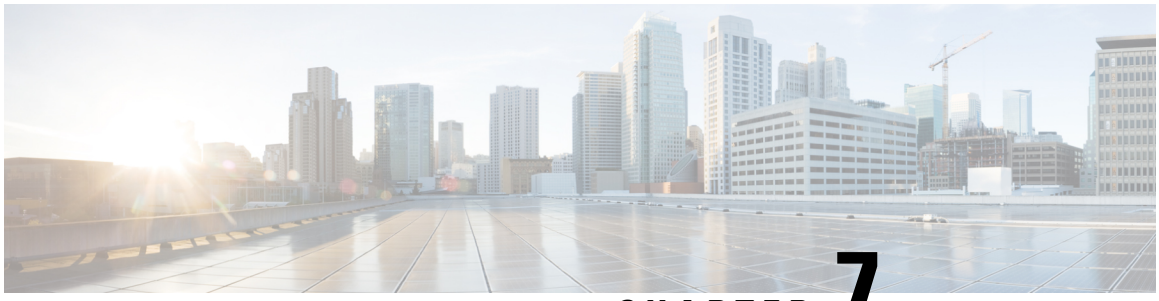
Proactive ARP and ND feature solves the above problem by ensuring that CEF proactively triggers ARP or ND in order to resolve the missing layer 2 adjacency information, retrying every 15 seconds until the next-hop information is resolved. Thus, when you configure a static route which has an incomplete next-hop information, this feature automatically triggers ARP or ND resolution.

Configuration

```
/* Enter the configuration mode and configure Proactive ARP/ND */
Router# configure
Router(config)# cef proactive-arp-nd enable
Router(config)# commit
```

Running Configuration

```
Show running-config
cef proactive-arp-nd enable
end
```



CHAPTER 7

Implementing LPTS

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 effective current rates of the flow types from the output of the following show command:

```
show lpts pifib hardware police
```

Configuration Example

Configure the LPTS policer for the OSPF and BGP flowtypes with the following values globally for all nodes:

- ospf unicast default rate 3000
- bgp default rate 4000

```
Router#configure
Router(config)#lpts pifib hardware police
Router(config-lpts-policer-global)#flow ospf unicast default rate 3000
Router(config-lpts-policer-global)#flow bgp default rate 4000
Router(config-lpts-policer-global)#commit
```

Running Configuration

```
Router#show running-config lpts
lpts pifib hardware police
 flow ospf unicast default rate 3000
 flow bgp default rate 4000
!
```

Verification

```
Router#show lpts pifib hardware police

-------------------------------------------------------------
                Node 0/RP0/CPU0:
-------------------------------------------------------------
FlowType                Policer Type    Cur. Rate Burst     Accepted     Dropped      npu
----------------------- ------- ------- --------- --------- ------------ ------------ ---------
Fragment                2       np      542       1000      0            0            0
OSPF-mc-known           3       np      1627      1000      0            0            0
OSPF-mc-default         4       np      1084      1000      0            0            0
OSPF-uc-known           5       np      542       1000      0            0            0
OSPF-uc-default         6       np      3000      1000      0            0            0
BFD-default             10      np      8136      1000      0            0            0
BFD-MP-known            11      np      8136      1000      0            0            0
BGP-known               16      np      17000     1000      0            0            0
BGP-cfg-peer            17      np      1627      1000      0            0            0
BGP-default             18      np      4000      1000      0            0            0
```

Configuration Example

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

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

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

Running Configuration

```
Router#show running-config lpts
lpts pifib hardware police location 0/0/CPU0
 flow ospf unicast default rate 3000
 flow bgp default rate 4000
!
```

