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

Notwithstanding any other warranty herein, all document files and software of these suppliers are provided “as is” with all faults. Cisco and the above-named suppliers disclaim all warranties, expressed or implied, including, without limitation, those of merchantability, fitness for a particular purpose and noninfringement or arising from a course of dealing, usage, or trade practice.

In no event shall Cisco or its suppliers be liable for any indirect, special, consequential, or incidental damages, including, without limitation, lost profits or loss of data arising out of the use or inability to use this manual, even if Cisco or its suppliers have been advised of the possibility of such damages.

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

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

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

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

© 2019 Cisco Systems, Inc. All rights reserved.
## CONTENTS

### CHAPTER 1: Implementing Network Stack IPv4 and IPv6

- Implementing Fallback VRF 1
- Network Stack IPv4 and IPv6 Exceptions 2
- IPv4 and IPv6 Functionality 2
- IPv6 for Cisco IOS XR Software 3

#### How to Implement Network Stack IPv4 and IPv6

- Configuring IPv4 Addressing 3
- Configuring IPv6 Addressing 4
- IPv6 Multicast Groups 4
- Assigning Multiple IP Addresses to Network Interfaces 9
- Configuring IPv4 and IPv6 Protocol Stacks 10
- Enabling IPv4 Processing on an Unnumbered Interface 11
- IPv4 ICMP Rate Limiting 13
- IPv6 ICMP Rate Limiting 14

#### Selecting Flexible Source IP

- Configuring IPARM Conflict Resolution 15
  - Static Policy Resolution 15
  - Longest Prefix Address Conflict Resolution 16
  - Highest IP Address Conflict Resolution 17
  - Route-Tag Support for Connected Routes 17

#### Larger IPv6 Address Space

- IPv6 Address Formats 18
- IPv6 Address Type: Unicast 20
  - Aggregatable Global Address 20
  - Link-Local Address 21
  - IPv4-Compatible IPv6 Address 21
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.

• Implementing Fallback VRF, on page 1
• Network Stack IPv4 and IPv6 Exceptions, on page 2
• IPv4 and IPv6 Functionality, on page 2
• IPv6 for Cisco IOS XR Software, on page 3
• How to Implement Network Stack IPv4 and IPv6, on page 3

Implementing Fallback VRF

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

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

Restrictions

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

• You can configure only one fallback VRF route for each address family of each primary VRF.
• Ping, traceroute, or any slow path application is not supported on fallback VRF because there is no support for LPTS receive trap.
• Only 1000 VRFs and 1 global table are supported in Cisco NCS 560 Series Routers.
• If you configure a static default route to a VRF, the static default route takes precedence over the fallback VRF. If you configure the default route for a VRF, the global routing table is used for a route lookup. The default route is always directed to the configured next hop.
• If a route lookup for a packet fails in the primary VRF, the packet is recycled to do route lookup in the 
fallback VRF. Therefore, the routing performance of the packet goes down by up to 50 percent.

• If you configure both ACL-based forwarding (ABF) VRF redirect and VRF fallback for a packet, then 
the packet is recycled twice. Therefore, the routing performance of the packet goes down by up to 33 
percent.

• If a route for a packet is found in the fallback VRF, only the Glean IPv4 and Glean IPv6 adjacency 
packets are punted successfully.

• In a looped configuration, if the route for a packet is not found in both the primary and fallback VRF, 
the packet loops in the recycle path. Eventually, the packet is dropped in the recycle egress queue. The 
recycle queue is of highest priority. Therefore, if there is a high rate of looped traffic, other good recycled 
packets may be dropped.

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 scavenger-timeout command sets the lifetime for neighbor entries in the stale state. When 
the scavenger-timer for a neighbor entry expires, the entry is cleared.

• In Cisco IOS XR software, the show ipv4 interface and show ipv6 interface commands include the 
location node-id keyword. If a location is specified, only the interface entries in the specified location 
are displayed.

• Cisco IOS XR software allows conflicting IP address entries at the time of configuration. If an IP address 
conflict exists between two interfaces that are active, Cisco IOS XR software brings down the interface 
according to the configured conflict policy, the default policy being to bring down the higher interface 
instance.

IPv4 and IPv6 Functionality

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

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

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

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

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 *interface HundredGigE 0/9/0/1*.

Note

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

Router#configure HundredGigE0/9/0/1
Router(config)#interface HundredGigE 0/9/0/1
Router(config-if)#ipv4 address 192.168.1.27/8
Router(config-if)#commit
Running Configuration

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

Verification

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

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

interface HundredGigE0/9/0/1 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.168.1.27/8
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Multicast reserved groups joined: 224.0.0.2 224.0.0.1
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000

Associated Commands

• ipv4 address
• show ipv4 interface

Configuring IPv6 Addressing

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

Note

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

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

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

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

Configuration Example

An IPv6 address of 2001:0DB8:0:1::1/64 is assigned to the interface HundredGigE 0/9/0/1:

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

Running Configuration

```
Router#show running-config interface HundredGigE0/9/0/1
interface HundredGigE0/9/0/1
ipv4 address 192.168.1.27 255.0.0.0
ipv4 address 1.0.0.1 255.255.255.0 secondary
ipv4 address 2.0.0.1 255.255.255.0 secondary
ipv6 address 2001:db8:0:1::1/64
!
```

