IP Addresses and Services Configuration Guide for Cisco NCS 5000 Series Routers, IOS XR Release 24.1.1

First Published: 2023-11-30

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

CONTENTS

Preface

This preface contains these sections:

Changes to This Document

Table 1: Changes to This Document

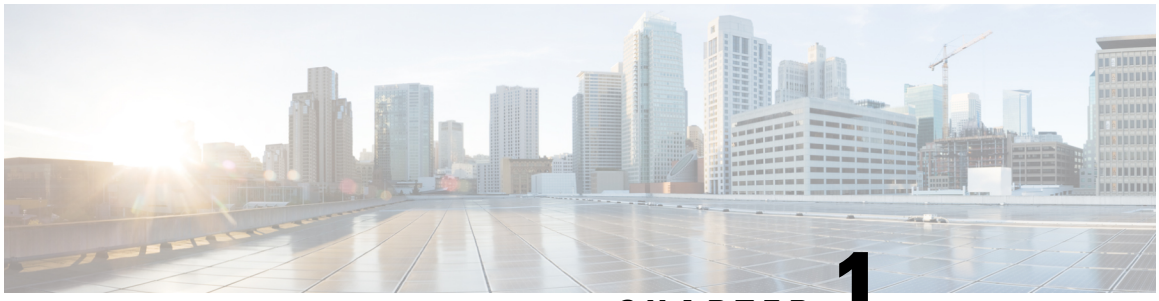
Date	Change Summary
September 2024	Republished for Release 24.3.1
June 2024	Republished for Release 24.2.1
February 2024	Initial release of this document

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business 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 DevNet.
- 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 NCS 5000 Series Routers*, and tells you where they are documented.

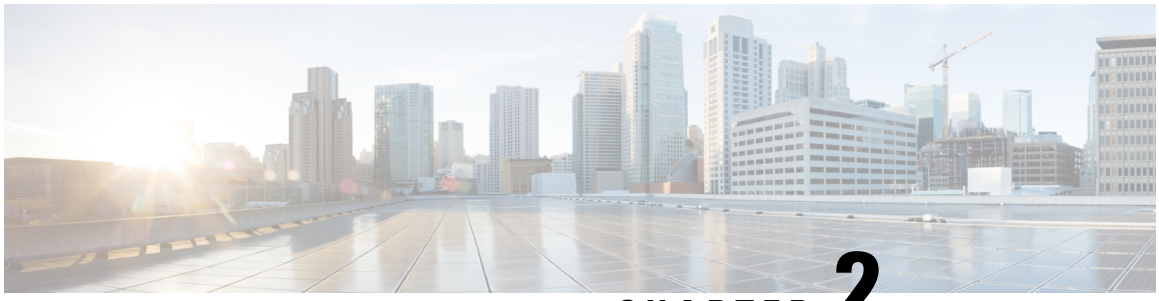
IP Addresses and Services Features Added or Modified in IOS XR Release 24.x.x

This section describes the new and changed IP addresses features for Cisco IOS XR.

IP Addresses Features Added or Modified in IOS XR Release 24.x.x

Table 2: New and Changed Features

Feature	Description	Changed in Release	Where Documented
None	No new features introduced	Not applicable	Not applicable



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 TenGigE 0/0/0/1 conflicts with TenGigE 0/0/0/2, then the IPv4 protocol on TenGigE 0/0/0/2, is brought down and IPv4 remains active on TenGigE 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, I*nternet Protocol, Version 6 (IPv6) Specification* issued by the Internet Engineering Task Force (IETF). Further RFCs describe the architecture and services supported by IPv6.

How to Implement Network Stack IPv4 and IPv6

This section contains the following procedures:

Configuring IPv4 Addressing

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

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

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

Configuration Example

An IPv4 address of 192.168.1.27 and a network mask of "/8" is assigned to the TenGigE interface-0/0/0/0.

Note The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means the corresponding address bit belongs to the network address. The network mask can be indicated as a slash (/) and a number- a prefix length. The prefix length is a decimal value that indicates how many of the high-order contiguous bits of the address comprise the prefix (the network portion of the address). A slash must precede the decimal value, and there is no space between the IP address and the slash.

```
Router#configure
Router(config)#interface TenGigE 0/0/0/0
Router(config-if)#ipv4 address 192.168.1.27/8
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface TenGigE 0/0/0/0
interface TenGigE0/0/0/0
 ipv4 address 192.168.1.27 255.0.0.0
!
```

Verification

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

```
Router# show ipv4 interface tenGigE 0/0/0/0
TenGigE0/0/0/0 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.168.1.27/8
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Multicast reserved groups joined: 224.0.0.2 224.0.0.1
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
```

Associated Commands

- ipv4 address
- show ipv4 interface

Configuring IPv6 Addressing

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

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

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

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

IPv6 Multicast Groups

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

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

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

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

Configuration Example

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

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

Running Configuration

```
Router# show running-config interface tenGigE 0/0/0/0
interface TenGigE0/0/0/0
 ipv4 address 192.168.1.27 255.0.0.0
 ipv4 address 1.0.0.1 255.255.255.0 secondary
 ipv4 address 2.0.0.1 255.255.255.0 secondary
 ipv6 address 2001:db8:0:1::1/64
!
```

Verification

Verify that the 10-Gigabit Ethernet interface is active and IPv6 is enabled.

```
Router#show ipv6 interface tenGigE 0/0/0/0
TenGigE0/0/0/0 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
  Global unicast address(es):
    2001:db8:0:1::1, subnet is 2001:db8:0:1::/64
  Joined group address(es): ff02::1:ff00:1 ff02::1:ffa6:1c75 ff02::2
      ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Associated Commands

- ipv6 address
- show ipv6 interface

Configuration Example

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

Running Configuration

```
Router# show running-config interface tenGigE 0/0/0/0
interface TenGigE0/0/0/0
 ipv4 address 192.168.1.27 255.0.0.0
 ipv4 address 1.0.0.1 255.255.255.0 secondary
 ipv4 address 2.0.0.1 255.255.255.0 secondary
 ipv6 address 2001:db8:0:1::/64 eui-64
!
```

Verification

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

```
Router#show ipv6 interface tenGigE 0/0/0/0
TenGigE0/0/0/0 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
  Global unicast address(es):
    2001:db8:0:1:c672:95ff:fea6:1c75, subnet is 2001:db8:0:1::/64
  Joined group address(es): ff02::1:ffa6:1c75 ff02::2 ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Associated Commands

- ipv6 address
- show ipv6 interface

Configuration Example

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

Running Configuration

```
Router# show running-config interface tenGigE 0/0/0/0
interface TenGigE0/0/0/0
 ipv6 address fe80::260:3eff:fe11:6770 link-local
!
```

Verification

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

```
Router#show ipv6 interface tenGigE 0/0/0/0
TenGigE0/0/0/0 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::260:3eff:fe11:6770
  Global unicast address(es):
    2001:db8:0:1:260:3eff:fe11:6770, subnet is 2001:db8:0:1::/64
  Joined group address(es): ff02::1:ff11:6770 ff02::2 ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
```

```
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Associated Commands

- ipv6 address
- show ipv6 interface

Configuration Example

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

```
Router#configure
Router(config)#interface tenGigE 0/0/0/0
Router(config-if)#ipv6 enable
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface tenGigE 0/0/0/0
interface TenGigE0/0/0/0
ipv6 enable
!
```

Verification

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

```
Router#show ipv6 interface tenGigE 0/0/0/0
TenGigE0/0/0/0 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
  No global unicast address is configured
  Joined group address(es): ff02::1:ffa6:1c75 ff02::2 ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
```

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

Associated Commands

- ipv6 address
- show ipv6 interface

Assigning Multiple IP Addresses to Network Interfaces

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

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

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

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

Configuration Example

A secondary IPv4 address of 192.168.1.27 is assigned to the TenGigE interface-0/0/0/1.

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

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

Running Configuration

```
Router#show running-config interface TenGigE 0/0/0/1
interface TenGigE0/0/0/1
```

```
 ipv4 address 192.168.1.27 255.255.255.0 secondary
!
```

Verification

```
Router#show ipv4 interface tenGigE 0/0/0/1
TenGigE0/0/0/1 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is unassigned
  Secondary address 192.168.1.27/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Multicast reserved groups joined: 224.0.0.2 224.0.0.1
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
```

Associated Commands

- ipv4 address
- show ipv4 interface

Configuring IPv4 and IPv6 Protocol Stacks

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

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

Configuration Example

An IPv4 address of 192.168.99.1 and an IPv6 address of 2001:0DB8:c18:1::3/64 is configured on the TenGigE interface-0/0/0/0.

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

Running Configuration

```
Router# show running-config interface tenGigE 0/0/0/0
interface TenGigE0/0/0/0
 ipv4 address 192.168.99.1 255.255.255.0
 ipv6 address 2001:db8:c18:1::3/64
!
```

Verification

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

```
Router#show ipv4 inter tenGigE 0/0/0/0
TenGigE0/0/0/0 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.168.99.1/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Multicast reserved groups joined: 224.0.0.2 224.0.0.1
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000

Router#show ipv6 inter tenGigE 0/0/0/0
TenGigE0/0/0/0 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
  Global unicast address(es):
    2001:db8:c18:1::3, subnet is 2001:db8:c18:1::/64
  Joined group address(es): ff02::1:ff00:3 ff02::1:ffa6:1c75 ff02::2
      ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 1000000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```

Associated Commands

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

Enabling IPv4 Processing on an Unnumbered Interface

This section describes the process of enabling an IPv4 point-to-point interface without assigning an explicit IP address to the interface. Whenever the unnumbered interface generates a packet (for example, for a routing update), it uses the address of the interface you specified as the source address of the IP packet. It also uses

the specified interface address in determining which routing processes are sending updates over the unnumbered interface. Restrictions are as follows:

- Interfaces using High-Level Data Link Control (HDLC), PPP, and Frame Relay encapsulations can be unnumbered. Serial interfaces using Frame Relay encapsulation can also be unnumbered, but the interface must be a point-to-point sub-interface.
- You cannot use the **ping** EXEC command to determine whether the interface is up, because the interface has no IP address. The Simple Network Management Protocol (SNMP) can be used to remotely monitor interface status.
- You cannot support IP security options on an unnumbered interface.

If you are configuring Intermediate System-to-Intermediate System (IS-IS) across a serial line, you should configure the serial interfaces as unnumbered, which allows you to conform with RFC 1195, which states that IP addresses are not required on each interface.

Configuration Example

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

```
Router#configure
Router(config)#interface TenGigE 0/0/0/1
Router(config-if)#ipv4 unnumbered loopback 0
Router(config-if)#commit
```

Running Configuration

```
Router#show running-config interface tenGigE 0/0/0/1
interface TenGigE0/0/0/1
 ipv4 point-to-point
 ipv4 unnumbered Loopback0
!
```

Verification

```
Router#show int tenGigE 0/0/0/1
TenGigE0/0/0/1 is up, line protocol is up
  Interface state transitions: 5
  Hardware is TenGigE, address is 00e2.2a33.445b (bia 00e2.2a33.445b)
  Layer 1 Transport Mode is LAN
  Internet address is 10.0.0.2/32
  MTU 1514 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
     reliability 255/255, txload 194/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 10000Mb/s, link type is force-up
  output flow control is off, input flow control is off
  Carrier delay (up) is 10 msec
  loopback not set,
  Last link flapped 01:38:49
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters 02:34:16
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 7647051000 bits/sec, 12254894 packets/sec
     1061401410 packets input, 82789675614 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 5 broadcast packets, 19429 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
```

```
     76895885948 packets output, 6192569128048 bytes, 0 total output drops
     Output 7 broadcast packets, 18916 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     2 carrier transitions
```

```
Router #show run int lo 0
interface Loopback0
 ipv4 address 10.0.0.2 255.255.255.255
```

Associated Commands

- ipv4 unnumbered
- show ipv4 interface

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

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...
ipv4 conflict-policy static
```

Verification

```
Router#show arm conflicts
F Forced down
| Down interface & addr           Up interface & addr    VRF
F Te0/0/0/19 10.0.0.2/24    Te0/0/0/1 10.0.0.2/24  default
Forced down interface       Up interface           VRF
```

Associated Commands

- ipv4 conflict-policy
- ipv6 conflict-policy

Longest Prefix Address Conflict Resolution

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

Configuration Example

Configures longest prefix address conflict resolution.

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

Running Configuration

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

Verification

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

F Te0/0/0/19 192.85.1.2/24   HundredGigE0/0/0/1  192.85.1.1/24 default
```

```
Forced down interface         Up interface                  VRF
```

Associated Commands

- ipv4 conflict-policy
- ipv6 conflict-policy

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 Te0/0/0/19 192.85.1.2/24   HundredGigE0/0/0/1  192.85.1.1/24 default

Forced down interface         Up interface                  VRF
```

Associated Commands

- ipv4 conflict-policy
- ipv6 conflict-policy

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 192.168.1.27 that has a route tag of 20 to the TenGigE interface 0/0/0/1.

```
Router# configure
Router(config)#interface TenGigE 0/0/0/1
Router(config-if)#ipv4 address 192.168.1.27 255.0.0.0
Router(config-if)#route-tag 20
Router(config)#commit
```

Running Configuration

```
Router#show running-config interface tenGigE 0/0/0/1
interface TenGigE0/0/0/1
 ipv4 address 192.168.1.27 255.0.0.0 route-tag 20
!
```

Verification

Verify the parameters of the route.