Verification

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

```
Router#show lpts pifib hardware police location 0/0/CPU0
-------------------------------------------------------------
                Node 0/0/CPU0:
-------------------------------------------------------------
FlowType               Policer Type    Cur. Rate Burst     Accepted     Dropped      npu
---------------------- ------- ------- --------- --------- ------------ ------------ ---------
Fragment               2       np      542       1000      0            0            0
Fragment               2       np      542       1000      0            0            1
OSPF-mc-known          3       np      1627      1000      0            0            0
OSPF-mc-known          3       np      1627      1000      0            0            1
OSPF-mc-default        4       np      1084      1000      0            0            0
OSPF-mc-default        4       np      1084      1000      0            0            1
OSPF-uc-known          5       np      542       1000      0            0            0
OSPF-uc-known          5       np      542       1000      0            0            1
OSPF-uc-default        6       np      3000      1000      0            0            0
OSPF-uc-default        6       np      3000      1000      0            0            1
BFD-default            10      np      8136      1000      0            0            0
BFD-default            10      np      8136      1000      0            0            1
BFD-MP-known           11      np      8136      1000      0            0            0
BFD-MP-known           11      np      8136      1000      0            0            1
BGP-known              16      np      17000     1000      0            0            0
BGP-known              16      np      17000     1000      0            0            1
BGP-cfg-peer           17      np      1627      1000      0            0            0
BGP-cfg-peer           17      np      1627      1000      0            0            1
BGP-default            18      np      4000      1000      0            0            0
BGP-default            18      np      4000      1000      0            0            1
```

Associated Commands

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

LPTS and NPU Traps

Network Processing Unit (NPU) traps are raised by the routers for inspection. NPU traps are raised in response to the type of packets received by the router and can indicate either exception packets, error packets, or non-LPTS control packets.

- Examples of exception packets include glean adjacency traffic or packets with IPv4 options.
- Examples of error packets include IPv4 packet with bad checksum or IPv6 packets with a hop count of zero.

- Examples of non-LPTS control packets include those packets that do not get processed through LPTS (for example, LACP, LLDP and other L2 control packets).

Each of the NPU traps are policed at a rate that is pre-programmed by the router's system design. Packets are policed per NPU and excess traffic is dropped by the NPU with respect to the system design. Some NPU trap packets that are allowed by NPU policers are sent to the CPU if they need additional processing. Others that exceed the NPU policer rate are dropped by the NPU.

Verification

Use the command `show controllers npu stats traps-all instance NPU-Number|all location RP|LC` command to check the NPU trap statistics for all the NPUs or per NPU of a router.

For fixed systems, the NPU trap statistics is available for the location 0/RP0/CPU0 and is provided through the command `show controllers npu stats traps-all instance all location 0/RP0/CPU0`. For distributed systems, NPU trap statistics is available for the line card locations and is provided through the command `show controllers npu stats traps-all instance all location 0/1/CPU0`. You can use the command `clear controller npu stats traps-all instance NPU-Number|all location RP|LC`

In the following example:

- **(D)** indicates the trap packets that are dropped in the NPU.
- **(D*)** indicates the trap packets that are dropped in NPU but are available for analysis.
- The **Accepted** count in the output indicates the ones that are available for analysis.

```
Router# show controllers npu stats traps-all instance all location 0/RP/cpu0
Fri Oct 11 05:17:22.720 UTC

Trap Type                                NPU  Trap     TrapStats     Policer             Packet             Packet
                                         ID    ID      ID                                Accepted       Dropped
=============================================================================================================
ETHERNET_ACL_DROP(D)                     0    0        0x0           1                   0                  0
ETHERNET_ACL_FORCE_PUNT(D*)              0    1        0x0           1                   0                  0
ETHERNET_VLAN_MEMBERSHIP(D*)             0    2        0x0           1                   0                  0
ETHERNET_ACCEPTABLE_FORMAT                  0    3        0x0           258                    0
 0
UNKNOWN_VLAN_OR_BUNDLE_MEMBER(D*)        0    4        0x0           259                 0                  0
NOT_MY_MAC(D*)                           0    5        0x0           260                 0                  0
ETHERNET_NO_SIP_MAPPING(D*)              0    6        0x0           1                   0                  0
ETHERNET_NO_VNI_MAPPING(D*)              0    7        0x0           1                   0                  0
ETHERNET_NO_VSID_MAPPING(D*)             0    8        0x0           1                   0                  0
ARP                                         0    9        0x0           264                    0
 0
ETHERNET_SA_ERROR(D*)                    0    11       0x0           266                 0                  0
ETHERNET_DA_ERROR(D*)                    0    12       0x0           1                   0                  0
ETHERNET_SA_MULTICAST(D*)                0    13       0x0           268                 0                  0
DHCPV4_SERVER                               0    14       0x0           269                    0
 0
DHCPV4_CLIENT                               0    15       0x0           270                    0
 0
ETHERNET_INGRESS_STP_BLOCK(D*)           0    18       0x0           1                   0                  0
PTP_OVER_ETHERNET                           0    16       0x0           274                    0
 0
.                                           .    .        .             .                      .
 .
.                                           .    .        .             .                      .
 .
.                                           .    .        .             .                      .
 .
.                                           .    .        .             .                      .
 .
.                                           .    .        .             .                      .
 .
OAMP_BFD_INCORRECT_TTL(D*)               0    157      0x0           412                 0                  0
OAMP_BFD_INVALID_PROTOCOL(D*)            0    158      0x0           413                 0                  0
OAMP_BFD_INVALID_UDP_PORT(D*)            0    159      0x0           414                 0                  0
OAMP_BFD_INCORRECT_VERSION(D*)           0    160      0x0           415                 0                  0
```

```
OAMP_BFD_INCORRECT_ADDRESS(D*)        0    161      0x0          416                    0                  0
OAMP_BFD_MISMATCH_DISCR                    0    162      0x0          417                    0                  0
OAMP_BFD_STATE_FLAG_CHANGE                 0    163      0x0          418                    0                  0
OAMP_BFD_SESSION_RECEIVED(D*)         0    164      0x0          419                    0                  0
OAMP_PFC_LOOKUP_FAILED(D*)            0    165      0x0          420                    0                  0
OAMP_PFC_DROP_INVALID_RX(D*)          0    166      0x0          1                      0                  0
APP_SGACL_DROP (D*)                   0    168      0x0          1                      0                  0
```

Defining Dynamic LPTS Flow Type

The Dynamic LPTS flow type feature enables you to configure LPTS flow types and also enables you to define the maximum LPTS entries for each flow type in the TCAM. The dynamic LPTS flow type configuration is on per line card basis, hence you can have multiple profiles configured across line cards.

When the router boots, the default LPTS flow types are programmed in the TCAM. For each flow type the maximum flow entries are predefined. Later, at runtime, you have an option to choose the flow type based on network requirements and also configure the maximum flow entry value. The maximum flow entry value of zero denotes that a flow type is not configured.

Note You can get the default maximum flow values for both configurable flow and non-configurable flow from the output of the following show command:

```
show lpts pifib dynamic-flows statistics location <location specification>
```

The list of configurable and non-configurable flow types are listed in below tables. You can also use **show lpts pifib dynamic-flows statistics location** command to view the list of configurable and non-configurable flow types:

Note The sum of maximum LPTS entries configured for all flow types must not exceed 16000 entries per line card.

Configuration Example

In this example you will configure the BGP-known and ISIS-known LPTS flow type in the TCAM and define the maximum flow entries as 1800 and 500 for node location 0/1/CPU0. As the new maximum values are more than the default values, we have to create space in the TCAM by disabling other flow types so that the sum of maximum entries for all flow types per line card does not exceed 8000 entries. Hence RSVP-known flow type is set to zero in our example:

The maximum dynamic scale for any flow type should be configured such that all LPTS entries for that flow type are in hardware. One way to achieve that is to increase the dynamic scale. This may help avoid session flaps for NSR-enabled protocols like BGP and OSPF in case of triggers like RP fail overs.

```
Router#configure
Router(config)#lpts pifib hardware dynamic-flows location 0/1/CPU0
Router(config-pifib-flows-per-node)#flow bgp known max 1800
Router(config-pifib-flows-per-node)#flow rsvp known max 0
Router(config-pifib-flows-per-node)#commit
```

Running Configuration

```
Router#show running-config lpts pifib hardware dynamic-flows location 0/1/CPU0
lpts pifib hardware dynamic-flows location 0/1/CPU0
 flow bgp known max 1800
 flow rsvp known max 0
!
```

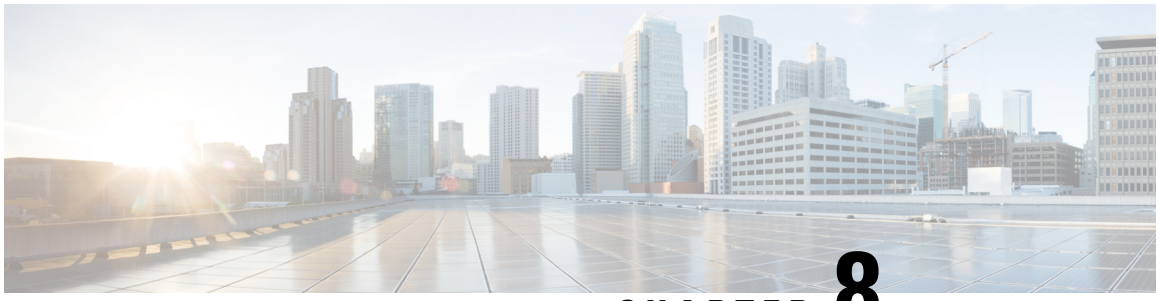
Verification

This show command displays dynamic flow statistics. You can see that the flow types BGP-known and ISIS-known are configured in the TCAM with newly configured maximum flow entry value. You can also see that the RSVP-known flow type is disabled:

```
Router#show lpts pifib dynamic-flows statistics location 0/1/CPU0

 Dynamic-flows Statistics:
 -------------------------
 (C - Configurable, T - TRUE, F - FALSE, * - Configured)
 Def_Max  - Default Max Limit
 Conf_Max - Configured Max Limit
 HWCnt    - Hardware Entries Count
 ActLimit - Actual Max Limit
 SWCnt    - Software Entries Count
 P, (+)   - Pending Software Entries


  FLOW-TYPE           C  Def_Max Conf_Max     HWCnt/ActLimit      SWCnt  P
 -------------------- -- ------- --------   -------/--------    -------- -
 Fragment             F        2       --         2/2                2
 OSPF-mc-known        T      600       --         2/600              2
 OSPF-mc-default      F        4       --         4/4                4
 OSPF-uc-known        T      300       --         1/300              1
 OSPF-uc-default      F        0       --         0/0                1  +
 BFD-default          F        2       --         2/2                2
 BFD-MP-known         T       40       --         1/40               0
 BGP-known            T*    2400     1800         6/900              6
 BGP-cfg-peer         T      900       --         0/900              0
 BGP-default          F        4       --         4/4                4
 PIM-mcast-default    F       40       --         0/40               0
 PIM-mcast-known      T      300       --         0/300              0
 PIM-ucast            F       40       --         2/40               2
 IGMP                 T     1200       --         0/1200             0
 ICMP-local           F        4       --         4/4                4
 ICMP-control         F        5       --         5/5                5
 LDP-TCP-known        T      300       --         0/300              0
 LDP-TCP-cfg-peer     T      300       --         0/300              0
 LDP-TCP-default      F       40       --         0/40               0
 LDP-UDP              T      300       --         0/300              0
 All-routers          T      300       --         0/300              0
 RSVP-default         F        4       --         1/4                1
 RSVP-known           T*     300        0         0/0                1  +
 SNMP                 T      300       --         8/300              8
 SSH-known            T       40       --         0/40               0
 SSH-default          T        1       --         1/1                2  +
 HTTP-known           T       40       --         0/40               0
 SHTTP-known          T       40       --         0/40               0
 TELNET-known         T       40       --         0/40               0
 TELNET-default       T        1       --         1/1                1
 UDP-known            T        0       --         0/0                0
 UDP-default          F        2       --         2/2                2
 TCP-known            T       40       --         0/40               0
 TCP-default          F        2       --         2/2                2
```

```
 Raw-default           F         2       --          2/2                 2
 GRE                   F         4       --          0/4                 0
 VRRP                  T       150       --          0/150               0
 DNS                   T        40       --          0/40                0
 NTP-known             T        40       --          0/40                0
 DHCPv4                T        40       --          0/40                0
 DHCPv6                T        40       --          0/40                0
 TPA                   T      1000       --          0/1000              0
 PM-TWAMP              T        10       --          0/10                0
-----------------------------------------------------
 Active TCAM Usage : 13421/16000 [Platform MAX: 16000]
 HWCnt/SWCnt       : 65/88
-----------------------------------------------------
```