Verification

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

```
Router#show ipv6 interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
Global unicast address(es):
  2001:db8:0:1::1, subnet is 2001:db8:0:1::/64
Joined group address(es):
  ff02::1::ff00:1 ff02::1::ffa6:1c75 ff02::2
  ff02::1
MTU is 1514 (1500 is available to IPv6)
ICMP redirects are disabled
ICMP unreachables are enabled
ND DAD is enabled, number of DAD attempts 1
ND reachable time is 0 milliseconds
ND cache entry limit is 100000000
ND advertised retransmit interval is 0 milliseconds
Hosts use stateless autoconfig for addresses.
Outgoing access list is not set
Inbound access list is not set
Table Id is 0xe0800000
Complete protocol adjacency: 0
Complete glean adjacency: 0
Incomplete protocol adjacency: 0
Incomplete glean adjacency: 0
```
Dropped protocol request: 0  
Dropped glean request: 0

**Associated Commands**

- ipv6 address
- interface
- show ipv6 interface

**Configuration Example**

An IPv6 address of 2001:0DB8:0:1::/64 is assigned to the `interface HundredGigE 0/9/0/1`. The `eui-64` keyword configures site-local and global IPv6 addresses with an interface identifier (ID) in the low-order 64 bits of the IPv6 address. Only the 64-bit network prefix for the address needs to be specified; the last 64 bits are automatically computed from the interface ID.

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

**Running Configuration**

```
Router#show running-config interface HundredGigE0/9/0/1
interface HundredGigE0/9/0/1
  ipv4 address 192.168.1.27 255.0.0.0
  ipv4 address 1.0.0.1 255.255.255.0 secondary
  ipv4 address 2.0.0.1 255.255.255.0 secondary
  ipv6 address 2001:db8:0:1::/64 eui-64
```

**Verification**

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

```
Router#show ipv6 interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv6 protocol is Up, VrfId is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
  Global unicast address(es):
    2001:db8:0:1::c672:95ff:fea6:1c75, subnet is 2001:db8:0:1::/64
  Joined group address(es): ff02::1:ffa6:1c75 ff02::2 ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 100000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
```
**Associated Commands**

- ipv6 address
- interface
- show ipv6 interface

**Configuration Example**

An IPv6 address of FE80::260:3EFF:FE11:6770 is assigned to the `interface HundredGigE 0/9/0/1`. The link-local keyword configures a link-local address on the interface that is used instead of the link-local address that is automatically configured when IPv6 is enabled on the interface.

```
Router#configure
Router(config)#interface HundredGigE 0/9/0/1
Router(config-if)#ipv6 address FE80::260:3EFF:FE11:6770 link-local
Router(config-if)#commit
```

**Running Configuration**

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

interface HundredGigE0/9/0/1
  ipv6 address fe80::260:3eff:fe11:6770 link-local

```

**Verification**

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

```
Router#show ipv6 interface HundredGigE0/9/0/1

HundredGigE0/9/0/1 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
IPv6 is enabled, link-local address is fe80::260:3eff:fe11:6770
Global unicast address(es):
  2001:db8:0:1::260:3eff:fe11:6770, subnet is 2001:db8:0:1::/64
Joined group address(es): ff02::1:ff11:6770 ff02::2 ff02::1
MTU is 1514 (1500 is available to IPv6)
ICMP redirects are disabled
ICMP unreachables are enabled
ND DAD is enabled, number of DAD attempts 1
ND reachable time is 0 milliseconds
ND cache entry limit is 1000000000
ND advertised retransmit interval is 0 milliseconds
Hosts use stateless autoconfig for addresses.
Outgoing access list is not set
Inbound access list is not set
Table Id is 0xe0800000
Complete protocol adjacency: 0
Complete glean adjacency: 0
Incomplete protocol adjacency: 0
Incomplete glean adjacency: 0
Dropped protocol request: 0
Dropped glean request: 0
```

**Associated Commands**

- ipv6 address
- interface
• show ipv6 interface

**Configuration Example**

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

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

**Running Configuration**

```
Router#show running-config interface HundredGigE0/9/0/1
interface HundredGigE0/9/0/1
ipv6 enable
!
```

**Verification**

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

```
Router#show ipv6 interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
No global unicast address is configured
Joined group address(es): ff02::1:ffa6:1c75 ff02::2 ff02::1
MTU is 1514 (1500 is available to IPv6)
ICMP redirects are disabled
ICMP unreachables are enabled
ND DAD is enabled, number of DAD attempts 1
ND reachable time is 0 milliseconds
ND cache entry limit is 1000000000
ND advertised retransmit interval is 0 milliseconds
Hosts use stateless autoconfig for addresses.
Outgoing access list is not set
Inbound access list is not set
Table Id is 0xe0800000
Complete protocol adjacency: 0
Complete glean adjacency: 0
Incomplete protocol adjacency: 0
Incomplete glean adjacency: 0
Dropped protocol request: 0
Dropped glean request: 0
```

**Associated Commands**

- ipv6 enable
- interface
- show ipv6 interface
Assigning Multiple IP Addresses to Network Interfaces

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

- There might not be enough host addresses for a particular network segment. For example, suppose your subnetting allows up to 254 hosts per logical subnet, but on one physical subnet you must have 300 host addresses. Using secondary IP addresses on the routers or access servers allows you to have two logical subnets using one physical subnet.

- Many older networks were built using Level 2 bridges, and were not subnetted. The judicious use of secondary addresses can aid in the transition to a subnetted, router-based network. Routers on an older, bridged segment can easily be made aware that many subnets are on that segment.

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

**Note**

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

**Caution**

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

**Configuration Example**

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

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

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

**Running Configuration**

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

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

**Verification**

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

HundredGigE0/9/0/1 is Up, ipv4 protocol is Up
Vrf is default (vrfid 0x60000000)
Internet address is unassigned
Secondary address 192.168.1.27/24
MTU is 1514 (1500 is available to IP)
```
Helper address is not set
Multicast reserved groups joined: 224.0.0.2 224.0.0.1
Directed broadcast forwarding is disabled
Outgoing access list is not set
Inbound access list is not set
Proxy ARP is disabled
ICMP redirects are never sent
ICMP unreachables are always sent
ICMP mask replies are never sent
Table Id is 0xe0000000