```
Router#show route 192.168.1.27
Routing entry for 192.168.1.27
  Known via "local", distance 0, metric 0 (connected)
  Tag 20
Routing Descriptor Blocks
    directly connected, via Tengige 0/0/0/1
      Route metric is 0
  No advertising protos.
```

Associated Commands

- route-tag

Larger IPv6 Address Space

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

IPv6 Address Formats

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

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

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

It is common for IPv6 addresses to contain successive hexadecimal fields of zeros. To make IPv6 addresses less cumbersome, two colons (::) can be used to compress successive hexadecimal fields of zeros at the beginning, middle, or end of an IPv6 address. (The colons represent successive hexadecimal fields of zeros.) Table 3: Compressed IPv6 Address Formats, on page 18 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 18 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 18 indicates the absence of an IPv6 address. For example, a newly initialized node on an IPv6 network may use the unspecified address as the source address in its packets until it receives its IPv6 address.

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

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

IPv6 Address Type: Unicast

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

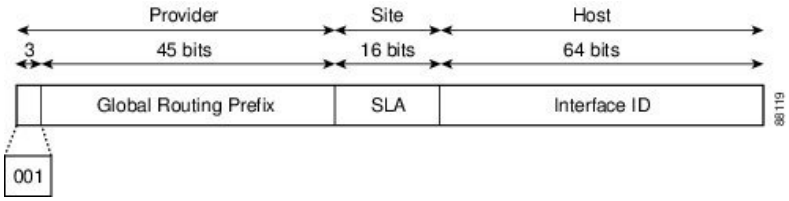
- Global aggregatable address
- Site-local address (proposal to remove by IETF)
- Link-local address
- IPv4-compatible IPv6 address

Aggregatable Global Address

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

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

Figure 1: Aggregatable Global Address Format



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

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

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

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

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

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

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

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

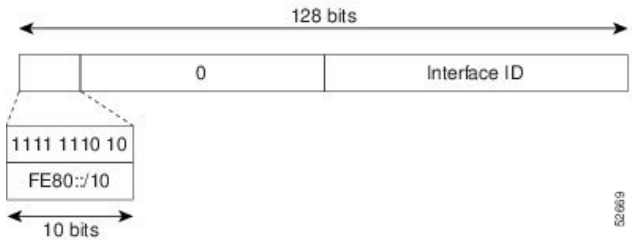
1. The router is queried for MAC addresses (from the pool of MAC addresses in the router).
2. If no MAC address is available, the serial number of the Route Processor (RP) or line card (LC) is used to form the link-local address.

Link-Local Address

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

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

Figure 2: Link-Local Address Format

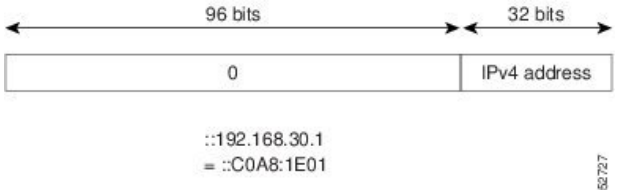


IPv4-Compatible IPv6 Address

An IPv4-compatible IPv6 address is an IPv6 unicast address that has zeros in the high-order 96 bits of the address and an IPv4 address in the low-order 32 bits of the address. The format of an IPv4-compatible IPv6 address is 0:0:0:0:0:0:A.B.C.D or ::A.B.C.D. The entire 128-bit IPv4-compatible IPv6 address is used as the

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

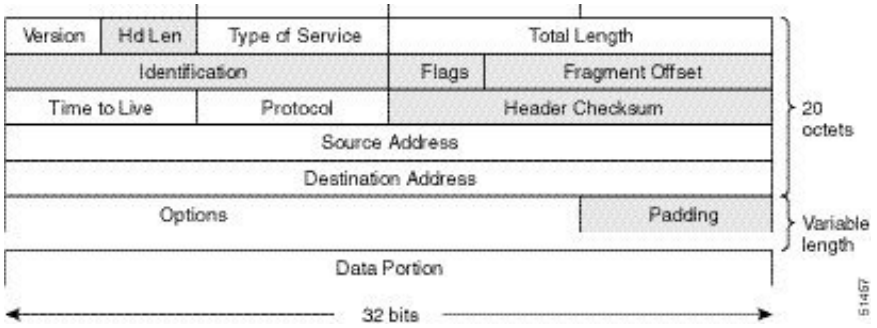
Figure 3: IPv4-Compatible IPv6 Address Format



Simplified IPv6 Packet Header

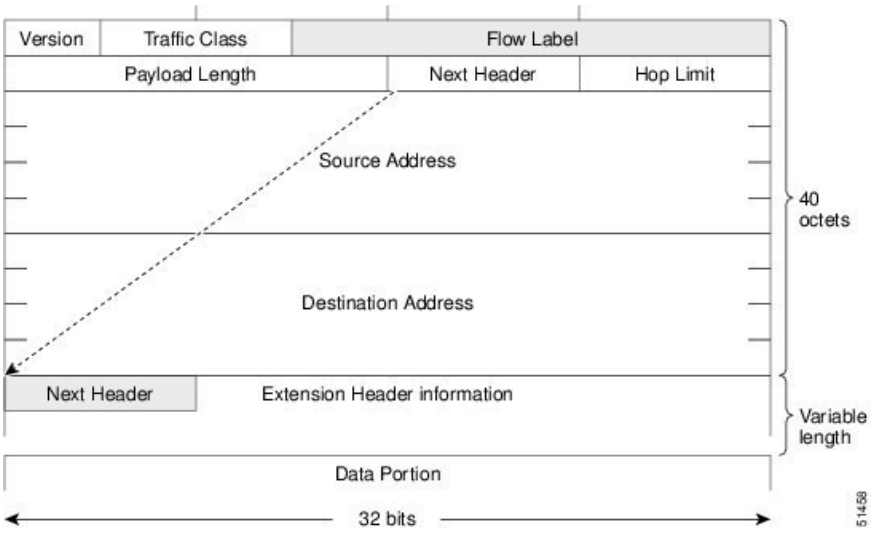
The basic IPv4 packet header has 12 fields with a total size of 20 octets (160 bits). The 12 fields may be followed by an Options field, which is followed by a data portion that is usually the transport-layer packet. The variable length of the Options field adds to the total size of the IPv4 packet header.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

Figure 4: IPv4 Packet Header Format



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

Figure 5: IPv6 Packet Header Format



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

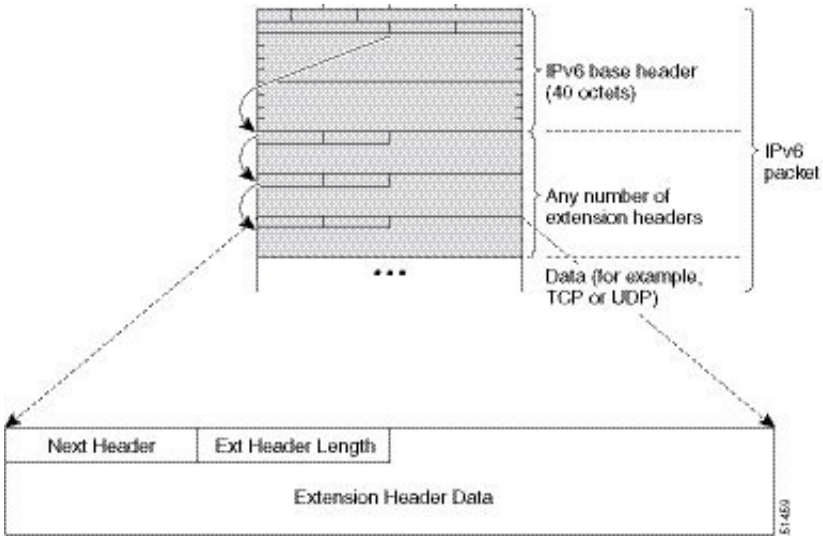
Table 4: Basic IPv6 Packet Header Fields

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

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

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

Figure 6: IPv6 Extension Header Format



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

Table 5: IPv6 Extension Header Types

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

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

Path MTU Discovery for IPv6

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

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

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

IPv6 Neighbor Discovery

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

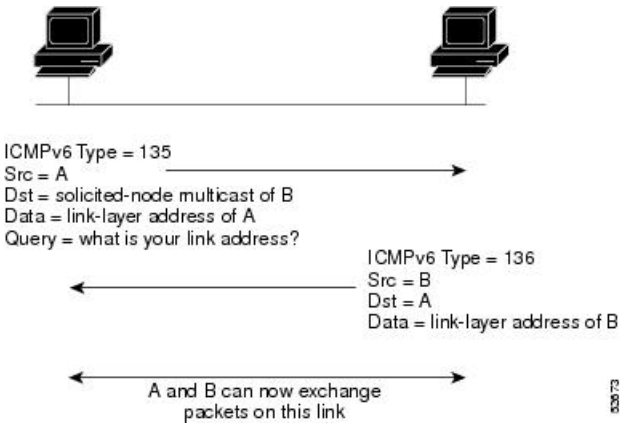
As all incoming control traffic goes through LPTS policer, if the ND packets come in a burst they are policed according to the configuration. For more details on LPTS, see LPTS Overview, on page 95.

IPv6 Neighbor Solicitation Message

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

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

Figure 7: IPv6 Neighbor Discovery—Neighbor Solicitation Message



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

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

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

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

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

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

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

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

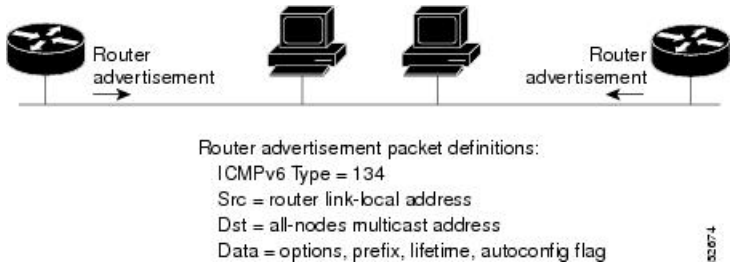
Neighbor solicitation messages are also used in the stateless autoconfiguration process to verify the uniqueness of unicast IPv6 addresses before the addresses are assigned to an interface. Duplicate address detection is performed first on a new, link-local IPv6 address before the address is assigned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

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

IPv6 Router Advertisement Message

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

Figure 8: IPv6 Neighbor Discovery—Router Advertisement Message



Router advertisement messages typically include the following information:

- One or more onlink IPv6 prefixes that nodes on the local link can use to automatically configure their IPv6 addresses

- Lifetime information for each prefix included in the advertisement
- Sets of flags that indicate the type of autoconfiguration (stateless or statefull) that can be completed
- Default router information (whether the router sending the advertisement should be used as a default router and, if so, the amount of time, in seconds, that the router should be used as a default router)
- Additional information for hosts, such as the hop limit and MTU a host should use in packets that it originates

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

The following router advertisement message parameters can be configured:

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

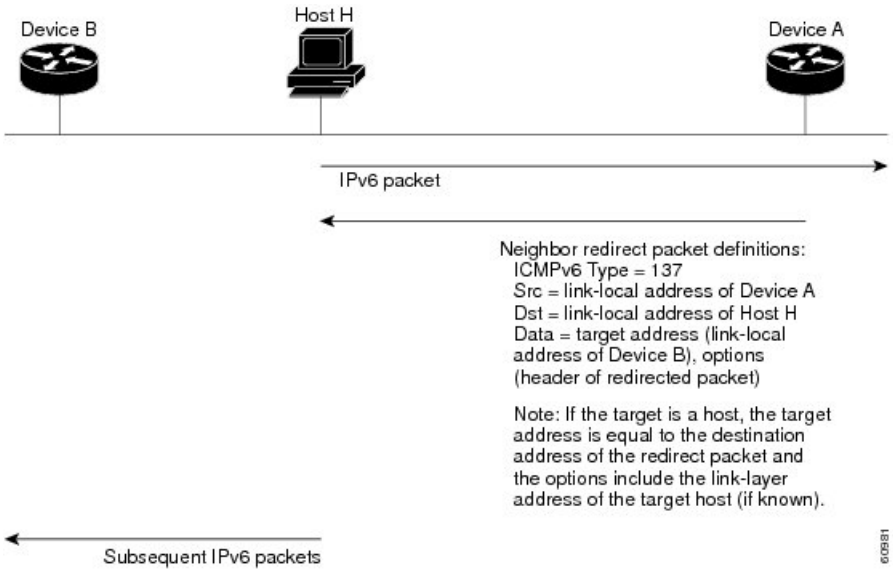
The configured parameters are specific to an interface. The sending of router advertisement messages (with default values) is automatically enabled on Ethernet and FDDI interfaces. For other interface types, the sending of router advertisement messages must be manually configured by using the **no ipv6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

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

IPv6 Neighbor Redirect Message

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

Figure 9: IPv6 Neighbor Discovery—Neighbor Redirect Message



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

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

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

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

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

Address Repository Manager

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

Address Conflict Resolution

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

Conflict Database

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

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

address 1: 10.1.1.1/16

address 2: 10.2.1.1/16

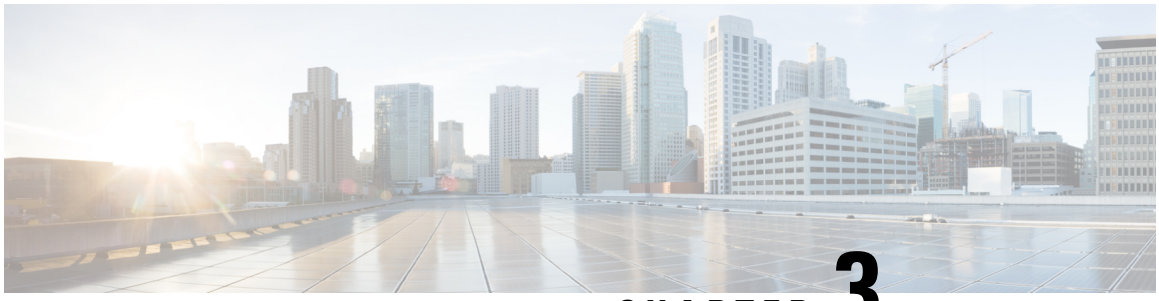
address 3: 10.3.1.1/16

address 4: 10.4.1.1/8

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

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

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



CHAPTER 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.

As all incoming control traffic goes through LPTS policer, if the ARP packets come in a burst they are policed according to the configuration. For more details on LPTS, see LPTS Overview, on page 95.

For more details on ARP, see Information About Configuring ARP, on page 38

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 33

Restrictions

The following restrictions apply to configuring ARP :

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

ARP Cache Entries

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

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

Defining a Static ARP Cache Entry

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

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

Configuration Example

A cache entry is created to establish connection between an IP address **20.0.0.2** and the MAC address **20.0.2**. 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 20.0.0.2 20.0.2 arpA alias
```

Running Configuration

```
Router# show running-config interface tenGigE 0/0/0/0
arp 1.0.0.2 1.0.2 arpa
arp vrf default 1.0.0.2 0001.0000.0002 ARPA
```

Verification

Verify that the State is static for proper functioning:

```
Router# show arp 20.0.0.2
------------------------------------------------------------------------------
0/RP0/CPU0
------------------------------------------------------------------------------
Address         Age        Hardware Addr   State        Type  Interface
20.0.0.2        -          0020.0000.0002  Static ARPA  TenGigE0/0/0/0
```

Related Topics

Defining a Static ARP Cache Entry, on page 32

Associated Commands

- arp
- show arp

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 TenGigE interface-**0/0/0/0**:

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

Running Configuration

```
Router# show running-config interface tenGigE 0/0/0/0
interface TenGigE0/0/0/0
 mtu 4000
 ipv4 address 1.0.0.1 255.255.255.0
 proxy-arp
```

```
!
!
```

Verification

Verify that proxy ARP is configured and enabled:

```
Router# show arp idb tenGigE 0/0/0/0 location 0/RP0/CPU0
  TenGigE0/0/0/0 (0x08000038):
  IPv4 address 1.0.0.1, Vrf ID 0x60000000
  VRF Name default
  Dynamic learning: Enable
  Dynamic entry timeout: 14400 secs
  Purge delay: off
  IPv4 caps added (state up)
  MPLS caps not added
  Interface not virtual, not client fwd ref,
  Proxy arp is configured, is enabled
  Local Proxy arp not configured
  Packet IO layer is NetIO
  Srg Role : DEFAULT
  Idb Flag : 262332
  IDB is Complete
```

Related Topics

- Enabling Proxy ARP, on page 33

Associated Commands

- proxy-arp

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 TenGigE interface-**0/0/0/0**

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

Running Configuration

```
Router# show running-config interface tenGigE 0/0/0/0
interface TenGigE0/0/0/0
ipv4 address 1.0.0.1 255.255.255.0
local-proxy-arp
!
```

Verification

Verify that local proxy ARP is configured:

```
Router# show arp idb tenGigE 0/0/0/0 location 0/RP0/CPU0
TenGigE0/0/0/0 (0x08000038):
  IPv4 address 1.0.0.1, Vrf ID 0x60000000
  VRF Name default
  Dynamic learning: Enable
  Dynamic entry timeout: 14400 secs
  Purge delay: off
  IPv4 caps added (state up)
  MPLS caps not added
  Interface not virtual, not client fwd ref,
  Proxy arp 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

Configure Learning of Local ARP Entries

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

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

1. Enter the interface configuration mode.

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

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

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

3. Configure local ARP learning on the interface.

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

4. Enable the interface and commit your configuration.

   ```
   Router(config-if)# no shut
   Router(config-if)# commit
   RP/0/0/CPU0:Dec 12 13:41:16.580 : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : interface
   GigabitEthernet 0/0/0/1, changed state to Down
   RP/0/0/CPU0:Dec 12 13:41:16.683 : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : interface
   GigabitEthernet 0/0/0/1 changed state to Up
   ```

5. Confirm your configuration.

   ```
   Router(config-if)# show running-configuration
   ..
   Building configuration...
   !! IOS XR Configuration 0.0.0
   !! Last configuration change at Mon Dec 12 13:41:16 2016
   !interface GigabitEthernet 0/0/0/1
    ipv4 address 12.1.3.4 255.255.255.0
    arp learning local
   !
   ```

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

```
Router(config-if)# do show arp idb gigabitEthernet 0/0/0/1 location 0/0/CPU0
Thu Dec 15 10:00:11.733 IST

GigabitEthernet 0/0/0/1 (0x00000040):
  IPv4 address 12.1.3.4, Vrf ID 0x60000000
  VRF Name default
  Dynamic learning: Local
  Dynamic entry timeout: 14400 secs
  Purge delay: off
  IPv4 caps added (state up)
  MPLS caps not added
  Interface not virtual, not client fwd ref,
  Proxy arp not configured, not enabled
  Local Proxy arp not configured
  Packet IO layer is NetIO
  Srg Role : DEFAULT
  Idb Flag : 2146444
  IDB is Complete
```

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

```
Router(config-if)# do show arp idb gigabitEthernet 0/0/0/1 location 0/0/CPU0
Thu Dec 15 10:13:28.964 IST

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

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

  IP Packet drop count for GigabitEthernet0_0_0_1: 0
```

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.

Enabling DAGR

Configuration Example

DAGR is enabled on the TenGigE interface 0/0/0/10 with the peer router having ipv4 address of 5.10.10.1. The failover attributes set are priority-timeout, route distance, route metric, and timers query.

```
Router#configure
Router(config)#interface TenGigE 0/0/0/10
outer(config-if)#arp dagr
Router(config-if-dagr)#peer ipv4 5.10.10.1
Router(config-if-dagr-peer)#priority-timeout 1
Router(config-if-dagr-peer#route distance normal 1 priority 1
Router(config-if-dagr-peer#route metric normal 1 priority 1
Router(config-if-dagr-peer#timers query 2 standby 1
Router(config-if-dagr-peer#commit
```

Running Configuration

```
Router# show running-config interface tenGigE 0/0/0/10
mtu 9216
 ipv4 address 10.0.0.1 255.255.255.0
 arp dagr
  peer ipv4 1.0.0.1
   route metric normal 1 priority 1
   timers query 2 standby 1
   priority-timeout 1
   route distance normal 1 priority 1
  !
  peer ipv4 10.0.0.3
  !
 !
!
!
```

Verification

```
Router#show arp dagr
------------------------------------------------------------------------------
0/RP0/CPU0
------------------------------------------------------------------------------
Interface                 Virtual IP      State     Query-pd Dist Metr
TenGigE0/0/0/10           1.0.0.1         Standby   1        None None
TenGigE0/0/0/10           10.0.0.3        Query     1        None None
```

Related Topics

Direct Attached Gateway Redundancy, on page 36

Associated Commands

- arp dagr
- route distance
- route metric
- priority-timeout
- peer (DAGR)
- show arp dagr

Information About Configuring ARP

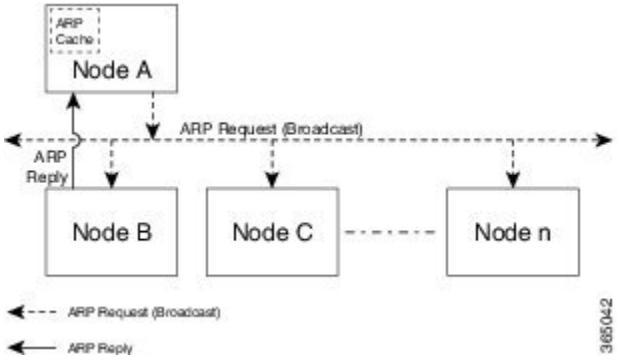
Addressing Resolution Overview

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

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

Address Resolution on a Single LAN

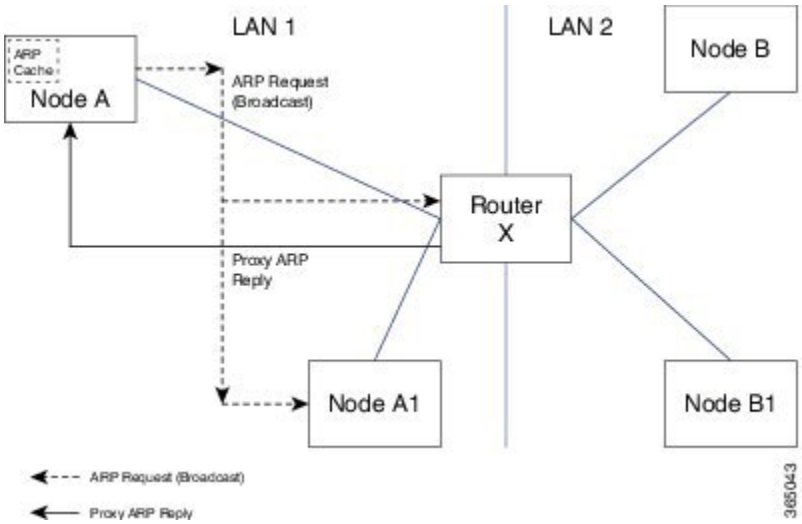
The following process describes address resolution when the source and destination devices are attached to the same LAN:



1. End System A (Node A) broadcasts an ARP request onto the LAN, attempting to learn the MAC address of End System B (Node B).
2. The broadcast is received and processed by all devices on the LAN, including End System B.
3. Only End System B replies to the ARP request. It sends an ARP reply containing its MAC address to End System A (Node A).
4. End System A (Node A) receives the reply and saves the MAC address of End System B in its ARP cache. (The ARP cache is where network addresses are associated with MAC addresses.)
5. Whenever End System A (Node A) needs to communicate with End System B, it checks the ARP cache, finds the MAC address of System B, and sends the frame directly, without needing to first use an ARP request.

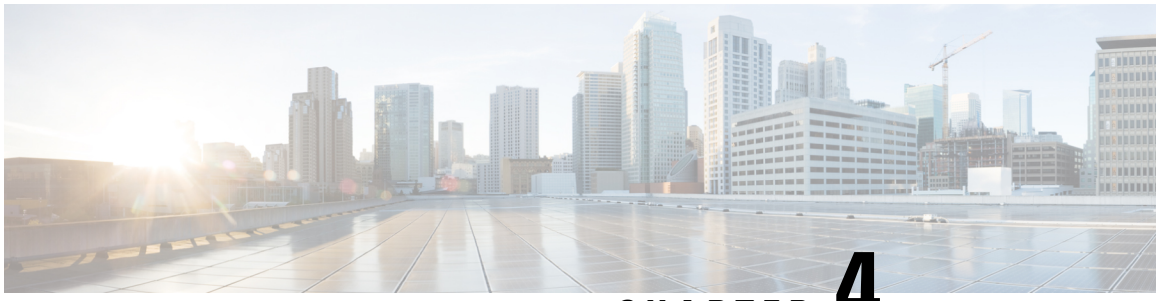
Address Resolution When Interconnected by a Router

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



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

3. Router X checks its routing table and finds that End System Z (Node B) is located on a different LAN.
4. Router X therefore acts as a proxy for End System Z (Node B). It replies to the ARP request from End System Y (Node A), sending an ARP reply containing its own MAC address as if it belonged to End System Z (Node B).
5. End System Y (Node A) receives the ARP reply and saves the MAC address of Router X in its ARP cache, in the entry for End System Z (Node B).
6. When End System Y (Node A) needs to communicate with End System Z (Node B), it checks the ARP cache, finds the MAC address of Router X, and sends the frame directly, without using ARP requests.
7. Router X receives the traffic from End System Y (Node A) and forwards it to End System Z (Node B) on the other LAN.



CHAPTER 4

Implementing DHCP

Implementing DHCP Relay Agent

Understanding DHCP

This topic provides an overview of DHCP related components.

DHCP Relay Agent

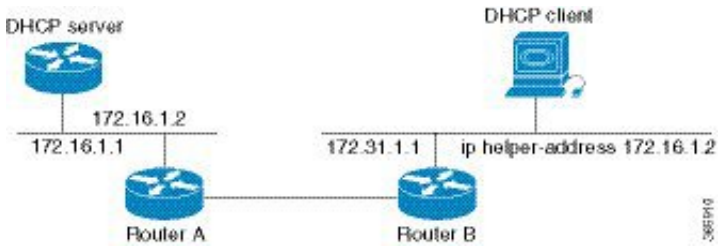
A DHCP relay agent is a host that forwards DHCP packets between clients and servers that do not reside on a shared physical subnet. Relay agent forwarding is distinct from the normal forwarding of an IP router where IP datagrams are switched between networks transparently.

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

If a client is on a network segment that does not include a server, a relay agent is needed on that network segment to ensure that DHCP packets reach the servers on another network segment. UDP broadcast packets are not forwarded, because most routers are not configured to forward broadcast traffic. You can configure a DHCP relay agent to forward DHCP packets to a remote server by configuring a DHCP relay profile and configure one or more helper addresses in it. You can assign the profile to an interface or a VRF.

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

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



DHCPv4 Server

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

DHCPv4 Client

The DHCP client functionality enables the router interfaces to dynamically acquire the IPv4 address using DHCP.

The DHCP provides configuration parameters to Internet hosts. DHCP consists of two components:

- a protocol to deliver host-specific configuration parameters from a DHCP server to a host.
- a mechanism to allocate network addresses to hosts.

DHCP is built on a client-server model, where designated DHCP server hosts allocate network addresses, and deliver configuration parameters to dynamically configured hosts.

A relay agent is required if the client and server are not on the same Layer 2 network. The relay agent usually runs on the router, and is required because the client device does not know its own IP address initially. The agent sends out a Layer 2 broadcast to find a server that has this information. The router relays these broadcasts to the DHCP server, and forwards the responses back to the correct Layer 2 address so that the correct device gets the correct configuration information.

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

Restrictions

- DHCP client can be enabled only on management interfaces.
- Either DHCP or static IP can be configured on an interface.

DHCP IPv4 Service-based Mode Selection

As part of DHCP IPv4 service based mode selection feature, a new mode called DHCP base is introduced. If an interface is configured in the DHCP base mode, then the DHCP selects the DHCP server mode to process

the client request by matching option 60 (class-identifier) value of the client request with the configured value under the DHCP base profile.

For example:

```
dhcp ipv4
profile DHCP_BASE base
  match option 60 41424355 profile DHCP_SERVER server
  default profile DEFAULT_PROFILE server
  relay information authenticate inserted
   !
profile DHCP_relay relay
  helper-address vrf default 10.10.10.1 giaddr 0.0.0.0
!
profile DHCP_SERVER server
  lease 1 0 0
  pool IP_POOL
!
profile DEFAULT_PROFILE server
  lease 1 0 0
  pool IP_POOL
!
!
interface gigabitEthernet 0/0/0/0 base profile DHCP_BASE
```

The pool is configured under server-profile-mode and server-profile-class-sub-mode. The class-based pool selection is always given priority over profile pool selection. The DHCPv4 server profile class sub-mode supports configuring DHCP options except few (0, 12, 50, 52, 53, 54, 58, 59, 61, 82, and 255).

Configuring and Enabling DHCP Relay Agent with DHCP MAC Address Verification

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

Configuration Example

```
Router# configure

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

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

Router(config-dhcpv4)# client-mac-mismatch action drop
/* Enables MAC address verification. If MAC address in the DHCPv4 protocol header does not
 match the L2 header source MAC address in the DHCPv4 relay profile,
 the frame is dropped  */

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

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

Router(config-dhcpv4-relay-profile)# relay information policy drop
/* (Optional) Configures the reforwarding policy for a DHCP relay agent;
```

```
that is, whether the relay agent will drop or keep (using the 'keep' keyword)
 the relay information. */

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

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

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

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

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

Running Configuration

Confirm your configuration.