In the above show command output, the last column **P** specifies the pending software flow entries for the flow type.



CHAPTER 8

Configuring Transports

Information About Configuring NSR, TCP, UDP Transports

To configure NSR, TCP, UDP, and RAW transports, you must understand the following concepts:

NSR Overview

Nonstop Routing (NSR) is provided for Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), and Label Distribution Protocol (LDP) protocols for the following events:

- Route Processor (RP) failover
- Process 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).

NSR is a method to achieve High Availability (HA) of the routing protocols. TCP connections and the routing protocol sessions are migrated from the active RP to standby RP after the RP failover without letting the peers know about the failover. Currently, the sessions terminate and the protocols running on the standby RP reestablish the sessions after the standby RP goes active. Graceful Restart (GR) extensions are used in place of NSR to prevent traffic loss during an RP failover but GR has several drawbacks.

You can use the **nsr process-failures switchover** command to let the RP failover be used as a recovery action when the active TCP or active LDP restarts. When standby TCP or LDP restarts, only the NSR capability is lost till the standby instances come up and the sessions are resynchronized but the sessions do not go down. In the case of the process failure of an active OSPF, a fault-management policy is used.

For more information, refer to chapter *Implementing OSPF Routing Configuration Guide for Cisco 8000 Series Routers*.

TCP Overview

TCP is a connection-oriented protocol that specifies the format of data and acknowledgments that two computer systems exchange to transfer data. TCP also specifies the procedures the computers use to ensure that the data

arrives correctly. TCP allows multiple applications on a system to communicate concurrently, because it handles all demultiplexing of the incoming traffic among the application programs.

UDP Overview

The User Datagram Protocol (UDP) is a connectionless transport-layer protocol that belongs to the IP family. UDP is the transport protocol for several well-known application-layer protocols, including Network File System (NFS), Simple Network Management Protocol (SNMP), Domain Name System (DNS), and TFTP.

Any IP protocol other than TCP and UDP is known as a RAW protocol.

For most sites, the default settings for the TCP, UDP, and RAW transports need not be changed.

Prerequisites for Configuring NSR, TCP, UDP, Transports

The following prerequisites are required to implement NSR, TCP, UDP, Transports:

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

Configuring Failover as a Recovery Action for NSR

When the active TCP or the NSR client of the active TCP 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 8000 Series Routers*.

For information on how to configure NSR on a per-process level for each process, refer to the *Routing Configuration Guide for Cisco 8000 Series Routers*.

Configuration Example

Configure failover as a recovery action for active instances to switch over to a standby to maintain nonstop routing.

```
Router#configure
Router(config)#nsr process-failures switchover
Router(config)#commit
```

Running Configuration

```
Router#show running-configuration nsr process-failures switchover
nsr process-failures switchover
```

Associated Commands

- nsr process-failures switchover