Associated Commands
• ipv4 address
• show ipv4 interface

Configuring IPv4 and IPv6 Protocol Stacks

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

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

Configuration Example

An IPv4 address of 192.168.99.1 and an IPv6 address of 2001:0DB8:c18:1::3/64 is configured on the interface HundredGigE 0/9/0/1.

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

Running Configuration

Router# show running-config interface HundredGigE0/9/0/1
interface HundredGigE0/9/0/1
ipv4 address 192.168.99.1 255.255.255.0
ipv6 address 2001:0DB8:c18:1::3/64
!

Verification

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

Router# show ipv4 interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv4 protocol is Up
Vrf is default (vrfid 0x60000000)
Internet address is 192.168.99.1/24
MTU is 1514 (1500 is available to IP)
Helper address is not set
Multicast reserved groups joined: 224.0.0.2 224.0.0.1
Directed broadcast forwarding is disabled
Outgoing access list is not set
Inbound access list is not set
Proxy ARP is disabled
ICMP redirects are never sent
ICMP unreachables are always sent
ICMP mask replies are never sent
Table Id is 0xe0000000

```
Router# show ipv6 interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
IPv6 is enabled, link-local address is fe80::c672:95ff:fea6:1c75
Global unicast address(es):
  2001:db8:c18:1::3, subnet is 2001:db8:c18:1::/64
Joined group address(es):
  ff02::1:ff00:3 ff02::1:ff01:1c75 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 100000000
ND advertised retransmit interval is 0 milliseconds
Hosts use stateless autoconfig for addresses.
Outgoing access list is not set
Inbound access list is not set
Table Id is 0xe0800000
Complete protocol adjacency: 0
Complete glean adjacency: 0
Incomplete protocol adjacency: 0
Incomplete glean adjacency: 0
Dropped protocol request: 0
Dropped glean request: 0
```

Associated Commands
- `ipv4 address`
- `ipv6 address`
- `show ipv4 interface`
- `show ipv6 interface`

### Enabling IPv4 Processing on an Unnumbered Interface

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

- Interfaces using High-Level Data Link Control (HDLC), PPP, and Frame Relay encapsulations can be unnumbered. Serial interfaces using Frame Relay encapsulation can also be unnumbered, but the interface must be a point-to-point sub-interface.
- You cannot use the `ping` EXEC command to determine whether the interface is up, because the interface has no IP address. The Simple Network Management Protocol (SNMP) can be used to remotely monitor interface status.
- You cannot support IP security options on an unnumbered interface.
If you are configuring Intermediate System-to-Intermediate System (IS-IS) across a serial line, you should configure the serial interfaces as unnumbered, which allows you to conform with RFC 1195, which states that IP addresses are not required on each interface.

Configuration Example

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

```
Router#configure
Router(config)#interface HundredGigE 0/9/0/1
Router(config-if)#ipv4 unnumbered loopback 0
Router(config-if)#commit
```

Running Configuration

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

Verification

```
Router#show interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 is up, line protocol is up
   Interface state transitions: 5
   Hardware is Hundredgige, address is 00e2.2a33.445b (bia 00e2.2a33.445b)
   Layer 1 Transport Mode is LAN
   Internet address is 10.0.0.2/32
   MTU 1514 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
   Encapsulation ARPA,
   Full-duplex, 10000Mb/s, link type is force-up
   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
   1051401410 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
IPv4 ICMP Rate Limiting

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

Configuration Example

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

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

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

Running Configuration

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

Verification

```
Router# show ipv4 interface HundredGigE0/9/0/1
HundredGigE0/9/0/1 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 192.85.1.1/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Multicast reserved groups joined: 224.0.0.2 224.0.0.1 224.0.0.2
                                             224.0.0.5 224.0.0.6
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound common access list is not set, access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachable are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
```

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

```
Router# show ipv4 traffic
ICMP statistics:
  Sent: 0 admin unreachable, 5 network unreachable
        0 host unreachable, 0 protocol unreachable
        0 port unreachable, 0 fragment unreachable
        0 time to live exceeded, 0 reassembly ttl exceeded
```
IPv6 ICMP Rate Limiting

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

Configuration Example

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

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

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

Running Configuration

Router# show running-config
Building configuration...
!! IOS XR Configuration version = 6.0.0.26I
!! Last configuration change at Mon Dec 14 22:07:35 2015 by root
hostname test-83
logging console debugging
username root
group root-lr
group cisco-support
secret 5 $1$d2NC$RbAdqdU7kw/kEJoMP/IJG1
!
cdp
ipv6 icmp error-interval 50 20
icmp ipv4 rate-limit unreachable DF 1000
icmp_ipv4 rate-limit unreachable 1000
ipv4 conflict-policy static

Associated Commands
- ipv6 icmp error-interval

Selecting Flexible Source IP

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

Configuration Example

Enables RFC compliance for source address selection.