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

DHCP MAC Address Verification

Use the following show command to check if DHCP MAC address is being verified on the router.

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

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

Enabling DHCP Relay Agent on an Interface

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

Configuration Example

The DHCP relay agent is disabled by default.

```
router#configure

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

router(config-dhcpv4)#interface HundredGigE 0/2/0/2 relay profile client
/* Attaches a relay profile to an interface.
To disable the DHCP relay on the interface, use the 'no interface HundredGigE 0/2/0/2 none'
 command. */

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

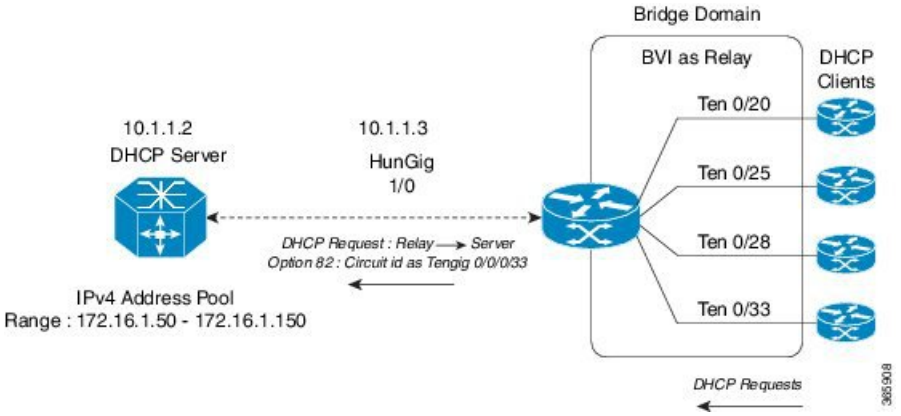
Running Configuration

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

Support for DHCP Option 82 on Bridge-Group Virtual Interface (BVI) Interface

DHCP option 82 provides additional security when DHCP is used to allocate network addresses. It enables the DHCP relay agent to prevent DHCP client requests from untrusted source. You can configure the relay agent to insert the Option 82 circuit ID in the DHCP packet before the relay agent sends the packet to the DHCP server. When the DHCP relay profile is attached to a bridge virtual interface (BVI), you can assign the name of the ingress Layer 2 interface as the value of Option 82 circuit ID. The DHCP packet that is sent from the relay agent to the server carries the packet's ingress Layer 2 interface name as Option 82 circuit ID.

Figure 11: DHCP Option 82 on BVI Interface



Configuration Example

```
router#configure

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

```
router(config-dhcpv4)#profile bvi1_profile relay
/* Enters the relay profile configuration mode. */

router(config-dhcpv4-relay-profile)#helper-address vrf default 10.1.1.2
/* Forwards UDP broadcasts, including DHCP. */

router(config-dhcpv4-relay-profile)#relay information option
/* Enables adding both circuit-id and remote-id to the DHCP packet */

router(config-dhcpv4-relay-profile)#interface BVI1 relay information option format-type
circuit-id format-string "%s" l2-interface
/*  Enables the DHCP relay agent to add the value of Option 82 Circuit ID field to the DHCP
 packet and assigns the ingress Layer 2 interface
as the value of Option 82 Circuit ID field. */

router(config-dhcpv4-relay-profile)# interface BVI1 relay profile bvi1_profile
/* Enables DHCP for IPV4 on a BVI interface and attaches the profile as the relay profile
for the BVI interface. */
```

Running Configuration

```
Router#show running-config dhcp ipv4

dhcp ipv4
 profile irb relay
  helper-address vrf default 10.1.1.2
  relay information option
 !
 interface BVI1 relay information option format-type circuit-id format-string "%s"
l2-interface
 interface BVI1 relay profile irb
```

Configuring DHCPv4 Server Profile on Bridge-Group Virtual Interface (BVI) Interface

DHCP server accepts address assignment requests and renewals and assigns the IP addresses from predefined groups of addresses contained within Distributed Address Pools (DAPS). DHCP server can also be configured to supply additional information to the requesting client. DHCP server can accept broadcasts from locally attached LAN segments or from DHCP requests that have been forwarded by other DHCP relay agents within the network.

DHCP IPv4 service-based mode selection

As part of DHCP IPv4 service based mode selection feature, a new mode called DHCP base is introduced. If an interface is configured in the DHCP base mode, then the DHCP selects the DHCP server mode to process the client request by matching option 60 (class-identifier) value of the client request with the configured value under the DHCP base profile.

The pool is configured under server-profile-mode and server-profile-class-sub-mode. The class-based pool selection is always given priority over profile pool selection. The DHCPv4 server profile class sub-mode supports configuring DHCP options except few (0, 12, 50, 52, 53, 54, 58, 59, 61, 82, and 255).

Configuration Example

```
Router#configure

Router(config)#dhcp ipv4

Router(config-dhcpv4)#profile bvi1_server server
/* Enters the server profile configuration mode. */

Router(config-dhcpv4-server-profile)#lease 0 0 0
/* Also use 'days minutes seconds' to specify lease for an IP address assigned from the
pool. */

Router(config-dhcpv4-server-profile)#bootfile http://10.1.1.1/auto/images/server_bvi.iso
/* Configures the boot file b1. */

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

Router(config-dhcpv4-server-profile)#broadcast-flag policy unicast-always
/*Broadcasts only BOOTREPLY packets if the DHCP IPv4 broadacst flag is set in the DHCP IPv4
 header */

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

Router(config-dhcpv4-server-profile-class)#match vrf default
/* Matches the class based on the VRF. */

Router(config-dhcpv4-server-profile-class)#match l2-interface TenGigE0/0/0/0
/* Matches the class based on the interface. */

Router(config-dhcpv4-server-profile-class)#bootfile http://10.1.1.1/auto/images/TENGClassA.iso
/* Configures the boot file b1. */

Router(config-dhcpv4-server-profile-class)#pool bvi_pool
/* Configures the DAPS pool name. */

Router(config-dhcpv4-server-profile-class)#option 66 ascii tftp://10.1.1.1/image.iso
/* Configures the DHCP option code. */

Router(config-dhcpv4-server-profile-class)#option 160 ascii http://10.1.1.1/image.iso
/* Configures the DHCP option code. */
```

Running Configuration

```
Router# show dhcp ipv4 server profile name BVI_Server
dhcp ipv4
 profile BVI_Server server
  lease 0 0 1
  bootfile http://10.1.1.1/auto/images/server_bvi.iso
  pool bvi_pool
  broadcast-flag policy UnicastAlways
  class ClassA
   match vrf default
   match l2-interface TenGigE0/0/0/0
   bootfile http://10.1.1.1/auto/images/server_TENGClassA.iso
   pool bvi_pool
   option 66 ascii tftp://10.1.1.1/image.iso
   option 160 ascii http://10.1.1.1/image.iso
```

```
!
```

Configuring Multiple Classes with a Pool

This section discusses configuring multiple classes with a pool.

Configuration Example

```
router#configure

router(config)#dhcp ipv4
/* Enables DHCP for IPv4 and enters DHCP IPv4 configuration mode. */

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

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

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

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

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

router(config-dhcpv4-server-profile-class)#exit
/* Exits the server profile class submode. */

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

router(config-dhcpv4-server-profile-class)#pool pool_B

/* Configures the pool name. */

router(config-dhcpv4-server-profile-class)#match vrf VRF1

/* The DHCP server selects a pool from a class by matching the options in the received
DISCOVER packet with the match command.
 none of the classes match, then pools configured under the profile mode are selected. The
 DHCP server requests DAPS to allocate
 an address from that pool. */

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

Running Configuration

```
Router#show running-config dhcp ipv4
dhcp ipv4
profile TEST server
pool POOL_TEST
class Class_A
pool pool_A
match option 60 hex abcd
exit
```

```
class Class_B
pool pool_B
match vrf VRF1
!
```

Configuring a Server Profile DAPS with Class Match Option

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

Configuration Example

```
router#configure

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Running Configuration

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

Configuring Server Profile without DAPS Pool Match Option

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

Configuration Example

```
router#configure

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

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

router(config-dhcpv4-server-profile)#dns-server ISP1.com
/* Configures the name of the DNS server or IP address. */

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

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

router(config-dhcpv4-server-profile)#dns-server ISP2.com
/* Configures the name of the DNS server or IP address. */

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

router(config-dhcpv4)#commit
```

Running Configuration

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

```
  dns-server ISP1.com

  exit
 profile ISP2 server
  dns-server ISP2.com

  exit
!
```

Configuring an Address Pool for Each ISP on DAPS

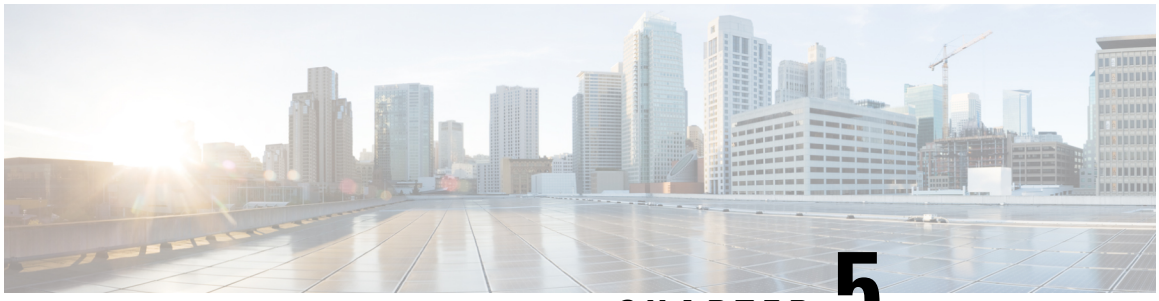
This section discusses configuring an address pool for each ISP on DAPS.

Configuration Example

```
router#configure

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

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

router(config-pool-ipv4)#exit

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

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

router(config-pool-ipv4)#exit

router(config-dhcpv4)#commit
```

Running Configuration

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



CHAPTER 5

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.

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.

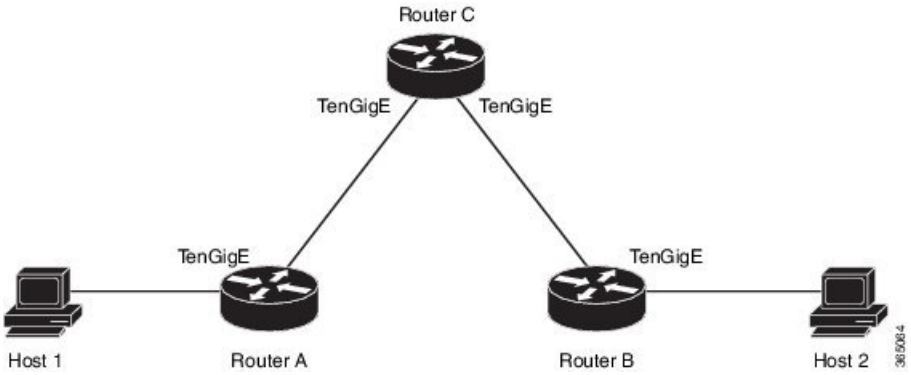
Checking Network Connectivity

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



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

To test the connectivity between Router A's TenGigE interface (10.0.0.2) and Router B's TenGigE interface (10.0.0.1 .), we use the extended **ping** command. With extended **ping**, we get the option to specify the source address of the **ping** packet.

Configuration Example

In this use case, the extended **ping** command verifies the IP connectivity between the two IP addresses Router A (10.0.0.2) and Router B (10.0.0.1) .

```
Router# ping 10.0.0.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.1, timeout is 2 seconds:
!!!!!
```

```
Success rate is 100 percent (5/5)
Router#!!!!!

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

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

Router# ping
Protocol [ipv4]:
Target IP address: 10.0.0.1
Repeat count [5]: 5
Datagram size [100]: 1000
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 1
Extended commands? [no]: no
Sweep range of sizes? [no]:
Type escape sequence to abort.
Sending 5, 1000-byte ICMP Echos to 10.0.0.1, timeout is 1 seconds:
!!!!!
Success rate is 100 percent (5/5)
Router#!!!!!
```

Associated Commands

- 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: 1.1.1.1
- 2: 2.2.2.2
- 3: 3.3.3.3

```
Router# ping bulk ipv4 input cli batch
*/You must hit the Enter button and then specify one destination address per line*/
Please enter input via CLI with one destination per line and when done Ctrl-D/(exit) to
initiate pings:
1: 1.1.1.1
2: 2.2.2.2
3: 3.3.3.3
4:
Starting pings...
Target IP address: 1.1.1.1
Repeat count [5]: 5
Datagram size [100]: 1
% A decimal number between 36 and 18024.
Datagram size [100]: 1
% A decimal number between 36 and 18024.
Datagram size [100]: 1000
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 10
```

```
Extended commands? [no]: no
Sweep range of sizes? [no]: q
% Please answer 'yes' or 'no'.
Sweep range of sizes? [no]: q
% Please answer 'yes' or 'no'.
Sweep range of sizes? [no]:
Type escape sequence to abort.
Sending 5, 1000-byte ICMP Echos to 1.1.1.1, vrf is default, timeout is 1 seconds:
!!!!!
Success rate is 100 percent (5/5),
Target IP address: 2.2.2.2
Repeat count [5]:
Datagram size [100]: q
% A decimal number between 36 and 18024.
Datagram size [100]:
Timeout in seconds [2]:
Interval in milliseconds [10]:
Extended commands? [no]:
Sweep range of sizes? [no]:
Sending 5, 100-byte ICMP Echos to 1.1.1.1, vrf is default, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5),
Target IP address:  3.3.3.3
Repeat count [5]: 4
Datagram size [100]: 100
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 10
Extended commands? [no]: no
Sweep range of sizes? [no]: no
Sending 4, 100-byte ICMP Echos to 1.1.1.1, vrf is default, timeout is 1 seconds:
!!!!!
Success rate is 100 percent (4/5),
```

Associated Commands

- 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 10.0.0.2 to 20.1.1.1:

```
Router# traceroute 20.1.1.1
Type escape sequence to abort.
Tracing the route to 20.1.1.1
 1  10.0.0.1 39 msec  *  3 msec

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

Router #traceroute
Protocol [ipv4]:
Target IP address: 20.1.1.1
Source address: 10.0.0.2
Numeric display? [no]:
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:

Type escape sequence to abort.
Tracing the route to 20.1.1.1
 1  10.0.0.1 3 msec  *  3 msec
```

Associated Commands

- 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 (192.168.7.18 and 10.2.0.2 192.168.7.33) with two hosts. The host names are host1 and host2.

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

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

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

Verification

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

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

Associated Commands

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

TFTP Server

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

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

Configuring a Router as a TFTP Server

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

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

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

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

Configuration Example

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

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

Running Configuration

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

Verification

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

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

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

Associated Commands

- tftp server
- show cinetd services

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 TenGigE 0/0/0/0
Router(config)# commit
```

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 TenGigE interface 0/0/0/0 as the source address for TFTP connections:

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

Verification

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

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

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

Associated Commands

- 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 "file1.txt" from the local directory to the remote directory and the "file2.txt" from vice-versa.