```
Router#configure
Router(config)#icmp_ipv4 source rfc
Router(config)#commit
```

Running Configuration

```
Router#show running-config | in source rfc
Building configuration...
icmp_ipv4 source rfc
```

Associated Commands

Configuring IPARM Conflict Resolution

This task sets the IP Address Repository Manager (IPARM) address conflict resolution parameters:
- Static Policy Resolution
- Longest Prefix Address Conflict Resolution
- Highest IP Address Conflict Resolution
- Route-Tag Support for Connected Routes

Static Policy Resolution

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

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

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

Running Configuration

Router#show running-config | in ipv4 config
Building configuration...
!! IOS XR Configuration version = 6.0.0.26I
!! Last configuration change at Mon Dec 14 21:57:27 2015 by root
!
hostname sample-83/logging console debugging
username root
group root-lr
group test
secret 5 $1$d2NC$RbAdqdU7kw/eKJpMo/GJI1
!
cdp
ipv4 conflict-policy static
interface Loopback0
  ipv4 address 1.1.1.1 255.255.255.255
%
...

Verification

Router#show arm ipv4 conflicts
F Forced down
| Down interface & addr Up interface & addr VRF
F tenGigE 0/11/0/0 192.85.1.2/24 HundredGigE0/9/0/1 192.85.1.1/24 default

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
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.

Route# 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 longest-prefix
Building configuration...
ipv4 conflict-policy longest-prefix

Verification

Router# show arm ipv4 conflicts
F Forced down
| Down interface & addr | Up interface & addr | VRF
---|---|---
F tenGigE 0/11/0/0 192.85.1.2/24 HundredGigE0/9/0/1 192.85.1.1/24 default

Route-Tag Support for Connected Routes

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

Specifies an IPv4 address 10.0.54.2/30 that has a route tag of 20 to the interface HundredGigE 0/9/0/1.

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

Running Configuration

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

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

Verification

Verify the parameters of the route.

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

Associated Commands

• route-tag

Larger IPv6 Address Space

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

IPv6 Address Formats

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

2001:0DB8:0:0:8:800:200C:417A
It is common for IPv6 addresses to contain successive hexadecimal fields of zeros. To make IPv6 addresses less cumbersome, two colons (::) can be used to compress successive hexadecimal fields of zeros at the beginning, middle, or end of an IPv6 address. (The colons represent successive hexadecimal fields of zeros.) Table 1: Compressed IPv6 Address Formats, on page 19 lists compressed IPv6 address formats.

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

---

**Note**

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

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

Table 1: Compressed IPv6 Address Formats

<table>
<thead>
<tr>
<th>Type</th>
<th>Preferred Format</th>
<th>Compressed Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicast</td>
<td>2001:0:0:0:0:DB8:800:200C:417A</td>
<td>1080::0DB8:800:200C:417A</td>
</tr>
<tr>
<td>Multicast</td>
<td>FF01:0:0:0:0:0:101</td>
<td>FF01::101</td>
</tr>
<tr>
<td>Loopback</td>
<td>0:0:0:0:0:0:0:1</td>
<td>::1</td>
</tr>
<tr>
<td>Unspecified</td>
<td>0:0:0:0:0:0:0:0</td>
<td>::</td>
</tr>
</tbody>
</table>

The loopback address listed in Table 1: Compressed IPv6 Address Formats, on page 19 may be used by a node to send an IPv6 packet to itself. The loopback address in IPv6 functions the same as the loopback address in IPv4 (127.0.0.1).

---

**Note**

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

The unspecified address listed in Table 1: Compressed IPv6 Address Formats, on page 19 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](image)

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.

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.

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

**Table 2: Basic IPv6 Packet Header Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>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.</td>
</tr>
<tr>
<td>Traffic Class</td>
<td>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.</td>
</tr>
<tr>
<td>Flow Label</td>
<td>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.</td>
</tr>
<tr>
<td>Payload Length</td>
<td>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.</td>
</tr>
<tr>
<td>Next Header</td>
<td>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.</td>
</tr>
<tr>
<td>Hop Limit</td>
<td>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.</td>
</tr>
<tr>
<td>Source Address</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

**Table 3: IPv6 Extension Header Types**

<table>
<thead>
<tr>
<th>Header Type</th>
<th>Next Header Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hop-by-hop options</td>
<td>0</td>
<td>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.</td>
</tr>
<tr>
<td>Destination options</td>
<td>60</td>
<td>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.</td>
</tr>
<tr>
<td>Routing header</td>
<td>43</td>
<td>The routing header is used for source routing.</td>
</tr>
</tbody>
</table>
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.

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.

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.

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.

Path MTU discovery is supported only for applications using TCP.

### IPv6 Neighbor Discovery

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

### IPv6 Neighbor Solicitation Message

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

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). 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.
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.

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.
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.

For more details on ARP, see Information About Configuring ARP, on page 36

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 32

Restrictions

The following restrictions apply to configuring ARP:

• Reverse Address Resolution Protocol (RARP) is not supported.

• ARP throttling, which is the rate limiting of ARP packets in Forwarding Information Base (FIB), is not supported.

ARP Cache Entries

ARP establishes correspondences between network addresses (an IP address, for example) and Ethernet hardware addresses. A record of each correspondence is kept in a cache for a predetermined amount of time and then discarded.
You can also add a static (permanent) entry to the ARP cache that persists until explicitly removed.

**Defining a Static ARP Cache Entry**

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

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

**Configuration Example**

A cache entry is created to establish connection between an IP address **203.0.1.2** and the MAC address **0010.9400.000c**. Additionally, the cache entry is created as an alias entry such that the interface to which the entry is attached will respond to ARP request packets for this network layer address with the data link layer address in the entry.

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

**Running Configuration**

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

**Verification**

Verify that the State is static for proper functioning:

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

**Proxy ARP and Local Proxy ARP**

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

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

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

- The target IP address is not on the same physical network (LAN) on which the request is received.
- The networking device has one or more routes to the target IP address.
- All of the routes to the target IP address go through interfaces other than the one on which the request is received.
When local proxy ARP is enabled, the networking device responds to ARP requests that meet all the following conditions:

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

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

### Enabling Proxy ARP

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

#### Configuration Example

Proxy ARP is enabled on the HundredGigE interface-0/9/0/0:

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

#### Running Configuration

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

#### Verification

Verify that proxy ARP is configured and enabled:

```
Router# show arp idb interface HundredGigE0/9/0/0 location 0/RP0/CPU0
  interface HundredGigE0/9/0/0 (0x08000038):
    IPv4 address 1.0.0.1, Vrf ID 0x60000000
    VRF Name default
    Dynamic learning: Enable
    Dynamic entry timeout: 14400 secs
    Purge delay: off
    IPv4 caps added (state up)
    MPLS caps not added
    Interface not virtual, not client fwd ref,
    **Proxy arp is configured, is enabled**
    Local Proxy arp not configured
    Packet IO layer is NetIO
    Srg Role : DEFAULT
```
Enabling Local Proxy ARP

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

Configuration Example

Local proxy ARP is enabled on the HundredGigE interface-0/9/0/0

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

Running Configuration

Router#show running-config interface HundredGigE0/9/0/1
interface HundredGigE0/9/0/0
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 interface HundredGigE0/9/0/0 location 0/RP0/CPU0
HundredGigE0/9/0/1 (0x08000038):
  IPv4 address 1.0.0.1, Vrf ID 0x60000000
  VRF Name default
  Dynamic learning: Enable
  Dynamic entry timeout: 14400 secs
  Purge delay: off
  IPv4 caps added (state up)
  MPLS caps not added
  Interface not virtual, not client fwd ref,
  Proxy arp not configured, not enabled
  **Local Proxy arp is configured**
  Packet IO layer is NetIO
  Srg Role : DEFAULT
  Idb Flag : 264332
  IDB is Complete

Associated Commands

- local-proxy-arp
- show arp idb

Configure Learning of Local ARP Entries

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

Use the following procedure to configure local ARP learning on an interface.
1. Enter the interface configuration mode.
   
   Router(config)# interface TenGigE 0/11/0/0

2. Configure the IPv4/IPv6 address for the interface.
   
   Router(config-if)# ipv4 address 12.1.3.4 255.255.255.0

3. Configure local ARP learning on the interface.
   
   Router(config-if)# arp learning local

4. Enable the interface and commit your configuration.
   
   Router(config-if)# no shut
   Router(config-if)# commit

5. Confirm your configuration.
   
   Router(config-if)# show running-configuration

6. Verify if local ARP learning is working as configured on the interface.
   
   Router(config-if)# do show arp idb TenGigE 0/11/0/0 location 0/RP0/CPU0

7. (Optional) You can monitor the ARP traffic on the interface.
   
   Router(config-if)# do show arp idb TenGigE 0/11/0/0 location 0/RP0/CPU0
Addressing 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.
Address Resolution When Interconnected by a Router
A DHCP relay agent is a host that forwards DHCP packets between clients and servers that do not reside on a shared physical subnet. Relay agent forwarding is distinct from the normal forwarding of an IP router where IP datagrams are switched between networks transparently.

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

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

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

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

- Prerequisites for Configuring DHCP Relay Agent, on page 40
- Limitations for DHCP Relay Feature, on page 40
- How to Configure and Enable DHCP Relay Agent, on page 40
- Configure a DHCP Proxy Profile, on page 45
- DHCP Server, on page 46
- DHCP Client, on page 50
- DHCP Proxy Binding Table Reload Persistency, on page 50
Prerequisites for Configuring DHCP Relay Agent

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

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

• A configured and running DHCP client and DHCP server.

• Connectivity between the relay agent and DHCP server

Limitations for DHCP Relay Feature

These are the limitations for implementing DHCP relay feature:

• The multicast addresses are not supported. The `helper-address` command in DHCP relay profile submode will only support global unicast IP address as the helper address.

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

Note

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

How to Configure and Enable DHCP Relay Agent

This section contains the following tasks:

Configuring and Enabling the DHCP Relay Agent

Configuration Example

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

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

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

Router(config-dhcpv4-relay-profile)# helper-address vrf A 10.10.10.1 giaddr 40.1.1.2
Router(config-dhcpv4-relay-profile)# broadcast-flag policy check
/* Configures VRF addresses for forwarding UDP broadcasts, including DHCP. */

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

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

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

Running Configuration
Router#show running-config
Tue May 23 10:56:14.463 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Tue May 23 10:56:08 2017 by annaseque
!
dhcp ipv4
vrf vrf1 relay profile client
profile r1 relay
  helper-address vrf A 10.10.10.1 giaddr 40.1.1.2
  broadcast-flag policy check
  relay information option vpn
  relay information option vpn-mode rfc
  relay information option allow-untrusted
!

Enabling 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-dhcpipv4)#interface HundredGigE 0/2/0/2 relay profile client
/* Attaches a relay profile to an interface.
To disable the DHCP relay on the interface, use the 'no interface HundredGigE 0/2/0/2 none'
command. */

router(config-dhcpv4-if)#commit

Running Configuration
Router#show running-config dhcp ipv4
dhcp ipv4
interface HundredGigE 0/2/0/2 relay profile client
!
Disabling DHCP Relay on an Interface

This task describes how to disable the DHCP relay on an interface by assigning the none profile to the interface.

Router# configure terminal
Router(config)# dhcp_ipv6
Router(config-dhcpv6)# interface type name none
Router(config-dhcpv6-if)# commit

Enabling DHCP Relay on a VRF

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

/CPU0:router# configure_terminal
Router(config)# dhcp_ipv6
Router(config-dhcpv6)# vrf vrf-name relay profile profile-name
Router(config-dhcpv6-if)# commit

Configure a DHCP Relay Profile with Multiple Helper Addresses

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

1. Enter the DHCPv4 or DHCPv6 configuration mode.
   
   Router(config)# dhcp_ipv6

2. Configure the DHCPv4 or DHCPv6 relay profile.
   
   Router(config-dhcpv6)# profile helper relay

3. Configure helper addresses.

   You can configure up to 16 IPv4 and IPv6 addresses.

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

4. Confirm your configuration.

   Router(config-dhcpv6-relay-profile)# show configuration

   !! IOS XR Configuration 0.0.0
   dhcp ipv6
   profile helper relay
   helper-address vrf default 2001:1:1::2
   !
   end

5. Commit your configuration.

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

6. Exit the configuration mode and verify the configured helper addresses.

   Router# show dhcp ipv6 relay profile name helper
   ...
   Profile: helper
   Helper Addresses:
DHCP relay agent notification for prefix delegation allows the router working as a DHCPv6 relay agent to find prefix delegation options by reviewing the contents of a DHCP RELAY-REPLY packet that is being relayed by the relay agent to the client. When the relay agent finds the prefix delegation option, the relay agent extracts the information about the prefix being delegated and inserts an IPv4 or IPv6 subscriber route matching the prefix delegation information onto the relay agent. Future packets destined to that prefix via relay are forwarded based on the information contained in the prefix delegation. The IPv4 or IPv6 subscriber route remains in the routing table until the prefix delegation lease time expires or the relay agent receives a release packet from the client releasing the prefix delegation.

The relay agent automatically does the subscriber route management.

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

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

Configuring DHCP Stateful Relay Agent for Prefix Delegation

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

Configuration Example

1. Configure a DHCP profile.
2. Configure the DHCP relay agent.
3. Enable IPv4 or IPv6 DHCP on an interface that acts as an IPv4 or IPv6 DHCP stateful relay agent.
4. Configure the profile name.
**DHCP Relay Profile: Example**

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

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

**DHCP Relay on an Interface: Example**

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

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

**DHCP Relay on a VRF: Example**

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

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

**Relay Agent Information Option Support: Example**

The following example shows how to enable the relay agent and the insertion and removal of the DHCP relay information option:
Relay Agent Giaddr Policy: Example

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

dhcp ipv4
    profile client relay
    giaddr policy drop
    !

Configure a DHCP Proxy Profile

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

Configuration Example

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

• The value of the address argument can be a specific DHCP server address or a network address (if other DHCP servers are on the destination network segment). Using the network address enables other servers to respond to DHCP requests.

• For multiple servers, configure one helper address for each server.

Configuration

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

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

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

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

DHCP Service-based Mode Selection

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

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

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

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

Configuring DHCP Server Profile

You can configure routers with DHCPv4 or DHCPv6 server profile.

Perform this task to configure the DHCPv6 server profile.