```
Router#scp /usr/file1.txt root@209.165.200.1:/root/file1.txt
** Transfers a file from a local directory to a remote directory**
Router# commit

Router#scp root@209.165.200.1:/root/file2.txt /usr/file2.txt
** Transfers a file from a remote directory to a local directory**
Router# commit
```

Associated Commands

- scp

Cisco inetd

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

Telnet

Enabling Telnet allows inbound Telnet connections into a networking device.

Configuration Example

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

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

Verification

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

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

Associated Commands

- telnet
- show cinetd services

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 1 to be the logging source interface for the default vrf.

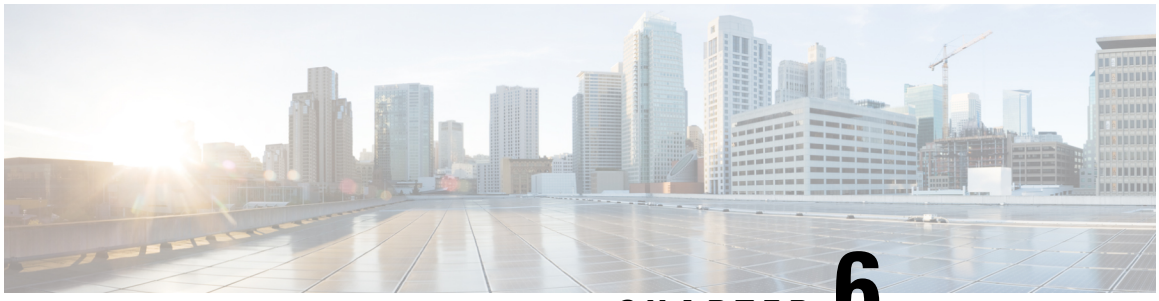
```
Router#configure
Router(config)#logging source-interface loopback 1 vrf default
Router(config)#commit
```

Running Configuration

```
Router#show running-config logging
logging source-interface Loopback1
```

Associated Commands

- logging source-interface



CHAPTER 6

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

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

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

Traditional ACLs are configured on internal TCAMs of routers.

Note

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

Purpose of IP Access Lists

- Filter incoming or outgoing packets on an interface.
- Filter packets for mirroring.
- Redirect traffic as required.

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

How an IP Access List Works

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

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

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

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

ACL Workflow

The following image illustrates the workflow of an ACL.

IP Access List Process and Rules

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

- The software tests the source or destination address or the protocol of each packet being filtered against the conditions in the access list, one condition (permit or deny statement) at a time.
- The packet is matched with ACE within and ACL in the order of the sequence number.
- If a packet and an access list statement match, the remaining statements in the list are skipped and the packet is permitted or denied as specified in the matched statement. The first entry that the packet matches determines whether the software permits or denies the packet. That is, after the first match, no subsequent entries are considered.
- If the access list denies the address or protocol, the software discards the packet.
- If no conditions match, the software drops the packet because each access list ends with an unwritten or implicit deny statement. That is, if the packet has not been permitted or denied by the time it was tested against each statement, it is denied.
- The access list should contain at least one permit statement or else all packets are denied.
- Because the software stops testing conditions after the first match, the order of the conditions is critical. The same permit or deny statements specified in a different order could result in a packet being passed under one circumstance and denied in another circumstance.
- Only one access list per interface, per protocol, per direction is allowed.

- Inbound access lists process packets arriving at the router. Incoming packets are processed before being routed to an outbound interface. An inbound access list is efficient because it saves the overhead of routing lookups if the packet is to be discarded because it is denied by the filtering tests. If the packet is permitted by the tests, it is then processed for routing. For inbound lists, permit means continue to process the packet after receiving it on an inbound interface; **deny** means to discard the packet.
- Outbound access lists process packets before they leave the router. Incoming packets are routed to the outbound interface and then processed through the outbound access list. For outbound lists, permit means send it to the output buffer; deny means discard the packet.
- An access list cannot be removed if that access list is being applied by an access group in use. To remove an access list, remove the access group that is referencing the access list and then remove the access list.
- Before removing an interface, which is configured with an ACL that denies certain traffic, you must remove the ACL and commit your configuration. If this is not done, then some packets are leaked through the interface as soon as the **no interface <interface-name>** command is configured and committed.
- An access list must exist before you can use the **ipv4 access group** command.
- ACL-based Forwarding (ABF) is not supported in common ACLs.
- Filtering of MPLS packets with the explicit-null or de-aggregation label is supported on the ingress direction.
- If the Ternary Content-Addressable Memory (TCAM) utilization is high and large ACLs are modified, then an error may occur. During such instances, remove the ACL from the interface and reconfigure the ACL. Later, reapply the ACL to the interface.
- **ACL policy on an MPLS interface** - An IP access control list (IP ACL) is in effect when applied on an MPLS interface on the egress PE (NCS 5000 Series) router. When an MPLS tunnel terminates on this router, the ACL policy is applied on the packets. So, ensure that correct entries are in the ACL **permit** and **deny** commands, so that the traffic is forwarded. If not, the traffic might be dropped.
- 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.

ACL Filtering by Wildcard Mask and Implicit Wildcard Mask

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

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

- A wildcard mask bit 0 means *check* the corresponding bit value.
- A wildcard mask bit 1 means *ignore* that corresponding bit value.

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

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

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

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 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 MPLS packets through interface ACL is not supported.

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.

Implementing Access Lists

Implementing ACLs involve:

1. Creating Standard or Extended ACLs-

   Create an ACL that includes an action element (permit or deny) and a filter element based on criteria such as source address, destination address, protocol, and protocol-specific parameters.

2. Applying the ACL to specific interfaces-

   After configuring an access list, you must reference the access list to make it work. Access lists can be applied on inbound interfaces. 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.

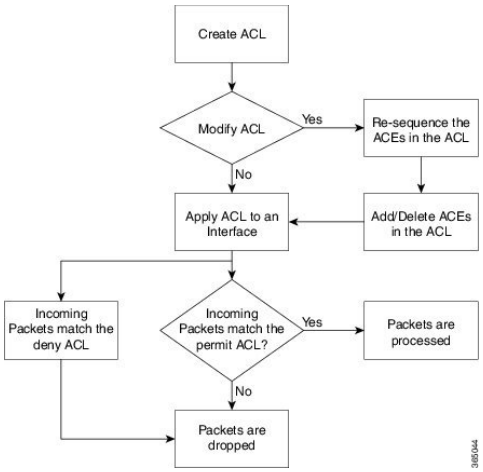
   When you apply an access list that has not yet been defined to an interface, the software acts as if the access list has not been applied to the interface and accepts all packets. Note this behavior if you use undefined access lists as a means of security in your network.

The other actions that can be performed on an ACL are:

- Copying Access Lists—

  Users can create a copy of an existing access list using the **copy access-list ipv4 | ipv6** command.

- Adding or Deleting Access-List Entries



Note If there is no match to either permit or deny the packets, then the default action is to drop the packet.

Configuring Extended Access Lists

Configuration Example

Creates an IPv4 named access list "acl_1". This access list permits ICMP protocol packets with any source and destination IPv4 address and denies TCP protocol packets with any source and destination IPv4 address and destination port greater than 5000.

```
Router#configure
Router(config)#ipv4 access-list acl_1


Router(config-ipv4-acl)#20 permit icmp any any
Router(config-ipv4-acl)#30 deny tcp any any gt 5000
Router(config-ipv4-acl)#commit
```

Running Configuration

```
Router#  show running-config ipv4 access-list acl_1
ipv4 access-list acl_1
 20 permit icmp any any
 30 deny tcp any any gt 5000
!
```

Verification

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

```
Router# show access-lists acl_1
ipv4 access-list acl_1
 20 permit icmp any any
 30 deny tcp any any gt 5000
Router#
```

Associated Commands

IPv4 Commands:

- ipv4 access-list
- permit (IPv4)
- remark(IPv4)
- deny (IPv4)

What to Do Next

After creating an access list, you must apply it to a line or an interface. ACL commit fails while adding and removing unique Access List Entries (ACE). This happens due to the absence of an assigned manager process. The user has to exit the ACL configuration mode and re-enter it before adding the first ACE.

Configuring Standard Access Lists

Configuration Example

Creates an IPv4 named access list "acl_1" with a remark "Do not allow user1 to telnet out". This access list permits packets from the source address of "172.16.0.0" with a wild-card mask of "0.0.255.255"

```
Router# configure
Router(config)# ipv4 access-list acl_1


Router(config-ipv4-acl)# 10 remark Do not allow user1 to telnet out
Router(config-ipv4-acl)# 20 permit 172.16.0.0 0.0.255.255
/* Repeat the above step as necessary, adding statements by sequence number where you
planned.
Use the no sequence-number command to delete an entry */

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

Running Configuration

```
Router# show running-config ipv4 access-list acl_1
ipv4 access-list acl_1
 10 remark Do not allow user1 to telnet out
 20 permit ipv4 172.16.0.0 0.0.255.255 any
!
```

Verification

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

```
Router# show access-lists ipv4 acl_1
ipv4 access-list acl_1
 10 remark Do not allow user1 to telnet out
 20 permit ipv4 172.16.0.0 0.0.255.255 any
```

Associated Commands

IPv4 Commands:

- ipv4 access-list
- remark(IPv4)
- permit (IPv4)
- deny (IPv4)
- show access-lists ipv4

What to Do Next

After creating an access list, you must apply it to a line or an interface. ACL commit fails while adding and removing unique Access List Entries (ACE). This happens due to the absence of an assigned manager process. The user has to exit the config-ipv4-acl or config-ipv6-acl mode to configuration mode and re-enter the same mode before adding the first ACE.

Applying Access Lists

Configuration Example

Applies the access lists on the interface, which acts as filters on packets inbound from TenGigE interface 0/0/0/2.

```
Router# configure
Router(config)# interface TenGigE 0/0/0/2
Router(config-if)# ipv4 access-group acl_1 ingress


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

Verification

Verify that the ACL applied (acl_1) on the interface is listed:

```
Router #show access-lists interface TenGigE 0/0/0/2
Input ACL
(common): N/A      (interface): acl_1
Output ACL: N/A
Router#
```

Associated Commands

IPv4 Commands:

- ipv4 access-group
- show access-lists ipv4

Ingress ACL over BVI

You can configure ACL over Bridge Virtual Interface (BVI). If you configure ACLs over BVIs, L3 traffic flows are filtered based on the ACLs applied on BVI interfaces and L2 traffic flows are filtered based on the ACLs applied on bridge domain interfaces. Otherwise, both L2 and L3 traffic are filtered based on the ACLs applied on bridge domain interfaces. Therefore, configuration of ACL over BVI reduces load on core traffic for routing and improves overall performance of the network.

BVI can have bridge association ports from different TCAM pipes. Therefore, ACL rules are replicated and programmed on all the TCAM pipes.

Restrictions

Consider the following restrictions when you configure ACL over BVI:

- On a BVI interface, Multicast packets cannot be filtered using ACL rules. So, the Multicast packets would not be accounted in the ACL rules. To overcome this discrepancy, the same ACL rule can be configured as a bridge domain ACL to do corresponding actions (permit/deny).
- Though the multicast packets are not filtered on the basis of ACL over BVI, you need to configure an ACL rule to prevent OSPF packets from hitting the implicit deny rule and be dropped in the control plane. If you do not configure an ACL rule over BVI to handle OSPF packets, OSPF sessions wouldn't be established.
- ACL TCAM region is shared among interfaces, but configuration of BVI over ACL cannot be shared to other interfaces (physical/sub-interface/Bundle-interface/Bundle-Sub Interface) or vice-versa.

Configure ACL over BVI

Use the following steps to configure ACL over BVI.

1. Enter the global configuration mode and configure an ACL.
2. Configure ACL over BVI.

Configuration Example

```
/* Enter the global configuration mode and configure an ACL */
Router# configure
Router(config)# ipv4 access-list alpha-acl
Router(config-ipv4-acl)# permit ipv4 any any
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# exit

/* Configure ACL over BVI */
Router(config)# int BVI 1
Router(config-if)# ipv4 access-group alpha-acl ingress
Router(config-l2vpn-bg-bd-bvi)# commit
```

Verification

Use the following command to verify the ACL applied on hardware (L3 interfaces):

```
Router# show access-lists alpha-acl hardware ingress location 0/RP0/CPU0

ipv4 access-list alpha-acl
10 permit ipv4 any any (100 hw matches)
```

Use the following command to get detailed information about ACL applied on hardware:

```
Router# show feature-mgr client pfilter-ea feature-info feature-name alpha-acl direction
ingress lookup ipv4 interface BVI 1 location 0/RP0/CPU0

Feature name         : alpha-acl
Lookup Type                  : IPV4_PFILTER (L3)
```

```
Direction                     : IN
Reference Count      : 1
NPU                  : 0
NP Pipe ID           : 0
Interfaces Attached  : 0x8000014
TCAM Key Size        : 92
TCAM Result Size     : 55
TCAM Region ID       : 0
VMR ID               : 0x1
Number of Entries    : 2
ACL ID               : 2
Number of ACEs       : 2
         Sequence Number: 10
                 Number of TCAM Entries: 1
                 Logical Stats Pointer:  {0xd6, 0xd6}
         Sequence Number: PFILTER_EA_DUMMY_DENY_SEQUENCE
                 Number of TCAM Entries: 1
                 Logical Stats Pointer:  {0xd7, 0xd7}
New Region           : No
Checksum             : 1074541312

Feature name         : alpha-acl
Lookup Type                  : IPV4_PFILTER (L3)
.
.
.
```

Sequencing Access List Entries and Revising the Access List

Configuration Example

Assigns sequence numbers to entries in an access list and explains how to add or delete an entry to or from an access list. In this configuration, it is assumed that a user wants to revise an access list.

Resequences entries in the access list “acl_1”. The starting value in the resequenced access list is 20, and increment value is 15.

Note When an ACL is configured under an interface and its resequenced and rolled back, the interface experiences traffic loss for a short period of time.

```
Router#configure
Router(config)#ipv4 access-list acl_1

Router(config-ipv4-acl)#10 permit 10.1.1.1
*/Repeat the above step as necessary adding statements by sequence number where you planned*/
Router(config-ipv4-acl)#20 permit 10.2.0.0 0.0.255.25
Router(config-ipv4-acl)#no 25
*/Use the no sequence-number command to delete an entry with that specific sequence number*/
Router(config-ipv4-acl)#30 permit tcp host 10.2.2.2 255.255.0.0 any eq telnet
Router(config-ipv4-acl)#commit
end
Router#resequence access-list ipv4 acl_1 20 15
```

Running Configuration

```
Before resequencing:
Router#show running-config ipv4 access-list acl_1
ipv4 access-list acl_1
 10 remark Do not allow user1 to telnet out
 20 permit ipv4 172.16.0.0 0.0.255.255 any
!
*/After resequencing using the resequence access-list ipv4 acl_1 20 15 command*/:

Router# show running-config ipv4 access-list acl_1
ipv4 access-list acl_1
 20 permit ipv4 host 10.1.1.1 any
 35 permit ipv4 10.2.0.0 0.0.255.25 any
 50 permit ipv4 10.1.1.1/24 any
 65 permit tcp host 10.2.2.2 any eq telnet
!
```

Verification

Verify that the access list (acl_1) contains all permit and deny options (after resequencing) that were configured:

```
Router#show access-lists ipv4 acl_1
ipv4 access-list acl_1
20 permit ipv4 host 10.1.1.1 any
35 permit ipv4 10.2.0.0 0.0.255.25 any
50 permit ipv4 10.1.1.1/24 any
65 permit tcp host 10.2.2.2 any eq telnet
```

Associated Commands

IPv4 Commands:

- resequence access-list ipv4
- ipv4 access-list
- permit (IPv4)
- show access-lists ipv4

Use case

This use case explains how to create, resequence, add new entries, delete an entry and verify the ACL:

```
*/ 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
```

```
/* Resequence the ACL */
Router(config)#resequence ipv4 access-list acl_1 10 20
/* 10 indicates the starting value in the resequenced ACL and the subsequent entries in the
 original ACL are incremented by 20 */

/* Verify the entries of the ACL */
Router#show access-lists ipv4 acl_1
10 permit ip host 10.3.3.3 host 172.16.5.34
30 permit icmp any any
50 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

/* When an entry is added without a sequence number, it is automatically given a sequence
number
that puts it at the end of the access list. Because the default increment is 10, the entry
 will have a sequence
number 10 higher than the last entry in the existing access list */

/* Verify the entries of the ACL */
Router(config)#show access-lists ipv4 acl_1
10 permit ip 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) */
50 permit tcp any host 10.3.3.3
60 permit 10 .4.4.4 0.0.0.255---/* newly added ACE (without the sequence number) */
/* The entry with the sequence number 30, that is, "30 permit icmp any any" is deleted from
 the ACL */
```

Understanding Prefix Lists

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

BGP Filtering Using Prefix Lists

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

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

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

How the System Filters Traffic by Prefix List

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

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

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

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

Configuring Prefix Lists

Configuration Example

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

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


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

Router(config-ipv4_pfx)#commit
```

Running Configuration

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

Verification

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

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

Associated Commands

IPv4 Commands:

- ipv4 prefix-list
- show prefix-list ipv4

Sequencing Prefix List Entries and Revising the Prefix List

Configuration Example

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

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

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


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

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

Running Configuration

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

Verification

Verify that the prefix list has been resequenced:

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

Associated Commands

IPv4 Commands:

- resequence prefix-list ipv4
- ipv4 prefix-list
- show prefix-lists ipv4

Understanding Access Lists and Prefix Lists

Atomic ACL Updates By Using the Disable Option

Atomic ACL updates involve the insertion, modification, or removal of Access List Entries (ACEs) on an interface that is in operation. Such atomic updates consume up to 50% of TCAM resources. There can be an instance where multiple modifications are required and the available resources are not sufficient. The solution to this problem is to disable atomic ACL updates such that the old ACEs are deleted before the new ACEs are added.

Note When you configure the **atomic-disable** statement in an ACL, any ACE modification detaches the ACL, until the modification is complete. In addition to this, the ACL rules are not applied during the modification process. Hence, it is recommended to configure to either permit or deny all traffic until the modification is complete.

Configuration for Disabling Atomic ACL Updates

To disable atomic updates on the hardware, by permitting packets that match the ACE rule, use the following configuration.

```
RP/0/RP0/CPU0:router# hardware access-list atomic-disable default-action permit
```

To disable atomic updates on the hardware, by denying all packets until the modification is complete, use the following configuration.

```
RP/0/RP0/CPU0:router# hardware access-list atomic-disable
```

Modifying ACLs when Atomic ACL Updates are Disabled

On disabling atomic ACL updates on the hardware, use the steps in this section to modify ACLs.

Add an ACE

Use the following steps to add an ACE.

1. Locate the ACL you want to modify.

```
RP/0/RP0/CPU0:router(config)# do show access-lists
...
!
ipv4 access-list list1
 10 permit ipv4 10.1.1.0/24 any
 20 permit ipv4 20.1.1.0/24 any
```

```
!
```

2. Add the ACE to the ACL.

```
RP/0/RP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RP0/CPU0:router(config-ipv4-acl)# 30 permit ipv4 30.1.1.0/24 any
RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if your ACE was added successfully.

```
RP/0/RP0/CPU0:router(config)# do show access-lists
...
!
ipv4 access-list list1
 10 permit ipv4 10.1.1.0/24 any
 20 permit ipv4 20.1.1.0/24 any
 30 permit ipv4 30.1.1.0/24 any
!
```

You have successfully added an ACE.

Delete an ACE

Use the steps in this section to delete an ACE.

1. Locate the ACL containing the ACE that you want deleted.

```
RP/0/RP0/CPU0:router(config)# do show access-lists
...
!
ipv4 access-list list1
 10 permit ipv4 10.1.1.0/24 any
 20 permit ipv4 20.1.1.0/24 any
 30 permit ipv4 30.1.1.0/24 any
!
```

2. Delete the ACE.

```
RP/0/RP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RP0/CPU0:router(config-ipv4-acl)# no 30
RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the ACE has been removed from the ACL.

```
RP/0/RP0/CPU0:router(config-ipv4-acl)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 10.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

You have successfully deleted an ACE.

Replace an ACE

Use the steps in this section to replace an ACE.

1. Locate the ACL you want to modify.

```
RP/0/RP0/CPU0:router(config-ipv4-acl)#do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 10.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

2. Configure the new ACE to replace the existing ACE.

```
RP/0/RP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit ipv4 11.1.1.0/24 any
RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the ACE replacement is successful.

```
RP/0/RP0/CPU0:router(config)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 11.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

You have successfully replaced an ACE.

Delete an ACE and Add a New ACE

Use the following steps to delete an ACE and add a new ACE.

1. Locate the ACL you want to modify.

```
RP/0/RP0/CPU0:router(config)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 11.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

2. Delete the required ACE, and add the new ACE.

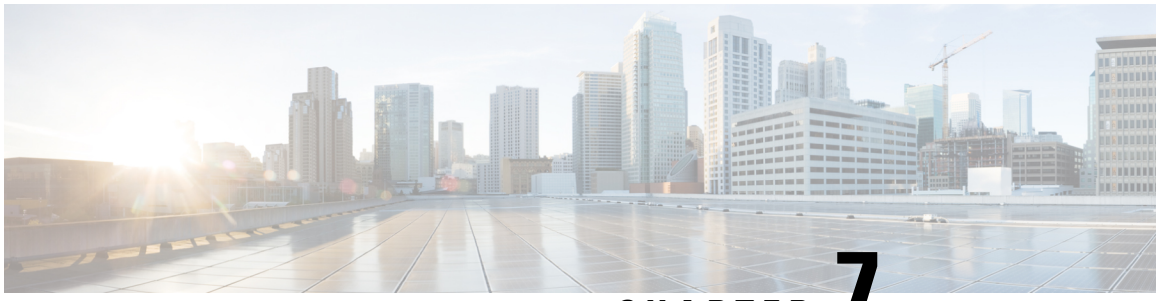
```
RP/0/RP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RP0/CPU0:router(config-ipv4-acl)# no 20
RP/0/RP0/CPU0:router(config-ipv4-acl)# permit ipv4 12.1.1.0/24 any
RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the modification is successful.

```
RP/0/RP0/CPU0:router(config)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 11.1.1.0 0.0.0.255 any
 20 permit ipv4 12.1.1.0 0.0.0.255 any
```

You have successfully deleted an ACE, and added a new ACE.

Similarly, you can combine the addition, removal, and replacement of ACEs.



CHAPTER 7

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.

Components

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

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

  Adjacency FIB entries are maintained only on a local node, and adjacency entries linked to FIB entries could be different.

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

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

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

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

CEF Benefits

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

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

- IPv4 CEF database—Stores IPv4 Unicast routes for forwarding IPv4 unicast packets
- IPv6 CEF database—Stores IPv6 Unicast routes for forwarding IPv6 unicast packets
- MPLS LFD database—Stores MPLS Label table for forwarding MPLS packets

Verifying CEF

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

- show cef {*ipv4 address*| *ipv6 address*} hardware egress

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

```
Router# show cef 37.37.37.37/32 hardware egress
37.37.37.37/32, version 46, internal 0x1000001 0x0 (ptr 0x8b0477f8) [1], 0x0 (0x0), 0x0
(0x0)
local adjacency 1.0.0.2
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
   via 1.0.0.2/32, TenGigE0/0/0/0, 5 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8b43bb30 0x0]
    next hop 1.0.0.2/32
    local adjacency
 Show-data Print at RPLC
 LEAF - HAL pd context :
 sub-type : IPV4, ecd_marked:0, has_collapsed_ldi:0
```

```
 collapse_bwalk_required:0, ecdv2_marked:0
Leaf H/W Result:
  VRF            Net Address                     Mask                  Status
    0            37.37.37.37         255.255.255.255              Programmed

TX H/W Result for NP:0 (index: 0x18739 (BE)):
 Next Hop Data
 Next Hop Valid:         YES
 Next Hop Index:         100153
 --More-- RP/0/RP0/CPU0:Aug  6 16:38:59.302 : vic_1[180]: %PLATFORM-VIC-4-SIGNAL : Interface
 HundredGigE0/0/1/3, Detected Signal failure
RP/0/RP0/CPU0:Aug  6 16:38:59.312 : ifmgr[172]: %PKT_INFRA-LINK-3-UPDOWN : Interface
HundredGigE0/0/1/3, changed state to Up
 Egress Next Hop IF:     100045
 Hw Next Hop Intf:       6
 HW Port:                1
 Next Hop Flags:         COMPLETE-->This indicates that the router can forward traffic.
 Next Hop MAC:           4055.395f.46b7

NHINDEX H/W Result for NP:0 (index: 0 (BE)):
NhIndex is NOT required on this platform

NHINDEX STATS: pkts 0, bytes 0 (all NPs combined, no stats)

RX H/W Result on NP:0 [Adj ptr:0x40 (BE)]:
Rx-Adj is NOT required on this platform
```

- show cef {ipv4| ipv6} summary

Displays a summary of the IPv4 or IPv6 CEF table.

```
Router#show cef ipv4 summary
Router ID is 0.0.0.0
IP CEF with switching (Table Version 0) for node0_RP0_CPU0
  Load balancing: L4
  Tableid 0xe0000000 (0x8a2d7380), Vrfid 0x60000000, Vrid 0x20000000, Flags 0x1019
  Vrfname default, Refcount 88
  39 routes, 0 protected, 0 reresolve, 0 unresolved (0 old, 0 new), 9048 bytes
  11 rib, 0 lsd, 17:0 aib, 0 internal, 8 interface, 4 special, 1 default routes
  39 load sharing elements, 32176 bytes, 2 references
  2 shared load sharing elements, 1600 bytes
  37 exclusive load sharing elements, 30576 bytes
  0 route delete cache elements
  24 local route bufs received, 0 remote route bufs received,  0 mix bufs received
  11 local routes, 0 remote routes
  26 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, 2 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
 20 next hops
  0 incomplete next hops
 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 1.0.0.2 detail
1.0.0.2/32, version 0, internal 0x1020001 0x0 (ptr 0x8a35d1a8) [1], 0x0 (0x0), 0x0 (0x0)
 Updated Aug  3 11:00:37.829
 local adjacency 1.0.0.2
 Prefix Len 32, traffic index 0, Adjacency-prefix, precedence n/a, priority 15
  gateway array (0x8a08af70) reference count 1, flags 0x4000, source aib-hi (3), 0 backups
                [1 type 3 flags 0x48401 (0x8a1d9e18) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Aug  3 11:00:37.829
 LDI Update time Aug  3 11:00:37.829
   via 1.0.0.2/32, TenGigE0/0/0/0, 3 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8b7df0a0 0x0]
    next hop 1.0.0.2/32
    local adjacency
    Load distribution: 0 (refcount 1)
    Hash  OK  Interface                 Address
    0     Y   TenGigE0/0/0/0            1.0.0.2
```

- show adjacency detail

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

```
Router#show adjacency detail
------------------------------------------------------------------------------
0/RP0/CPU0
------------------------------------------------------------------------------
Interface                  Address                 Version  Refcount Protocol
Te0/0/0/29                  (interface)                 30        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4
                           0 packets, 0 bytes
Te0/0/0/77                  (interface)                 78        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4
                           0 packets, 0 bytes
Te0/0/0/20                  (interface)                 21        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4
                           0 packets, 0 bytes
Te0/0/0/33                  (interface)                 34        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4
                           0 packets, 0 bytes
Hu0/0/1/3                   (interface)                 84        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4
                           88 packets, 3696 bytes
Te0/0/0/24                  (interface)                 25        1(    0)
```

Unicast Reverse Path Forwarding

Configuration of Unicast IPv4 and IPv6 Reverse Path Forwarding (uRPF) enables a router to verify the reachability of the source address in packets being forwarded. Configuring uRPF, both strict and loose modes, helps to mitigate problems caused by the introduction of spoofed IP source addresses into a network. Configuration of uRPF discards IP packets that lack a verifiable IP source address after a reverse lookup in the CEF table.

When **strict uRPF** is enabled, the source address of the packet is checked in the FIB. If the packet is received on the same interface that would be used to forward the traffic to the source of the packet, the packet passes

the check and is further processed. Otherwise, the packet is dropped. Configure strict uRPF only where there is natural or configured symmetry. Internal interfaces are likely to have a routing asymmetry, that is, multiple routes to the source of a packet. Therefore, you should not implement strict uRPF on interfaces that are internal to the network.

Implementation of strict mode uRPF requires maintenance of a uRPF interfaces list for the prefixes. The list contains only the interfaces configured with strict mode uRPF. The interfaces are provided by the prefix path. The uRPF interface list is shared among the prefixes wherever possible.

When **loose uRPF** is enabled, the source address of the packet is checked in the FIB. If the source address exists and matches a valid forwarding entry, the packet passes the check and is further processed. Otherwise, the packet is dropped.

Note The behavior of strict uRPF varies slightly on the basis of platforms, the number of recursion levels, and the number of paths in Equal-Cost Multipath (ECMP) scenarios. A platform may switch to loose uRPF check for some or all prefixes, even though strict uRPF is configured. For example, if ECMP Path is eight or more, strict mode is converted to loose mode.

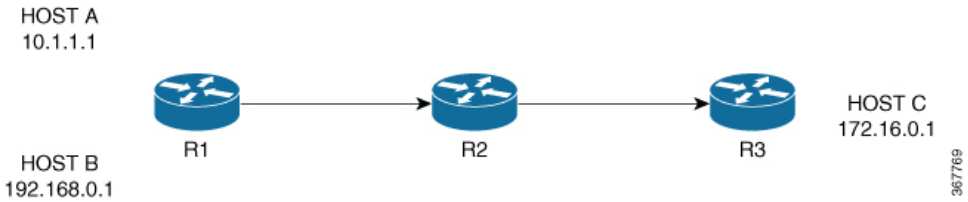
Loose and strict uRPF supports two options: **allow self-ping** and **allow default**. The **allow self-ping** option allows the source of the packet to ping itself. The **allow default** option allows the lookup result to match a default routing entry. When the **allow default** option is enabled with the strict mode of the uRPF, the packet is processed further only if it arrives through the default interface.

Restrictions

Consider the following restrictions when you configure uRPF:

- Global configuration followed by cold reload is required to enable or disable uRPF on the router.
- Configuration of uRPF per interface enables or disables uRPF mode in the hardware of the corresponding interface.
- Configuration of uRPF reduces the route scale to half. Half of the routing table is used for destination lookup and the other half is used for source lookup of received packets.
- Unicast RPF allows packets with 0.0.0.0 source addresses and 255.255.255.255 destination addresses to pass so that Bootstrap Protocol and Dynamic Host Configuration Protocol (DHCP) functions properly.
- Unicast RPF allows packets whose destination IP address is not a unicast address.
- The behavior of strict uRPF varies slightly on the basis of platforms, the number of recursion levels, and the number of paths in Equal-Cost Multipath (ECMP) scenarios. A platform may switch to loose uRPF check for some or all prefixes, even though strict uRPF is configured.
- The **allow self-ping** option is the default option in uRPF configuration for both strict and loose mode. You cannot disable the **allow-self-ping** option. The **allow-default** option needs to be configured specifically per interface.

Figure 12: uRPF Topology



In Figure 1, uRPF is enabled on Router R2 and R2 has the following FIB table entries:

- 10.1.1.0/24 [110/3] via 10.25.24.1, 2d08h, HundredGigE0/0/0/24
- 12.1.1.0/24 [110/3] via 10.25.24.1, 2d08h, HundredGigE0/0/0/25
- 14.0.5.0/24 [110/3] via 20.0.0.2, 2d08h, HundredGigE0/0/0/1

R2 allows the packet from Host 1 to be routed because Host 1 subnet is available in R2's FIB table. However, R2 does not allow Host 3 to be routed because Host 3 subnet is not available in R2's FIB table. If strict uRPF is enabled on R2, then the source address 10.1.1.1/24 should be reachable through the same interface from which it is received. If loose uRPF is enabled on R2, then it is not mandatory that the source address 10.1.1.1/24 be reachable through the same interface from which it is received. The only criteria for a packet to be forwarded is that the host address should be present in R2's FIB table.

Configure Unicast Reverse Path Forwarding

Configuring Unicast Reverse Path Forwarding (uRPF) enables a router to verify the reachability of the source address of packets being forwarded. If the source IP address is not valid, the packet is discarded. This capability can limit the appearance of spoofed addresses in a network.

To configure uRPF on a hardware module, use the following steps:

1. Configure a hardware module in the global configuration mode and enable uRPF.
2. Reboot the router.
3. Configure an interface.
4. Configure IPv4 or IPv6 uRPF through strict or loose mode under the interface.

Configuration

Use the following configuration to configure uRPF in strict mode:

```
/* Configure a hardware module in the global configuration mode and enable uRPF. */
Router# configure
Router(config)# hw-module urpf enable
Mon Sep 17 09:34:00.945 UTC
In order to activate/deactivate urpf, you must manually reboot the box.
Router(config)# commit


/* Reboot the router manually. */

/* Configure an interface. */
```

```
Router(config)# interface BVI1
Router(config-ipv6-acl)# description BVI INTERFACE
Router(config-ipv6-acl)# commit

/* Configure IPv4 or IPv6 uRPF through strict or loose mode under the interface. */
Router(config)# ipv4 address 11.1.1.1 255.255.255.0
Router(config-pifib-policer-global)# ipv4 verify unicast source reachable-via rx
Router(config-pifib-policer-global)# commit
```

Use the following configuration to configure uRPF in loose mode:

```
/* Configure a hardware module in the global configuration mode and enable uRPF. */
Router# configure
Router(config)# hw-module urpf enable
Mon Sep 17 09:34:00.945 UTC
In order to activate/deactivate urpf, you must manually reboot the box.
Router(config)# commit


/* Reboot the router manually. */

/* Configure an interface. */
Router(config)# interface BVI1
Router(config-ipv6-acl)# description BVI INTERFACE
Router(config-ipv6-acl)# commit

/* Configure IPv4 or IPv6 uRPF through strict or loose mode under the interface. */
Router(config)# ipv4 address 11.1.1.1 255.255.255.0
Router(config-pifib-policer-global)# ipv4 verify unicast source reachable-via any
Router(config-pifib-policer-global)# commit
```

Verification

Use the following command to check the uRPF status:

```
Router#show prm server  profile-status software
Software PROFILE STATUS VARIABLES
**************************************
BGP-3107 LB             : 0
LB Seed                 : 0x65c5208d
LB Seed Cfg             : NO
LB Sub-Sel-Offset Cfg   : NO
LB SubSel               : Hash Field B1
LB Offset               : 13
SR-PE Mode              : 0
Current URPF            : YES
Configured URPF         : YES
Current HW Profile      : SP-Profile
Configured HW Profile   : SP-Profile
**************************************
```

Per-Flow Load Balancing

The system inherently supports the 5-tuple hash algorithm. Load balancing describes the functionality in a router that distributes packets across multiple links based on Layer 3 (network layer) and Layer 4 (transport layer) routing information. If the router discovers multiple paths to a destination, the routing table is updated with multiple entries for that destination.

Per-flow load balancing performs these functions:

- Incoming data traffic is evenly distributed over multiple equal-cost connections.
- Incoming data Data traffic is evenly distributed over multiple equal-cost connections member links within a bundle interface.
- Layer 2 bundle and Layer 3 (network layer) load balancing decisions are taken on IPv4, IPv6, and MPLS flows which are supported for the 5-tuple hash algorithm.
- A 5-tuple hash algorithm provides more granular load balancing and used for load balancing over multiple equal-cost Layer 3 (network layer) paths. The Layer 3 (network layer) path is on a physical interface or on a bundle interface. In addition, load balancing over member links can occur within a Layer 2 bundle interface.
- The 5-tuple load-balance hash calculation contains:
  - Source IP address
  - Destination IP address
  - Router ID
  - Source port
  - Destination port

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 TenGigE interface. The destination IP address is 37.37.37.37/32 and the next hop address is 1.0.0.2.


```
Router(config)# router static address-family ipv4 unicast
Router(config-static-afi)# 37.37.37.37/32 tengige 0/0/0/0 1.0.0.2
Router(config-static-afi)# commit
```

Running Configuration

```
Router#show running-config router static address-family ipv4 unicast
router static
 address-family ipv4 unicast
```

```
 37.37.37.37/32 tengigE 0/0/0/0 1.0.0.2
 !
!
```

Verification

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

```
Router#show cef 37.37.37.37/32 hardware egress
37.37.37.37/32, version 46, internal 0x1000001 0x0 (ptr 0x8b0477f8) [1], 0x0 (0x0), 0x0
(0x0)
local adjacency 1.0.0.2
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
   via 1.0.0.2/32, TenGigE0/0/0/0, 5 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8b43bb30 0x0]
    next hop 1.0.0.2/32
    local adjacency
 Show-data Print at RPLC
 LEAF - HAL pd context :
 sub-type : IPV4, ecd_marked:0, has_collapsed_ldi:0
 collapse_bwalk_required:0, ecdv2_marked:0
Leaf H/W Result:
  VRF            Net Address                    Mask                  Status
    0            37.37.37.37         255.255.255.255              Programmed

TX H/W Result for NP:0 (index: 0x18739 (BE)):
 Next Hop Data
 Next Hop Valid:         YES
 Next Hop Index:         100153
 --More--  vic_1[180]: %PLATFORM-VIC-4-SIGNAL : Interface HundredGigE0/0/1/3, Detected
Signal failure
: ifmgr[172]: %PKT_INFRA-LINK-3-UPDOWN : Interface HundredGigE0/0/1/3, changed state to Up

 Egress Next Hop IF:     100045
 Hw Next Hop Intf:       6
 HW Port:                1
 Next Hop Flags:         COMPLETE--->This indicates that the router can forward traffic.
 Next Hop MAC:           4055.395f.46b7

NHINDEX H/W Result for NP:0 (index: 0 (BE)):
NhIndex is NOT required on this platform

NHINDEX STATS: pkts 0, bytes 0 (all NPs combined, no stats)

RX H/W Result on NP:0 [Adj ptr:0x40 (BE)]:
Rx-Adj is NOT required on this platform
```

Associated Commands

- show cef

BGP Accounting Policy Statistics for Interfaces and Subinterfaces

Table 6: Feature History Table

Feature Name	Release	Description
BGP Accounting Policy Statistics for Interfaces and Subinterfaces	Release 7.9.1	Border Gateway Protocol (BGP) policy accounting measures and classifies IP traffic that is received from different peers. You can identify and account for all traffic by customer and bill accordingly. Policy accounting is enabled on an individual input interface basis. Using BGP policy accounting, you can now account for traffic according to the route it traverses. This feature is supported on routers that have the Cisco NC57 based line cards with external TCAM (eTCAM) and operate in native mode. This feature introduces the **hw-module fib bgppa stats-mode** command.

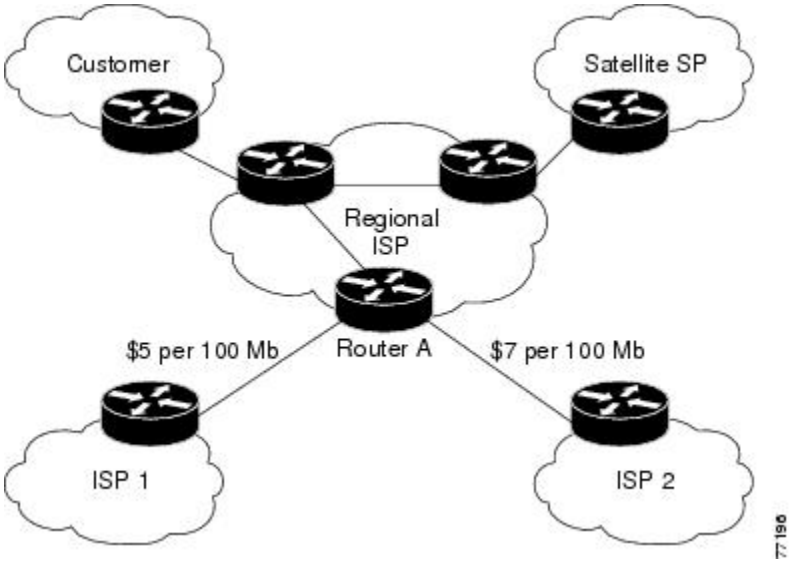
Border Gateway Protocol (BGP) policy accounting measures and classifies IP traffic that is received from, different peers. Policy accounting is enabled on an individual input interface basis, and counters based on parameters such as community list, autonomous system number, or autonomous system path are assigned to identify the IP traffic.

Note There are two types of route policies. The first type (regular BGP route policies) is used to filter the BGP routes advertised into or out from the BGP links. This type of route policy is applied to the specific BGP neighbor. The second type (specific route policy) is used to set up a traffic index for the BGP prefixes. This route policy is applied to the global BGP IPv4 or IPv6 address family to set up the traffic index when the BGP routes are inserted into the RIB table. BGP policy accounting uses the second type of route policy.

Using BGP policy accounting, you can account for traffic according to the route it traverses. Service providers can identify and account for all traffic by customer and bill accordingly. In the Figure 13: Sample Topology for BGP Policy Accounting, BGP policy accounting can be implemented in Router A to measure packet and byte volumes in autonomous system buckets. Customers are billed appropriately for traffic that is routed from a domestic, international, or satellite source.

Note BGP policy accounting measures and classifies IP traffic for BGP prefixes only.

Figure 13: Sample Topology for BGP Policy Accounting



Based on the specified routing policy, BGP policy accounting assigns each prefix a traffic index (bucket) associated with an interface. BGP prefixes are downloaded from the RIB to the FIB along with the traffic index.

There are a total of 15 (1 to 15) traffic indexes (bucket numbers) that can be assigned for BGP prefixes. Internally, there is an accounting table associated with the traffic indexes to be created for each input (ingress). The traffic indexes allow you to account for the IP traffic, where the destination IP address is BGP prefixes.

Note The default traffic index for BGP route is 0.

This feature introduces the **hw-module fib bgppa stats-mode** command. After configuring the command, you must reload the router for the feature to take effect.

Restriction

- This feature is applicable for the following address families:
  - IPv4
  - IPv6
- This feature supports input destination-based accounting only.

Configuration

For main interface:

```
Router# config
Router(config)# hw-module fib bgppa stats-mode main-intf

Router(config)# commit
```

For sub interface:

```
Router# config
Router(config)# hw-module fib bgppa stats-mode sub-intf

Router(config)# commit
```

Note After configuring the command, you must reload the router for the feature to take effect.

Running Configuration

For main interface:

```
hw-module fib bgppa stats-mode main-intf
!
```

For sub interface:

```
hw-module fib bgppa stats-mode sub-intf
!
```

Verification

The show output displays that the BGP policy accounting is configured.

```
Router#show ipv4 int bundle-ether 54
Bundle-Ether54 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 54.1.1.2/24
  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
IP Input BGP policy accounting is configured
```

The following example shows the stats details.