```
Router# configure
Router(config)# dhcp ipv6
Router(config-dhcpv6)# profile profile-name server
```
Configuring Multiple Classes with a Pool

Perform this task to configure multiple classes with a pool.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# dhcp ipv6
RP/0/RP0/CPU0:router(config-dhcpv6)# profile profile-name server
RP/0/RP0/CPU0:router(config-dhcpv6)# pool pool-name
RP/0/RP0/CPU0:router(config-dhcpv6)# class class-name
RP/0/RP0/CPU0:router(config-dhcpv6)# pool pool-name
RP/0/RP0/CPU0:router(config-dhcpv6)# match option [ ascii asciiString | hex hexString ]
RP/0/RP0/CPU0:router(config-dhcpv6)# exit
RP/0/RP0/CPU0:router(config-dhcpv6)# class class-name
RP/0/RP0/CPU0:router(config-dhcpv6)# pool pool-name
RP/0/RP0/CPU0:router(config-dhcpv6)# match vrf vrf-name
```

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 'dhcpp ipv4' command configures DHCP for IPv4 and enters the DHCPv4 configuration submode. */

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

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

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

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

```cpp
router#configure

router(config)#dhcp ipv4
/* The 'dhcp ipv4' command configures DHCP for IPv4 and enters the DHCPv4 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