For IPv4:

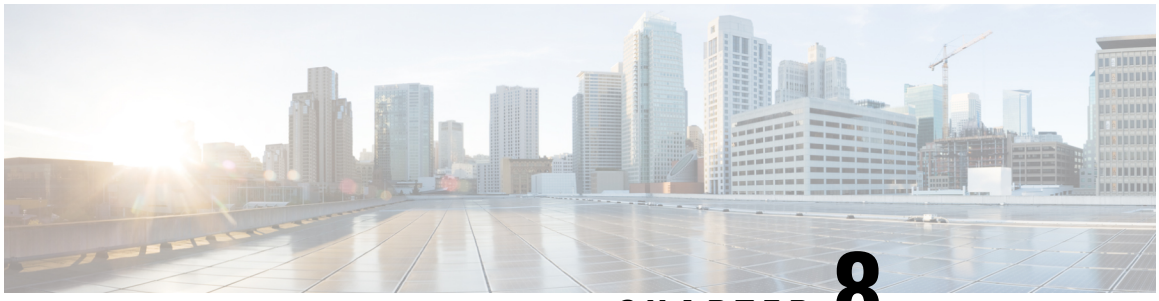
```
Router#show cef ipv4 interface bundle-ether 22 bgp-policy-statistics
Bundle-Ether22 is UP
Input BGP policy accounting on dst IP address enabled
  buckets      packets       bytes
  default    207598474 309736310837
```

```
  1            4946185  7379708020
  2            2471450  3687403400
  3            2472189  3688505988
  4            2472271  3688628332
  5            2468753  3683379476
  6            2468877  3683564484
  7            2472353  3688750676
  8            2472434  3688871528
  9            2472517  3688995364
  10           2468958  3683685336
  11           2469081  3683868852
  12           2472599  3689117708
  13           2467559  3681598028
  14           2467682  3681781544
  15           2472680  3689238560
```

For IPv6:

```
Router#show cef ipv6 interface hundredGigE 0/7/0/26 bgp-policy-statistics
HundredGigE0/7/0/26 is UP
Input BGP policy accounting on dst IP address enabled
  buckets      packets       bytes
  default    275658703 412385419688
  1             166908   249694368
  2              83455   124848680
  3              83455   124848680
  4              83455   124848680
  5              83456   124850176
  6              83456   124850176
  7              83456   124850176
  8              83456   124850176
  9              83457   124851672
  10             83457   124851672
  11             83457   124851672
  12             83458   124853168
  13             83458   124853168
  14             83458   124853168
  15             83459   124854664
```

Note The traffic index 0 is accounted into the *default* bucket.



CHAPTER 8

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

Table 7: Feature History Table

Feature Name	Release Information	Description
Monitor LPTS Host Path Drops via YANG Data Model	Release 7.3.2	This feature allows you to use the `Cisco-IOS-XR-lpts-pre-ifib-oper.yang` data model to monitor the policer action for Local Packet Transport Services (LPTS) flow type for all IOS XR platforms. To access this data model, see the Github repository.

In Cisco IOS XR, the control packets, which are destined to the Route Processor (RP), are policed using a set of ingress policers in the incoming ports. These policers are programmed statically during bootup by LPTS components. The policers are applied based on the flow type of the incoming control traffic. The flow type is determined by looking at the packet headers. The policer rates for these static ingress policers are defined in a configuration file, which are programmed on the route processor during bootup. You can change the policer values based on the flow types of these set of ingress policers. You are able to configure the rate per policer per node.

Configuration Example

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

- ospf unicast default rate 200
- bgp configured rate 200
- bgp default rate 100

```
Router#configure
Router(config)#lpts pifib hardware police
Router(config-pifib-policer-global)#flow ospf unicast default rate 200
Router(config-pifib-policer-global)#flow bgp configured rate 200
Router(config-pifib-policer-global)#flow bgp default rate 100
Router (config-pifib-policer-global)#commit
```

Running Configuration

```
lpts pifib hardware police
flow ospf unicast default rate 200
flow bgp configured rate 200
flow bgp default rate 100
!
```

Verification

```
Router#show run lpts pifib hardware police
lpts pifib hardware police
flow ospf unicast default rate 200
flow bgp configured rate 200
flow bgp default rate 100
```

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

Configuration Example

Configure the LPTS policer for the OSPF and BGP flow types with the following values:

- ospf unicast default rate 100
- bgp configured rate 300

```
Router#configure
Router(config)#lpts pifib hardware police
Router(config-pifib-policer-per-node)#flow ospf unicast default rate 200
Router(config-pifib-policer-per-node)#flow bgp configured rate 200
```

```
Router(config-pifib-policer-per-node)#flow bgp default rate 100
Router(config-pifib-policer-per-node)#commit
```

Running Configuration

```
lpts pifib hardware police location 0/RP0/CPU0
flow ospf unicast default rate 100
flow bgp configured rate 300
```

Verification

```
Router#show run lpts pifib hardware police
lpts pifib hardware police
flow ospf unicast default rate 100
flow bgp configured rate 300
!
```

Verification

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

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

Trap Type                                       NPU  Trap TrapStats   Policer Packet    Packet

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

RxTrapMimSaUnknown(RCY_CFM_DOWN_MEP_DMM)         0    7    0x7         32037   0         0

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

RxTrapSaMulticast                                0    11   0xb         32018   0         0

RxTrapArpMyIp                                    0    13   0xd         32001   0         0

RxTrapArp                                        0    14   0xe         32001   11        0

RxTrapDhcpv4Server                               0    18   0x12        32022   0         0

RxTrapDhcpv4Client                               0    19   0x13        32022   0         0

RxTrapDhcpv6Server                               0    20   0x14        32022   0         0

RxTrapDhcpv6Client                               0    21   0x15        32022   0         0

RxTrapL2Cache_LACP                               0    23   0x17        32003   0         0

RxTrapL2Cache_LLDP1                              0    24   0x18        32004   0         0

RxTrapL2Cache_LLDP2                              0    25   0x19        32004   1205548   0

RxTrapL2Cache_LLDP3                              0    26   0x1a        32004   0         0

RxTrapL2Cache_ELMI                               0    27   0x1b        32005   0         0

RxTrapL2Cache_BPDU                               0    28   0x1c        32027   0         0

RxTrapL2Cache_BUNDLE_BPDU                        0    29   0x1d        32027   0         0

RxTrapL2Cache_CDP                                0    30   0x1e        32002   0         0
```

RxTrapHeaderSizeErr	0	32	0x20	32018	0	0
RxTrapIpCompMcInvalidIp	0	35	0x23	32018	0	0
RxTrapMyMacAndIpDisabled	0	36	0x24	32018	0	0
RxTrapMyMacAndMplsDisable	0	37	0x25	32018	0	0
RxTrapArpReply	0	38	0x26	32001	2693	0
RxTrapFibDrop	0	41	0x29	32018	0	0
RxTrapMTU	0	42	0x2a	32020	0	0
RxTrapMiscDrop	0	43	0x2b	32018	0	0
RxTrapL2AclDeny	0	44	0x2c	32034	0	0
Rx_UNKNOWN_PACKET	0	46	0x2e	32018	0	0
RxTrapL3AclDeny	0	47	0x2f	32034	0	0
RxTrapOamY1731MplsTp(OAM_SWOFF_DN_CCM)	0	57	0x39	32029	0	0
RxTrapOamY1731Pwe(OAM_SWOFF_DN_CCM)	0	58	0x3a	32030	0	0
RxTrapOamLevel	0	64	0x40	32023	0	0
RxTrapRedirectToCpuOamPacket	0	65	0x41	32025	0	0
RxTrapOamPassive	0	66	0x42	32024	0	0
RxTrap1588	0	67	0x43	32038	0	0
RxTrapExternalLookupError	0	72	0x48	32018	0	0
RxTrapArplookupFail	0	73	0x49	32001	0	0
RxTrapUcLooseRpfFail	0	84	0x54	32035	0	0
RxTrapMplsControlWordTrap	0	88	0x58	32015	0	0
RxTrapMplsControlWordDrop	0	89	0x59	32015	0	0
RxTrapMplsUnknownLabel	0	90	0x5a	32018	0	0
RxTrapIpv4VersionError	0	98	0x62	32018	0	0
RxTrapIpv4ChecksumError	0	99	0x63	32018	0	0
RxTrapIpv4HeaderLengthError	0	100	0x64	32018	0	0
RxTrapIpv4TotalLengthError	0	101	0x65	32018	0	0
RxTrapIpv4Ttl0	0	102	0x66	32008	0	0
RxTrapIpv4Ttl1	0	104	0x68	32008	0	0
RxTrapIpv4DipZero	0	106	0x6a	32018	0	0
RxTrapIpv4SipIsMc	0	107	0x6b	32018	0	0
RxTrapIpv6VersionError	0	109	0x6d	32018	0	0

```
RxTrapIpv6HopCount0                              0    110  0x6e        32011   0          0
RxTrapIpv6LoopbackAddress                        0    113  0x71        32018   0          0
RxTrapIpv6MulticastSource                        0    114  0x72        32018   0          0
RxTrapIpv6NextHeaderNull                         0    115  0x73        32010   0          0
RxTrapIpv6Ipv4CompatibleDestination              0    121  0x79        32018   0          0
RxTrapMplsTtl1                                   0    125  0x7d        32012   316278     2249
RxTrapUcStrictRpfFail                            0    137  0x89        32035   0          0
RxTrapMcExplicitRpfFail                          0    138  0x8a        32033   0          0
RxTrapOamp(OAM_BDL_DN_NON_CCM)                   0    141  0x8d        32031   0          0
RxTrapOamEthUpAccelerated(OAM_BDL_UP_NON_CCM)    0    145  0x91        32032   0          0
RxTrapReceive                                    0    150  0x96        32017   125266112  0
RxTrapUserDefine_FIB_IPV4_NULL0                  0    151  0x97        32018   0          0
RxTrapUserDefine_FIB_IPV6_NULL0                  0    152  0x98        32018   0          0
RxTrapUserDefine_FIB_IPV4_GLEAN                  0    153  0x99        32016   0          0
RxTrapUserDefine_FIB_IPV6_GLEAN                  0    154  0x9a        32016   0          0
RxTrapUserDefine_IPV4_OPTIONS                    0    155  0x9b        32006   0          0
RxTrapUserDefine_IPV4_RSVP_OPTIONS               0    156  0x9c        32007   0          0
RxTrapUserDefine                                 0    157  0x9d        32026   0          0
RxTrapUserDefine_BFD                             0    163  0xa3        32028   0          0
RxTrapMC                                         0    181  0xb5        32033   0          0
RxNetflowSnoopTrap0                              0    182  0xb6        32018   0          0
RxNetflowSnoopTrap1                              0    183  0xb7        32018   0          0
RxTrapMimSaMove(CFM_DOWM_MEP_DMM)                1    6    0x6         32037   0          0
RxTrapMimSaUnknown(RCY_CFM_DOWN_MEP_DMM)         1    7    0x7         32037   0          0
RxTrapAuthSaLookupFail (IPMC default)            1    8    0x8         32033   0          0
RxTrapSaMulticast                                1    11   0xb         32018   0          0
RxTrapArpMyIp                                    1    13   0xd         32001   0          0
```

Starting Cisco IOS XR Software Release 7.3.2, you can use `Cisco-IOS-XR-lpts-pre-ifib-oper` YANG data model across all IOS XR platforms to retrieve the policer statistics of the flow type. The following example shows the sample RPC request:

```
===== RPC request ======
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <lpts-pifib xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-lpts-pre-ifib-oper">
```

```
        <nodes>
          <node>
            <node-name>0/0/CPU0</node-name>
            <pifib-hw-flow-policer-stats/>
          </node>
        </nodes>
      </lpts-pifib>
    </filter>
  </get>
</rpc>
##
```

The following example show the relevant snippet of the `ICMP-local` flow response to the RPC request:

```
<police-info>
       <flow-type>23</flow-type>
       <flow-name>ICMP-local</flow-name>
       <type>2</type>
       <type-name>Global</type-name>
       <domain-id>0</domain-id>
       <domain-name>default</domain-name>
       <npu-id>255</npu-id>
       <policer-rate>0</policer-rate>
       <burst-size>750</burst-size>
       <accepted>2000</accepted>
       <dropped>1000</dropped>
      </police-info>
      <police-info>
```

The policer stats of each flow type is the aggregate of all the NPU counters. In the example, the NPU ID of 255 indicates that the value is an aggregate of all NPU stats and provides a simplified view of policer stats per flow type.

Understanding ACL-based Policers

ACL-based LPTS policers are session-based policers that provide secure network access for each session.

Benefits

These are the benefits of ACL-based policer:

- Provides rate limit on incoming packets based on session.
- Modifies policer rates depending on traffic load.
- Blocks entire traffic based on a specific session without impacting other sessions with the same flow.

Restrictions

- It is recommended to have up to 10 prefixes in a single ACL. The ACEs in an ACL should be managed such that there is no overlap of prefixes.
- Up to 50 ACL-based LPTS policers can be configured on a router.
- ACL-based LPTS policers can be configured to an IPv4 LPTS session that has a particular flow type and that matches the default VRF.
- ACL-based LPTS policers can be configured to an IPv4 or IPv6 LPTS session that has a particular flow type and that matches the VRF ID.

Configure IPv4 ACL-based LPTS Policers

This section describes how you can configure IPv4 ACL-based LPTS policers.

Configuration Example

To configure ACL-based LPTS policers for IPv4 sessions, complete the following configurations:

1. Configure an ACL.
2. Configure LPTS policer for a particular IPv4 session matching an ACL and VRF.

Configuration

```
/* Enter the global configuration mode and configure an ACL */
Router# configure
Router(config)# ipv4 access-list ACL1_OSPF
Router(config-ipv4-acl)# 10 permit ipv4 host 192.168.1.5 any
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# exit

/* Configure LPTS policer for a particular IPv4 session matching an ACL and VRF. */
Router(config)# lpts pifib hardware police
Router(config-pifib-policer-global)# acl ACL1_OSPF rate 2000 vrf red8
Router(config-if)# commit
```

Verification

Use the following command to display the ACL-based LPTS policer entries attached to matching entries:

```
Router#show lpts pifib hardward entry statistics location 0/RP0/CPU0
Offset  NPU   L3   VRF id  L4  Intf Dest Pkts/Drops laddr,Port     raddr,Port  acl name
------  -----  ---  ------  ---  ---  -----  -------  --------        --------   -----
  37   0 [0] IPV4   *    any  any   Local  0/0     any,any           any,any
  38   0 [1] IPV4   *    any  any   Local  0/0     any,any           any,any
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
 187   0 [0] IPV4   *    ICMP any   Local  0/0     any,8             any,ECHO   ACL3_ICMP
 188   0 [1] IPV4   *    ICMP any   Local  0/0     any,8             any,ECHO   ACL3_ICMP
 189   0 [2] IPV4   *    ICMP any   Local  0/0     any,8             any,ECHO   ACL3_ICMP
 190   0 [3] IPV4   *    ICMP any   Local  0/0     any,8             any,ECHO   ACL3_ICMP
8963   0 [0] IPV4 default UDP OptV2 Local  22588/0 192.168.10.2,646  any,any    ACL4_UDP
8964   0 [1] IPV4 default UDP OptV2 Local  22590/0 192.168.10.2,646  any,any    ACL4_UDP
8965   0 [2] IPV4 default UDP OptV2 Local  0/0     192.168.10.2,646  any,any    ACL4_UDP
8966   0 [3] IPV4 default UDP OptV2 Local  0/0     192.168.10.2,646  any,any    ACL4_UDP
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
8279   0 [0] IPV4 default OSPF OptV2 Local 846248/0 192.168.10.5,any any,any    ACL1_OSPF
8280   0 [1] IPV4 default OSPF OptV2 Local 962717/0 192.168.10.5,any any,any    ACL1_OSPF
8281   0 [3] IPV4 default OSPF OptV2 Local 421169/0 192.168.10.5,any any,any    ACL1_OSPF
8281   0 [4] IPV4 default OSPF OptV2 Local 0/0      192.168.10.5,any any,any    ACL1_OSPF
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
8339   0 [0] IPV4 red8    OSPF OptV2 Local  0/0     192.168.10.5,any any,any    ACL1_OSPF
8340   0 [1] IPV4 red8    OSPF OptV2 Local  0/0     192.168.10.5,any any,any    ACL1_OSPF
8341   0 [2] IPV4 red8    OSPF OptV2 Local  11099/0 192.168.10.5,any any,any    ACL1_OSPF
8342   0 [1] IPV4 red8    OSPF OptV2 Local  0/0     192.168.10.5,any any,any    ACL1_OSPF
```

```
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
8363   0 [0] IPV4  red7   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8364   0 [1] IPV4  red7   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8365   0 [2] IPV4  red7   OSPF OptV2 Local  11099/0 192.168.10.5,any any,any    ACL1_OSPF
8366   0 [3] IPV4  red7   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
8375   0 [0] IPV4  red6   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8376   0 [1] IPV4  red6   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8377   0 [2] IPV4  red6   OSPF OptV2 Local  11103/0 192.168.10.5,any any,any    ACL1_OSPF
8378   0 [3] IPV4  red6   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
8391   0 [0] IPV4  red5   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8392   0 [1] IPV4  red5   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8393   0 [2] IPV4  red5   OSPF OptV2 Local  11104/0 192.168.10.5,any any,any    ACL1_OSPF
8394   0 [3] IPV4  red5   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
8411   0 [0] IPV4  red4   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8412   0 [1] IPV4  red4   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8413   0 [2] IPV4  red4   OSPF OptV2 Local  11101/0 192.168.10.5,any any,any    ACL1_OSPF
8414   0 [3] IPV4  red4   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
8427   0 [0] IPV4  red3   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8428   0 [1] IPV4  red3   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8429   0 [2] IPV4  red3   OSPF OptV2 Local  11107/0 192.168.10.5,any any,any    ACL1_OSPF
8430   0 [3] IPV4  red3   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
8439   0 [0] IPV4  red1   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8440   0 [1] IPV4  red1   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
8441   0 [2] IPV4  red1   OSPF OptV2 Local  11099/0 192.168.10.5,any any,any    ACL1_OSPF
8442   0 [3] IPV4  red1   OSPF OptV2 Local   0/0    192.168.10.5,any any,any    ACL1_OSPF
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
  .      .     .    .     .    .     .       .         .                .
8276   0 [1] IPV4 default OSPF OptV2 Local  1752/0  any,any         any,any
```

Configure IPv6 ACL-based LPTS Policers

This section describes how you can configure IPv6 ACL-based LPTS policers.

Configuration Example

To configure ACL-based LPTS policers for IPv6 sessions, complete the following configurations:

1. Configure a VRF.
2. Configure an ACL.
3. Configure LPTS policer for a particular IPv6 session matching an ACL and VRF ID.

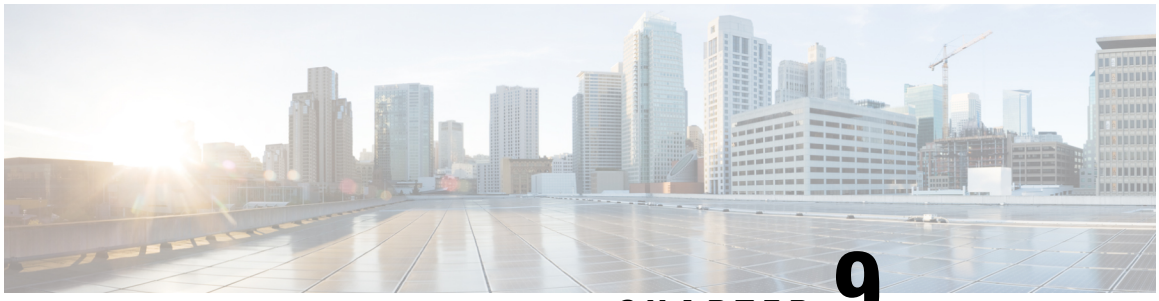
Configuration

```
/* Enter the global configuration mode and configure an VRF */
Router# configure
Router(config)# vrf red83
Router(config)# commit

/* Configure an ACL */
Router(config)# ipv6 access-list ACL33
Router(config-ipv6-acl)# 10 permit ipv6 host 4000:21::1 host 4000:21::2
Router(config-ipv6-acl)# 60 permit ipv6 host 4000:53::1 host 4000:53::2
Router(config-ipv6-acl)# commit
Router(config-ipv6-acl)# exit

/* Configure LPTS policer for a particular IPv6 session matching an ACL and VRF ID. */
Router(config)# lpts pifib hardware police
Router(config-pifib-policer-global)# acl ACL33 rate 5000 vrf red83 ipv6
Router(config-pifib-policer-global)# commit
```

Verification

Use the following command to display the ACL-based LPTS policer entries attached to matching entries:

```
Router#show lpts pifib hardward entry statistics location 0/RP0/CPU0

Offset  NPU   L3    VRF id  L4   Intf Dest Pkts/Drops  laddr,Port    raddr,Port    acl name
------  ----- ---   ------  ---  ---  ----- -------    --------      --------      -----
1498    0 [0] IPV6  red83   TCP  any  Local  383/0   4000:53::1,179  4000:53::1,179   ACL33
1499    0 [1] IPV6  red83   TCP  any  Local  0/0     4000:53::1,179  4000:53::2,43329 ACL33
1450    0 [2] IPV6  red83   TCP  any  Local  0/0     4000:53::1,179  4000:53::2,43329 ACL33
1451    0 [3] IPV6  red83   TCP  any  Local  0/0     4000:53::1,179  4000:53::2,43329 ACL33
```



CHAPTER 9

Implementing VRRP

Configuring VRRP

Table 8: Feature History Table

The Virtual Router Redundancy Protocol (VRRP) feature allows for transparent failover at the first-hop IP router, enabling a group of routers to form a single virtual router. For more information on VRRP and related concepts, see Understanding VRRP, on page 105.

Restrictions and Guidelines for Configuring VRRP

Guidelines for Configuring VRRP

Follow these guidelines for configuring VRRP:

- Upto 64 VRRP sessions are permitted on Cisco NCS 5001 and NCS 5002 routers (including Hot Standby Router Protocol [HSRP] sessions, if any).
- Upto 16 VRRP sessions are permitted on Cisco NCS 5011 routers (including Hot Standby Router Protocol [HSRP] sessions, if any).

Restrictions for Configuring VRRP

These restrictions apply when you configure VRRP:

- ICMP redirects are not supported.
- Protocol Independent Multicast (PIM) is not supported with VRRP.
- VRRP is not supported over BVI interface.

Understanding VRRP

The Virtual Router Redundancy Protocol (VRRP) feature allows for transparent failover at the first-hop IP router, enabling a group of routers to form a single virtual router.

VRRP Overview

A LAN client can use a dynamic process or static configuration to determine which router should be the first hop to a particular remote destination. The client examples of dynamic router discovery are as follows:

- Proxy ARP—The client uses Address Resolution Protocol (ARP) to get the destination it wants to reach, and a router responds to the ARP request with its own MAC address.
- Routing protocol—The client listens to dynamic routing protocol updates (for example, from Routing Information Protocol [RIP]) and forms its own routing table.
- IRDP (ICMP Router Discovery Protocol) client—The client runs an Internet Control Message Protocol (ICMP) router discovery client.

The drawback to dynamic discovery protocols is that they incur some configuration and processing overhead on the LAN client. Also, in the event of a router failure, the process of switching to another router can be slow.

An alternative to dynamic Cisco Discovery Protocols is to statically configure a default router on the client. This approach simplifies client configuration and processing, but creates a single point of failure. If the default gateway fails, the LAN client is limited to communicating only on the local IP network segment and is cut off from the rest of the network.

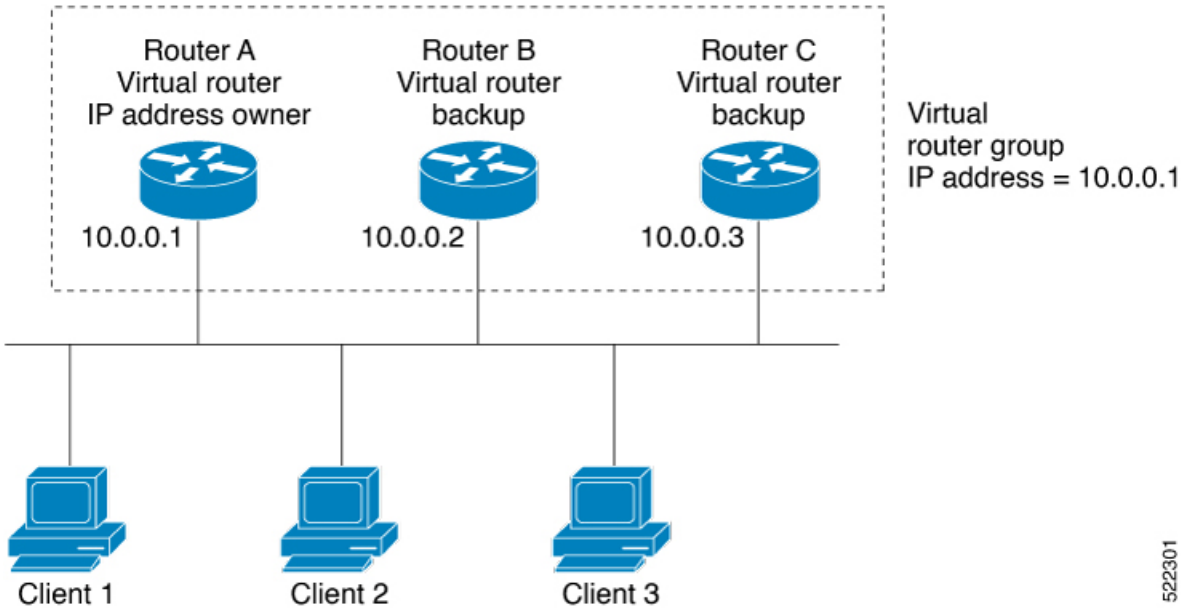
The Virtual Router Redundancy Protocol (VRRP) feature can solve the static configuration problem. VRRP is an IP routing redundancy protocol designed to allow for transparent failover at the first-hop IP router. VRRP enables a group of routers to form a single *virtual router* . The LAN clients can then be configured with the virtual router as their default gateway. The virtual router, representing a group of routers, is also known as a *VRRP group*.

When the virtual router group IP address is the same as the IP address of the physical interface of any router in the VRRP group, then such router becomes the *IP address owner* and the VRRP group operates in the *Owner* mode. When a VRRP group operates in Owner mode, the IP address owner is responsible for forwarding packets that are sent to the VRRP group.

For operating in Owner mode in case of IPv6 VRRP sessions, the link-local address that is configured for the VRRP session must be the same as the link-local address of the physical interface in a router. The link-local address can be autoconfigured by the router or can be an address that is configured by the administrator.

For example, Figure 14: Basic VRRP Topology, on page 107 shows a LAN topology in which VRRP is configured. In this example, Routers A, B, and C are *VRRP routers* (routers running VRRP) that compose a virtual router. The IP address of the virtual router is the same as that configured for the interface of Router A (10.0.0.1).

Figure 14: Basic VRRP Topology



Because the virtual router uses the IP address of the physical interface of Router A, Router A assumes the role of the *IP address owner* and is responsible for forwarding packets that are sent to the VRRP group IP address. Clients 1 through 3 are configured with the default gateway IP address of 10.0.0.1.

Routers B and C function as *backup virtual routers*. If the router that is IP address owner fails, the router that is configured with the higher priority becomes the IP address owner and provides uninterrupted service for the LAN hosts. When Router A recovers, it becomes the IP address owner again.

Note We recommend that you disable Spanning Tree Protocol (STP) on switch ports to which the virtual routers are connected. Enable RSTP or rapid-PVST on the switch interfaces if the switch supports these protocols.

Multiple Virtual Router Support

You can configure up to 255 virtual routers on a router interface. The actual number of virtual routers that a router interface can support depends on the following factors:

- Router processing capability
- Router memory capability
- Router interface support of multiple MAC addresses

In a topology where multiple virtual routers are configured on a router interface, the interface can act as an IP address owner for one or more virtual routers and as a backup for one or more virtual routers.

VRRP Router Priority

An important aspect of the VRRP redundancy scheme is VRRP router priority. Priority determines the role that each VRRP router plays and what happens if the IP address owner virtual router fails.

If a VRRP router owns the IP address of the virtual router and the IP address of the physical interface, this router functions as a IP address owner virtual router.

If no VRRP router owns the IP address, the priority of a VRRP router, combined with the reempt settings, determines if a VRRP router functions as an IP address owner router or a backup virtual router. By default, the highest priority VRRP router functions as IP address owner router, and all the others function as backups. Priority also determines the order of ascendancy to becoming an IP address owner virtual router if the IP address owner virtual router fails. You can configure the priority of each backup virtual router with a value of 1 through 254, using the vrrp priority command.

For example, if Router A, the IP address owner virtual router in a LAN topology, fails, an election process takes place to determine if backup virtual Routers B or C should take over. If Routers B and C are configured with the priorities of 101 and 100, respectively, Router B is elected to become IP address owner virtual router because it has the higher priority. If Routers B and C are both configured with the priority of 100, the backup virtual router with the higher IP address is elected to become the IP address owner virtual router.

By default, a preemptive scheme is enabled whereby a higher-priority backup virtual router that becomes available takes over from the current IP address owner virtual router. You can disable this preemptive scheme using the vrrp preempt disable command. If preemption is disabled, the backup virtual router that is elected to become IP address owner router upon the failure of the original higher priority IP address owner router, remains the IP address owner router even if the original IP address owner virtual router recovers and becomes available again.

VRRP Advertisements

The IP address owner virtual router sends VRRP advertisements to other VRRP routers in the same group. The advertisements communicate the priority and state of the IP address owner virtual router. The VRRP advertisements are encapsulated in IP packets and sent to the IP Version 4 multicast address assigned to the VRRP group. The advertisements are sent every second by default; the interval is configurable.

Benefits of VRRP

The benefits of VRRP are as follows:

- Redundancy— VRRP enables you to configure multiple routers as the default gateway router, which reduces the possibility of a single point of failure in a network.
- Load Sharing—You can configure VRRP in such a way that traffic to and from LAN clients can be shared by multiple routers, thereby sharing the traffic load more equitably among available routers.
- Multiple Virtual Routers—VRRP supports up to 100 virtual routers (VRRP groups) on a router interface, subject to the platform supporting multiple MAC addresses. You can configure up to 256 virtual routers on a router interface. Multiple virtual router support enables you to implement redundancy and load sharing in your LAN topology.
- Multiple IP Addresses—The virtual router can manage multiple IP addresses, including secondary IP addresses. Therefore, if you have multiple subnets configured on an Ethernet interface, you can configure VRRP on each subnet.
- Preemption—The redundancy scheme of VRRP enables you to preempt a backup virtual router that has taken over for a failing IP address owner virtual router with a higher-priority backup virtual router that has become available.
- Text Authentication—You can ensure that VRRP messages received from VRRP routers that comprise a virtual router are authenticated by configuring a simple text password.

- Advertisement Protocol—VRRP uses a dedicated Internet Assigned Numbers Authority (IANA) standard multicast address (224.0.0.18) for VRRP advertisements. This addressing scheme minimizes the number of routers that must service the multicasts and allows test equipment to accurately identify VRRP packets on a segment. The IANA assigns VRRP the IP protocol number 112.

Hot Restartability for VRRP

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

Understanding VRRP over BVI

The Virtual Router Redundancy Protocol (VRRP) protocol provides default gateway redundancy. It allows a group of routers to behave as a single virtual default gateway router in which one router acts as the IP address owner router and others routers act as Backup routers.

BVI (Bridge-Group Virtual Interface) is a virtual interface which provides L3 or routed functionality to a Bridge Group. L2 functionality is applicable to the interfaces which are part of a Bridge Group and BVI is the routed interface for that Bridge Group.

Usually, VRRP sessions run on top of interfaces of the multiple routers which are in the same home network. However, you can configure VRRP session over BVI. Thereby, instead of physical interfaces, VRRP sessions can run between BVI interfaces of multiple routers.

Configuring VRRP for IPv4 Networks

This section describes the procedure for configuring and verifying VRRP for IPv4 networks.

Configuration

Use the following configuration for configuring VRRP for IPv4 networks.

Note Certain customizations (as mentioned) are recommended to control the behavior of the VRRP group on committing the VRRP configuration on the Router. If the following customizations are not configured, then the router seizes control of the VRRP group, and immediately assumes the role of the IP address owner virtual router.

```
/* Enter the interface configuration mode and configure an IPv4 address for the interface.
 */
Router(config)# interface gigabitEthernet 0/0/0/1
Router(config-if)# ipv4 address 10.10.10.1 255.255.255.0
Router(config-if)# no shut
Router(config-if)# commit
Fri Dec  8 13:49:24.142 IST
Router:Dec  8 13:49:24.285 : ifmgr[402]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/1, changed state to Down
Router:Dec  8 13:49:24.711 : ifmgr[402]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/1, changed state to Up

Router(config-if)# exit
Router(config)# do show ip int brief
```

```
Fri Dec  8 13:50:05.505 IST

Interface                        IP-Address      Status          Protocol Vrf-Name
GigabitEthernet0/0/0/0           unassigned      Shutdown        Down     default

GigabitEthernet0/0/0/1           10.10.10.1      Up              Up       default
GigabitEthernet0/0/0/2           unassigned      Shutdown        Down     default

GigabitEthernet0/0/0/3           unassigned      Shutdown        Down     default

GigabitEthernet0/0/0/4           unassigned      Shutdown        Down     default


/* Enter the VRRP configuration mode and add the configured interface. */
Router(config)# router vrrp
Router(config-vrrp)# interface GigabitEthernet 0/0/0/1

/* CUSTOMIZATION: Configure a delay for the startup of the state machine when the interface
 comes up. */
Router(config-vrrp)# delay minimum 2 reload 10 */

/* Configure VRRP version 3 for IPv4 */
Router(config-vrrp-if)# address-family ipv4 vrrp 100 version 3
Router(config-vrrp-virtual-router)# address 10.10.10.1

/* CUSTOMIZATION: Disable the installation of routes for the VRRP virtual addresses. */
Router(config-vrrp-virtual-Router)# accept-mode disable

/* CUSTOMIZATION: Set a priority for the virtual Router. */
Router(config-vrrp-virtual-Router)# priority 254

/* CUSTOMIZATION: Configure a preempt delay value that controls the selection of the IP
address owner virtual Router. */
Router(config-vrrp-virtual-Router)# preempt delay 15

/* CUSTOMIZATION: Configure the interval between successive advertisements by the IP address
 owner virtual Router. */
Router(config-vrrp-virtual-Router)#timer 4

/* CUSTOMIZATION: Configure VRRP to track an interface. */
Router(config-vrrp-virtual-Router)# track interface GigabitEthernet0/0/0/1 30

 /* Commit the configuration */
Router(config-vrrp-virtual-Router)# commit
```

You have successfully configured VRRP for IPv4 networks.

Validation

Use the following commands to validate the configuration.

```
/* Validate the configuration */
Router(config-vrrp-virtual-router)# do show run interface GigabitEthernet 0/0/0/1
Fri Dec  8 15:04:38.140 IST
interface GigabitEthernet0/0/0/1
 ipv4 address 10.10.10.1 255.255.255.0
!

Router(config)# show running-config router vrrp
Fri Dec  8 13:50:18.959 IST
```

```
router vrrp
 interface GigabitEthernet0/0/0/1
   delay minimum 2 reload 10
  address-family ipv4
   vrrp 100 version 3
    priority 254
    preempt delay 15
    timer 4
    track interface GigabitEthernet0/0/0/2 30
    address 10.10.10.1
    accept-mode disable
   !
  !
 !
```

```
Router(config-vrrp-virtual-router)# do show vrrp ipv4 interface gigabitEthernet 0/0/0/1
Fri Dec  8 15:02:56.952 IST
IPv4 Virtual Routers:
                     A indicates IP address owner
                     | P indicates configured to preempt
                     | |
Interface   vrID Prio A P State    Master addr     VRouter addr
Gi0/0/0/1    100  255 A P Master   local           10.10.10.1
```

```
Router(config-vrrp-virtual-router)# end
Router# show vrrp detail
Fri Dec  8 15:08:36.469 IST
GigabitEthernet0/0/0/1 - IPv4 vrID 100
  State is Master, IP address owner
    1 state changes, last state change 01:19:06
    State change history:
    Dec  8 13:49:30.147 IST  Init     -> Master   Delay timer expired
  Last resign sent:     Never
  Last resign received: Never
  Virtual IP address is 10.10.10.1
  Virtual MAC address is 0000.5E00.0164, state is active
  Master router is local
  Version is 3
  Advertise time 1 secs
    Master Down Timer 3.003 (3 x 1 + (1 x 1/256))
  Minimum delay 1 sec, reload delay 5 sec
  Current priority 255
    Configured priority 100, may preempt
      minimum delay 0 secs
```

You have successfully validated VRRP for IPv4 networks.

Configuring VRRP for IPv6 Networks

This section describes the procedure for configuring and verifying VRRP for IPv6 networks.

Configuration

The following sample includes the configuration and customization of VRRP for IPv6 networks.

Note Certain customizations (as mentioned) are recommended to control the behavior of the VRRP group on committing the VRRP configuration on the Router. If the following customizations are not configured, then the Router seizes control of the VRRP group, and immediately assumes the role of the IP address owner virtual Router.

```
/* Enter the interface configuration mode and configure an IPv6 address */
Router# interface GigabitEthernet 0/0/0/2
Router(config-if)# ipv6 address 10::1/64
Router(config-if)# no shut

/* Exit the interface configuration mode and enter the vrrp configuration mode */
Router(config-if)# exit
Router(config)# Router vrrp

/* Add the configured interface for VRRP */
Router(config-vrrp)# interface GigabitEthernet 0/0/0/2

/* CUSTOMIZATION: Configure a delay for the startup of the state machine when the interface
 comes up. */
Router(config-vrrp)# delay minimum 2 reload 10 */

/* Enable the IPv6 global and link local address family on the interface  */
Router(config-vrrp-if)# address-family ipv6 vrrp 50
Router(config-vrrp-virtual-Router)# address linklocal autoconfig

/* CUSTOMIZATION: Disable the installation of routes for the VRRP virtual addresses. */
Router(config-vrrp-virtual-Router)# accept-mode disable

/* CUSTOMIZATION: Set a priority for the virtual Router. */
Router(config-vrrp-virtual-Router)# priority 254

/* CUSTOMIZATION: Configure a preempt delay value that controls the selection of the IP
address owner virtual Router. */
Router(config-vrrp-virtual-Router)# preempt delay 15

/* CUSTOMIZATION: Configure the interval between successive advertisements by the IP address
 owner virtual Router. */
Router(config-vrrp-virtual-Router)#timer 4

/* CUSTOMIZATION: Configure VRRP to track an interface. */
Router(config-vrrp-virtual-Router)# track interface GigabitEthernet0/0/0/2 30

/* Commit the configuration */
Router(config-vrrp-virtual-Router)# commit
```

You have successfully configured VRRP for IPv6 networks.

Validation

Use the following commands to validate the configuration.