```plaintext
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

```cpp
router#configure

router(config)#pool vrf ISP_1 ipv4 ISP1_POOL
/* Configures an IPv4 pool for the specified 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 specified 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
!

DHCP Client

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

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

Restrictions and Limitations

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

Enabling DHCP Client on an Interface

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

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

DHCP Proxy Binding Table Reload Persistency

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

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

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

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

Configuring DHCP Relay Binding Database Write to System Persistent Memory

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

Router# configure
Router(config)# dhcp_ipv6
Router(config-dhcpv6)# database relay [full-write-interval full-write-interval]
|incremental-write-interval incremental-write-interval|
Router(config-dhcpv6)# commit
Implementing DHCPv6 Relay Agent

A DHCPv6 relay agent is a host that forwards DHCPv6 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 packets are switched between networks transparently.

DHCPv6 clients use User Datagram Protocol (UDP) broadcasts to send DHCP DISCOVER 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 DHCPv6 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 DHCPv6 relay agent to forward DHCPv6 packets to a remote server by configuring a DHCPv6 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 DHCPv6 client broadcasts a request for an IPv6 address and additional configuration parameters on its local LAN. Acting as a DHCPv6 relay agent, Router B picks up the broadcast, changes the destination address to the DHCPv6 server’s address and sends the message out on another interface. The relay agent inserts the IPv6 address of the interface, on which the DHCPv6 client’s packets are received into the gateway address (giaddr) field of the DHCPv6 packet, which enables the DHCPv6 server to determine which subnet should receive the offer and identify the appropriate IPv6 address range. The relay agent unicasts the messages to the server address, in this case 2172:16:1::2 (which is specified by the helper address in the relay profile).

Figure 11: Forwarding UDP Broadcasts to a DHCPv6 Server Using a Helper Address

- Prerequisites for Configuring DHCP Relay Agent, on page 54
- Limitations for DHCP Relay Feature, on page 54
- How to Configure and Enable DHCP Relay Agent, on page 54
- Configure a DHCP Relay Profile with Multiple Helper Addresses, on page 55
Prerequisites for Configuring DHCP Relay Agent

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

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- A configured and running DHCP client and DHCP server.
- Connectivity between the relay agent and DHCP server

Limitations for DHCP Relay Feature

These are the limitations for implementing DHCP relay feature:

- The multicast addresses are not supported. The helper-address command in DHCP relay profile submode will only support global unicast IP address as the helper address.
- Only interface-id and remote-id DHCP option code are added by a relay agent while forwarding the packet to a DHCP server.

Note

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

How to Configure and Enable DHCP Relay Agent

This section contains the following tasks:

Configuring and Enabling the DHCPv6 Relay Agent

This task describes how to configure and enable a DHCPv6 relay agent.

```
RP/0/RSP0/CPU0:router# configure terminal
RP/0/RSP0/CPU0:router(config)# dhcp ipv6
RP/0/RSP0/CPU0:router(config-dhcpv6)# commit
```
Enabling DHCPv6 Relay Agent on an Interface

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

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

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

Disabling DHCP Relay on an Interface

This task describes how to disable the DHCP relay on an interface by assigning the none profile to the interface.

```
Router# configure terminal
Router(config)# dhcp ipv6
Router(config-dhcpv6)# interface type name none
Router(config-dhcpv6-if)# commit
```

Enabling DHCP Relay on a VRF

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

```
/CPU0:router# configure terminal
Router(config)# dhcp ipv6
Router(config-dhcpv6)# vrf vrf-name relay profile profile-name
Router(config-dhcpv6-if)# commit
```

Configure a DHCP Relay Profile with Multiple Helper Addresses

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

1. Enter the DHCPv4 or DHCPv6 configuration mode.

   ```
   Router(config)# dhcp_ipv6
   ```

2. Configure the DHCPv4 or DHCPv6 relay profile.

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

3. Configure helper addresses.

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

4. Confirm your configuration.
Router(config-dhcpv6-relay-profile)# show configuration

!! IOS XR Configuration 0.0.0
dhcp ipv6
profile helper relay
  helper-address vrf default 2001:1:1::2
!
end

5. Commit your configuration.

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

6. Exit the configuration mode and verify the configured helper addresses.

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

You have successfully configured the DHCPv6 relay helper address.

Configuring DHCP Relay Binding Database Write to System Persistent Memory

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

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

DHCP Server

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

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

### DHCP Service-based Mode Selection

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

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

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

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

### Configuring DHCP Server Profile

You can configure routers with DHCPv4 or DHCPv6 server profile.

Perform this task to configure the DHCPv6 server profile.

```conf
Router# configure
Router(config)# dhcp_ipv6
Router(config-dhcpv6)# profile profile-name server
Router(config-dhcpv6)# bootfile boot-file-name
Router(config-dhcpv6)# broadcast-flag policy unicast-always
Router(config-dhcpv6)# class class-name
Router(config-dhcpv6)# default-router address1 address2 ... address8
Router(config-dhcpv6)# lease {infinite | days minutes seconds }
Router(config-dhcpv6)# limit lease {per-circuit-id | per-interface | per-remote-id } value
```
### Configuring Multiple Classes with a Pool

Perform this task to configure multiple classes with a pool.

```bash
RP/0/RP0/CPU0:router# configure
```

```bash
RP/0/RP0/CPU0:router(config)# dhcp ipv6
```

```bash
RP/0/RP0/CPU0:router(config-dhcpv6)# profile profile-name server
```

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

```bash
RP/0/RP0/CPU0:router(config-dhcpv6-server-profile)# class class-name
```

```bash
RP/0/RP0/CPU0:router(config-dhcpv6-server-class)# pool pool-name
```

```bash
RP/0/RP0/CPU0:router(config-dhcpv6-server-class)# match option option [ sub-option sub-option ]
```