```
/* Validate the configuration */
Router(config-vrrp-virtual-router)# do show run interface GigabitEthernet 0/0/0/2
Fri Dec  8 14:55:48.378 IST
interface GigabitEthernet0/0/0/2
 ipv6 address 10::1/64
!
```

```
Router(config-vrrp-virtual-router)# do show running-config router vrrp
...
router vrrp
 interface GigabitEthernet0/0/0/2
  delay minimum 2 reload 10
  address-family ipv6
   vrrp 50
    priority 254
    preempt delay 15
    timer 4
    track interface GigabitEthernet0/0/0/2 30
    address linklocal autoconfig
    accept-mode disable
   !
  !
 !
!
```

```
Router(config-vrrp-virtual-router)# do show vrrp ipv6 interface gigabitEthernet 0/0/0/2
Fri Dec  8 14:59:25.547 IST
IPv6 Virtual Routers:
                     A indicates IP address owner
                     | P indicates configured to preempt
                     | |
Interface   vrID Prio A P State    Master addr     VRouter addr
Gi0/0/0/2      50  254   P Master   local
                                                  fe80::200:5eff:fe00:203
```

```
Router(config-vrrp-virtual-router)# end
Router# show vrrp detail
Fri Dec  8 15:08:36.469 IST
GigabitEthernet0/0/0/2 - IPv6 vrID 50
  State is Master
    2 state changes, last state change 00:18:01
    State change history:
    Dec  8 14:50:23.326 IST  Init     -> Backup   Virtual IP configured
    Dec  8 14:50:35.365 IST  Backup   -> Master   Master down timer expired
  Last resign sent:     Never
  Last resign received: Never
  Virtual IP address is fe80::200:5eff:fe00:203
  Virtual MAC address is 0000.5E00.0203, state is active
  Master router is local

  Advertise time 4 secs
    Master Down Timer 12.031 (3 x 4 + (2 x 4/256))
  Minimum delay 2 sec, reload delay 10 sec
  Current priority 254
    Configured priority 254, may preempt
      minimum delay 15 secs
    Tracked items: 1/1 up: 0 decrement
      Object name               State     Decrement
      GigabitEthernet0/0/0/2        Up            30
```

You have successfully validated VRRP for IPv6 networks.

Unicast VRRP

You can now configure VRRP to support Layer 3 unicast transport, allowing it to enhance its capacity to send data to cloud networks. Pairwise router redundancy enables high availability in cloud network scenarios. The

default route of the cloud native function needs a virtual IP (VIP) address because the paired routers do not have a pre-designated active member. Though HSRP provides a VIP, the cloud networks do not support Layer 2 multicast or broadcast transports. To overcome the limitations of Layer 2 multicast and broadcast transports, configure VRRP in Layer 3 unicast mode to support Layer 3 unicast transport.

This feature also enables VRRP to communicate state transition notifications using event-driven telemetry.

Restrictions for Unicast VRRP

- When you configure the unicast-peer command, the router neither sends nor receives multicast packets.
- You can configure the unicast-peer command only once, allowing for the participation of only two physical routers in a unicast VRRP session.

Configure Unicast VRRP

Configuration Example

The following example shows how to enable unicast transport through VRRP.

```
Router(config)# router vrrp
Router(config-vrrp)# interface GigabitEthernet0/0/0/0
Router(config-vrrp-if)# address-family ipv4
Router(config-vrrp-address-family)# vrrp 1
/* Configure the virtual IP address on the interface. */
Router(config-vrrp-virtual-router)# address 10.0.1.100
/* Configure the unicast-peer command to enable IPv4 unicast transport. */
Router(config-vrrp-virtual-router)# unicast-peer 10.0.1.1
Router(config-vrrp-virtual-router)# exit
Router(config-vrrp-address-family)# exit
Router(config-vrrp-if)# address-family ipv6
Router(config-vrrp-address-family)# vrrp 2
/* Configure the unicast-peer command to enable IPv6 unicast transport.  */
Router(config-vrrp-virtual-router)# unicast-peer FE80::260:3EFF:FE11:6770
Router(config-vrrp-virtual-router)# exit
Router(config-vrrp-address-family)# exit
```

Running Configuration

```
router vrrp
 interface GigabitEthernet0/0/0/0
  address-family ipv4
   vrrp 1
    address 10.0.0.100
    unicast-peer 10.0.1.1
   !
  !
  address-family ipv6
   vrrp 2
    unicast-peer  FE80::260:3EFF:FE11:6770
   !
  !
 !
!
```

Verification

Use the following command to verify if the unicast transport enabled in VRRP. The output shows that both IPv4 and IPv6 unicast peers have been configured, and the respective IP addresses are displayed.

```
Router# show vrrp detail
Fri Sep  8 15:02:35.268 IST
GigabitEthernet0/0/0/0 - IPv4 vrID 1
  State is Master
    2 state changes, last state change 04:00:02
    State change history:
    Sep  8 11:02:29.518 IST  Init     -> Backup   Virtual IP configured
    Sep  8 11:02:33.127 IST  Backup   -> Master   Master down timer expired
  Last resign sent:     Never
  Last resign received: Never
  Virtual IP address is 10.0.0.100
  Virtual MAC address is 0000.5E00.0101, state is active
  Master router is local
  Version is 2
  Advertise time 1 secs
    Master Down Timer 3.609 (3 x 1 + (156 x 1/256))
  Minimum delay 1 sec, reload delay 5 sec
  Current priority 100
    Configured priority 100, may preempt
      minimum delay 0 secs
  IPv4 Unicast Peer: 10.0.1.1 --> IPv4 unicast transport is enabled on VRRP.

GigabitEthernet0/0/0/0 - IPv6 vrID 2
  State is Init
    0 state changes, last state change never
    State change history:
  Last resign sent:     Never
  Last resign received: Never
  Virtual IP address is ::
  Virtual MAC address is 0000.5E00.0202, state is stored
  Master router is unknown
  Version is 3
  Advertise time 1 secs
    Master Down Timer 3.609 (3 x 1 + (156 x 1/256))
  Minimum delay 1 sec, reload delay 5 sec
  Current priority 100
    Configured priority 100, may preempt
      minimum delay 0 secs
  IPv6 Unicast Peer: FE80::260:3EFF:FE11:6770 --> IPv6 unicast transport is enabled on VRRP.
```

Use the following command to verify detailed statistics about the Virtual Router VRRP configuration. Note that the number of multicast packets received in the VRRP instance when it's configured to function in unicast mode is zero.

```
Router# show vrrp statistics
Fri Sep  8 15:03:03.521 IST
Invalid packets:
  Invalid checksum:                     0
  Unknown/unsupported versions:         0
  Invalid vrID:                         0
  Too short:                            0
Protocol:
  Transitions to Master                 1
Packets:
  Total received:                       0
  Adverts sent:                     14476
  Bad TTL:                              0
  Short Packets:                        0
  Failed authentication:                0
  Unknown authentication:               0
  Conflicting authentication:           0
  Unknown Type field:                   0
```

```
  Conflicting Advertise time:           0
  Conflicting Addresses:                0
  Received with zero priority:          0
  Sent with zero priority:              0
 Mcast packet in Ucast mode:           0 --> Multicast packet being received in unicast
mode.
```

Configure VRRP over BVI

To configure VRRP sessions over BVI, you must complete the following configurations:

1. Configure a set of interfaces as L2 interfaces and a set of VLAN sub-interfaces.
2. Configure a bridge group.
3. Configure a BVI.
4. Configure VRRP over BVI.

Configuration Example

```
/* Enter the global configuration mode and Configure a set of interfaces as L2 interfaces
and a set of VLAN sub-interfaces */
Router# configure
Router(config)# interface HundredGigE0/0/1/0.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# commit
Router(config-subif)# exit
Router(config)# interface HundredGigE0/0/1/1.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# commit
Router(config-subif)# exit

/* Enter the Layer 2 VPN configuration mode and Configure a bridge group */
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 5
Router(config-l2vpn-bg)# bridge-domain 5
Router(config-l2vpn-bg-bd)# interface HundredGigE 0/0/0/0.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface HundredGigE 0/0/0/1.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# routed interface BVI 10
Router(config-l2vpn-bg-bd-bvi)# commit
Router(config-l2vpn-bg-bd-bvi)# exit

/* Configure a BVI in the global configuration mode*/
Router(config-l2vpn-bg-bd)# interface BVI 10

Router(config-if)# ipv4 address 209.165.200.225 255.255.255.0
Router(config-if)# ipv6 address 2001:DB8:A:B::1/64
Router(config-if)# commit

/* Configure VRRP over BVI in the global configuration mode for IPv4 address*/
Router(config)# router VRRP
Router(config-vrrp)# interface BVI 10
Router(config-vrrp-if)# address-family ipv4
Router(config-vrrp-address-family)# VRRP 10
```

```
Router(config-vrrp-virtual-router)# priority 101
Router(config-vrrp-virtual-router)# address 209.165.200.226
Router(config-vrrp-virtual-router)# commit


/* Configure VRRP over BVI in the global configuration mode for IPv6 address*/
Router(config)# router VRRP
Router(config-vrrp)# interface BVI 10
Router(config-vrrp-if)# address-family ipv6
Router(config-vrrp-address-family)# VRRP 11
Router(config-vrrp-virtual-router)# address global 2001:DB8:A:B::2
Router(config-vrrp-virtual-router)#  address linklocal autoconfig
Router(config-vrrp-virtual-router)# commit
```

Verification

Use the following command to verify the bridge domain details:

```
Router# show l2vpn bridge-domain detail

Legend: pp = Partially Programmed.
Bridge group: 5, bridge-domain: 5, id: 1, state: up, ShgId: 0, MSTi: 0
Coupled state: disabled
VINE state: BVI Resolved
MAC learning: enabled
MAC withdraw: enabled
MAC withdraw for Access PW: enabled
MAC withdraw sent on: bridge port up
MAC withdraw relaying (access to access): disabled
Flooding:
Broadcast & Multicast: enabled
Unknown unicast: enabled
MAC aging time: 300 s, Type: inactivity
MAC limit: 32768, Action: none, Notification: syslog
MAC limit reached: no, threshold: 75%
MAC port down flush: enabled
MAC Secure: disabled, Logging: disabled
Split Horizon Group: none
Dynamic ARP Inspection: disabled, Logging: disabled
IP Source Guard: disabled, Logging: disabled
DHCPv4 Snooping: disabled
DHCPv4 Snooping profile: none
IGMP Snooping: disabled
IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control: disabled
Bridge MTU: 1500
MIB cvplsConfigIndex: 2
Filter MAC addresses:
P2MP PW: disabled
Multicast Source: Not Set
Create time: 26/05/2020 17:08:54 (00:11:30 ago)
No status change since creation
ACs: 3 (3 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
List of ACs:
AC: BVI10, state is up
Type Routed-Interface
MTU 1514; XC ID 0x80000001; interworking none
BVI MAC address:
c472.95a6.8b90
Virtual MAC addresses:
0000.5e00.010a
0000.5e00.020b
```

```
Split Horizon Group: Access
AC: HundredGigE0/0/1/0.1, state is up
Type VLAN; Num Ranges: 1
Rewrite Tags: []
VLAN ranges: [1, 1]
MTU 1500; XC ID 0x1; interworking none
MAC learning: enabled
Flooding:
Broadcast & Multicast: enabled
Unknown unicast: enabled
MAC aging time: 300 s, Type: inactivity
MAC limit: 32768, Action: none, Notification: syslog
MAC limit reached: no, threshold: 75%
MAC port down flush: enabled
MAC Secure: disabled, Logging: disabled
Split Horizon Group: none
E-Tree: Root
Dynamic ARP Inspection: disabled, Logging: disabled
IP Source Guard: disabled, Logging: disabled
DHCPv4 Snooping: disabled
DHCPv4 Snooping profile: none
IGMP Snooping: disabled
IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control: bridge-domain policer
Static MAC addresses:
Statistics:
packets: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 1435
bytes: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 114828
MAC move: 0
Storm control drop counters:
packets: broadcast 0, multicast 0, unknown unicast 0
bytes: broadcast 0, multicast 0, unknown unicast 0
Dynamic ARP inspection drop counters:
packets: 0, bytes: 0
IP source guard drop counters:
packets: 0, bytes: 0
AC: HundredGigE0/0/1/1.1, state is up
Type VLAN; Num Ranges: 1
Rewrite Tags: []
VLAN ranges: [1, 1]
MTU 1500; XC ID 0x2; interworking none
MAC learning: enabled
Flooding:
Broadcast & Multicast: enabled
Unknown unicast: enabled
MAC aging time: 300 s, Type: inactivity
MAC limit: 32768, Action: none, Notification: syslog
MAC limit reached: no, threshold: 75%
MAC port down flush: enabled
MAC Secure: disabled, Logging: disabled
Split Horizon Group: none
E-Tree: Root
Dynamic ARP Inspection: disabled, Logging: disabled
IP Source Guard: disabled, Logging: disabled
DHCPv4 Snooping: disabled
DHCPv4 Snooping profile: none
IGMP Snooping: disabled
IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control: bridge-domain policer
Static MAC addresses:
Statistics:
packets: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 1435
```

```
bytes: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 114828
MAC move: 0
Storm control drop counters:
packets: broadcast 0, multicast 0, unknown unicast 0
bytes: broadcast 0, multicast 0, unknown unicast 0
Dynamic ARP inspection drop counters:
packets: 0, bytes: 0
IP source guard drop counters:
packets: 0, bytes: 0
List of Access PWs:
List of VFIs:
List of Access VFIs:
```

Use the following command to show the VRRP details:

```
Router# show vrrp ipv4 detail

BVI10 - IPv4 vrID 10
State is Master
2 state changes, last state change 00:11:57
State change history:
May 26 17:08:59.470 UTC Init -> Backup Delay timer expired
May 26 17:09:03.075 UTC Backup -> Master Master down timer expired
Last resign sent: Never
Last resign received: Never
Virtual IP address is 209.165.200.226
Virtual MAC address is 0000.5E00.010a, state is active
Master router is local
Version is 2
Advertise time 1 secs
Master Down Timer 3.605 (3 x 1 + (155 x 1/256))
Minimum delay 1 sec, reload delay 5 sec
Current priority 101
Configured priority 101, may preempt
minimum delay 0 secs
```

```
Router# show vrrp ipv6 detail

BVI10 - IPv6 vrID 11
State is Master
2 state changes, last state change 00:04:29
State change history:
May 26 17:16:43.476 UTC Init -> Backup Virtual IP configured
May 26 17:16:47.085 UTC Backup -> Master Master down timer expired
Last resign sent: Never
Last resign received: Never
Virtual IP address is fe80::200:5eff:fe00:20b
Secondary Virtual IP address is 2001:db8:a:b::2
Virtual MAC address is 0000.5E00.020b, state is active
Master router is local
Version is 3
Advertise time 1 secs
Master Down Timer 3.609 (3 x 1 + (156 x 1/256))
Minimum delay 1 sec, reload delay 5 sec
Current priority 100
Configured priority 100, may preempt
minimum delay 0 secs
```

```
Router# show vrrp interface BVI10 detail
BVI10 - IPv4 vrID 10
State is Master
2 state changes, last state change 00:12:35
State change history:
May 26 17:08:59.470 UTC Init -> Backup Delay timer expired
May 26 17:09:03.075 UTC Backup -> Master Master down timer expired
```

```
Last resign sent: Never
Last resign received: Never
Virtual IP address is 209.165.200.226
Virtual MAC address is 0000.5E00.010a, state is active
Master router is local
Version is 2
Advertise time 1 secs
Master Down Timer 3.605 (3 x 1 + (155 x 1/256))
Minimum delay 1 sec, reload delay 5 sec
Current priority 101
Configured priority 101, may preempt
minimum delay 0 secs

BVI10 - IPv6 vrID 11
State is Master
2 state changes, last state change 00:04:51
State change history:
May 26 17:16:43.476 UTC Init -> Backup Virtual IP configured
May 26 17:16:47.085 UTC Backup -> Master Master down timer expired
Last resign sent: Never
Last resign received: Never
Virtual IP address is fe80::200:5eff:fe00:20b
Secondary Virtual IP address is 2001:db8:a:b::2
Virtual MAC address is 0000.5E00.020b, state is active
Master router is local
Version is 3
Advertise time 1 secs
Master Down Timer 3.609 (3 x 1 + (156 x 1/256))
Minimum delay 1 sec, reload delay 5 sec
Current priority 100
Configured priority 100, may preempt
minimum delay 0 secs
```

BFD for VRRP

Bidirectional Forwarding Detection (BFD) is a network protocol used to detect faults between two forwarding engines. BFD sessions operate in asynchronous mode. In asynchronous mode, both endpoints periodically send hello packets to each other. If a number of those packets are not received, the session is considered down.

Advantages of BFD

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

BFD Process

VRRP uses BFD to detect a link failure and facilitate fast failover times without excessive control packet overhead.

The VRRP process creates BFD sessions as required. When a BFD session goes down, each backup group monitoring the session transitions to the active state.

After a transition to active state triggered by a BFD session going down, VRRP does not participate in any state elections for 10 seconds.

Note IPv4 only supports BFD for VRRP.

Configuring BFD

Enabling BFD

```
Router# configure
Router(config)#  router vrrp
Router(config-vrrp)# interface <type> <interface-path-id>
Router(config-vrrp-if)# address-family ipv4
Router(config-vrrp-ipv4)# vrrp[group number] version <version-no> bfd
fast-detect [peer ipv4 <ipv4-address> <interface-type> <interface-path-id>]
commit
```

Verifying BFD on VRRP

```
router vrrp
interface TenGigE0/0/0/3.1
bfd minimum-interval 4
bfd multiplier 3
address-family ipv4
vrrp 1
priority 200
address 41.41.1.3
bfd fast-detect peer ipv4 41.41.1.2
```

Modifying BFD timers (minimum interval)

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

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

Modifying BFD timers (multiplier)

Multiplier is the number of consecutive BFD packets which must be missed from a BFD peer before declaring that peer unavailable. The default multiplier is 3.

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

Disabling State Change Logging

Configuration Example

Disables the task of logging the VRRP state change events via syslog.

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#message state disable
router(config-vrrp)#commit
```

Enabling Multiple Group Optimization (MGO) for VRRP

Configuration Examples

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

Configuring VRRP Session Name

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2
router(config-vrrp-if)#address-family ipv4
router(config-vrrp-address-family)#vrrp 1
/* Enables VRRP group configuration mode on a specific interface. */

router(config-vrrp-vritual-router)#name m1
/* Specifies the VRRP session name. */

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

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

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2
router(config-vrrp-if)#address-family ipv4
```

```
router(config-vrrp-address-family)#vrrp 2 slave
/* Enables VRRP slave configuration mode on a specific interface. */

router(config-vrrp-slave)#follow m1
/* Instructs the subordinate group to inherit its state from the specified group, m1 (MGO
session name). */

router(config-vrrp-slave)#address 10.2.3.2
/* Specifies the primary virtual IPv4 address for subordinate group. */

router(config-vrrp-slave)#address 10.2.3.3 secondary
/* Specifies the secondary virtual IPv4 address for subordinate group. */

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

Primary and Secondary Virtual IPv4 Addresses for the Subordinate Group

```
Router#configure
Router(config)#router vrrp
router(config-vrrp)#interface TenGigE 0/0/0/2
router(config-vrrp-if)#address-family ipv4

router(config-vrrp-address-family)#vrrp 2 slave
/* Enables VRRP slave configuration mode on a specific interface. */

router(config-vrrp-slave)#address 10.2.3.2
/* Specifies the primary virtual IPv4 address for subordinate group. */

router(config-vrrp-slave)#address 10.2.3.3 secondary
/* Specifies the secondary virtual IPv4 address for subordinate group. */

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

Running Configuration

```
Router#show running-config router vrrp 1
router vrrp
interface TenGigE 0/0/0/2
address-family ipv4
vrrp 1
name m1
!

/* Subordinate group */
Router#show running-config router vrrp 2
router vrrp
interface TenGigE 0/0/0/2
address-family ipv4
vrrp 2 slave
follow m1
address 10.2.3.2
address 10.2.3.3 secondary
!
```

Configuring SNMP Server Notifications for VRRP Events

MIB support for VRRP

VRRP enables one or more IP addresses to be assumed by a router when a failure occurs. For example, when IP traffic from a host reaches a failed router because the failed router is the default gateway, the traffic is transparently forwarded by the VRRP router that has assumed control. VRRP does not require configuration of dynamic routing or router discovery protocols on every end host. The VRRP router controlling the IP address(es) associated with a virtual router is called the IP address owner router, and forwards packets sent to these IP addresses. The election process provides dynamic fail over (standby) in the forwarding responsibility should the IP address owner router become unavailable. This allows any of the virtual router IP addresses on the LAN to be used as the default first hop router by end-hosts.

The advantage gained from using VRRP is a higher availability default path without requiring configuration of dynamic routing or router discovery protocols on every end-host. Simple Network Management Protocol (SNMP) traps provide information of the state changes, when the virtual routers (in standby) are moved to IP address owner router's state or if the standby router is made IP address owner router.

Configuration Example

Enables SNMP server notifications (traps) for VRRP.

```
Router#configure
Router(config)#snmp-server traps vrrp events
router(config)#commit
```

Use the **show snmp traps details** command to view details of SNMP server notifications.