```bash
RP/0/RP0/CPU0:router(config-dhcpv6-server-class)# exit
```

```bash
RP/0/RP0/CPU0:router(config-dhcpv6-server-class)# commit
```

### DHCP Client

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

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

#### Restrictions and Limitations

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

### Enabling DHCP Client on an Interface

The DHCPv4 or DHCPv6 client can be enabled at an interface level. The DHCP component receives a notification when DHCPv4 or DHCPv6 is enabled or disabled on an interface.
Router# configure
Router(config)# interface MgmtEth rack/slot/CPU0/port
Router(config)# interface interface_name ipv6 address dhcp
Implementing DHCPv6 Relay Agent

Enabling DHCP Client on an Interface
Implementing Host Services and Applications

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

Network Connectivity Tools

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

Ping

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

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

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

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

Configuration for Checking Network Connectivity

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

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

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

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

Configuration Example

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

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

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

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

Associated Commands

Checking Network Connectivity for Multiple Destinations

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

Configuration Example

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

- 1: 1.1.1.1
- 2: 2.2.2.2
- 3: 3.3.3.3

Router# ping bulk ipv4 input cli batch
*/You must hit the Enter button and then specify one destination address per line*/
Please enter input via CLI with one destination per line and when done Ctrl-D/(exit) to initiate pings:
1: 1.1.1.1
2: 2.2.2.2
3: 3.3.3.3
4:
Starting pings...
Target IP address: 1.1.1.1
Repeat count [5]: 5
Datagram size [100]: 1
% A decimal number between 36 and 18024.
Datagram size [100]: 1
% A decimal number between 36 and 18024.
Datagram size [100]: 1000
Timeout in seconds [2]: 1
Interval in milliseconds [10]: 10
Extended commands? [no]: no
Sweep range of sizes? [no]: q
% Please answer 'yes' or 'no'.
Sweep range of sizes? [no]: q
% Please answer 'yes' or 'no'.
Sweep range of sizes? [no]: q
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:
### Associated Commands

**Traceroute**

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

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

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

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

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

### Checking Packet Routes

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

**Configuration Example**

Trace the route from 10.0.0.2 to 20.1.1.1:

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

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

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

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

Associated Commands

Domain Services

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

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

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

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

Configuring Domain Services

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

Define a static hostname-to-address mapping. Associate (or map) the IPv4 addresses (192.168.7.18 and 10.2.0.2 192.168.7.33) with two hosts. The host names are host1 and host2.

**Defining the Domain Host**

```
Router# configure
Router(config)# domain ipv4 host host1 192.168.7.18
Router(config)# domain ipv4 host host2 10.2.0.2 192.168.7.33
Router(config)# commit
```

**Defining the Domain Name**

```
*/Define cisco.com as the default domain name/*
Router# configure
Router(config)# domain name cisco.com
Router(config)# commit
```

**Specifying the Addresses of the Name Servers**

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

**Verification**

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

<table>
<thead>
<tr>
<th>Host</th>
<th>Flags</th>
<th>Age(hr)</th>
<th>Type</th>
<th>Address(es)</th>
</tr>
</thead>
<tbody>
<tr>
<td>host2</td>
<td>(perm, OK)</td>
<td>0</td>
<td>IP</td>
<td>10.2.0.2 192.168.7.33</td>
</tr>
<tr>
<td>host1</td>
<td>(perm, OK)</td>
<td>0</td>
<td>IP</td>
<td>192.168.7.18</td>
</tr>
</tbody>
</table>
```

**Associated Commands**

**TFTP Server**

It is too costly 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 allowing you to use your router for its regular functions.

Typically, a router that is configured as a TFTP server provides other routers with system image or router configuration files from its flash memory. You can also configure the router to respond to other types of services 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 sydb disk0:
default v4 telnet tcp 23 10 0 nowait telnetd sydb
```

### Associated Commands

## File Transfer Services

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

The file system interface uses URLs to specify the location of a file. URLs commonly specify files or locations on the WWW. However, on Cisco routers, URLs also specify the location of files on the router or remote file servers.
When a router crashes, it can be useful to obtain a copy of the entire memory contents of the router (called a core dump) for your technical support representative to use to identify the cause of the crash. SCP, FTP, TFTP, rcp can be used to save the core dump to a remote server.

**FTP**

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

**Configuring a Router to Use FTP Connections**

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

- Passive-mode FTP
- Password
- IP address

**Configuration Example**

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

```
Router#configure
Router(config)#ftp client passive
(Optional) Router(config)#ftp client vrf vrfa
Router(config)#ftp client anonymous-password xxxx
Router(config)#ftp client source-interface HundredGigE 0/9/0/0
Router(config)#commit
```

**Running Configuration**

```
Router#show running-config ftp client passive
ftp client passive
ftp client vrf vrfa
Router#show running-config ftp client anonymous-password xxxx
ftp client anonymous-password xxxx
Router#show running-config ftp client source-interface HundredGigE 0/9/0/0
ftp client source-interface HundredGigE 0/9/0/0
```

**Associated Commands**

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

**TFTP**

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

Configuration Example

Configure the router to use TFTP connections and set the IP address of the HundredGigE 0/9/0/0 as the source address for TFTP connections:

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

Running Configuration

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

Verification

```
Router#show cinetd services
Vrf Name Family Service Proto Port ACL max_cnt curr_cnt wait Program Client Option
default v4 tftp udp 69 unlimited 0 wait tftpd sysdb disk0:
default v4 telnet tcp 23 10 0 nowait telnetd sysdb
```

Associated Commands

- `tftp client source-interface type`
- `show cinetd services`

SCP

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

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

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

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

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

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

**Configuration Example**

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

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

**Verification**

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

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

**Associated Commands**

- `scp`

---

Cisco inetd

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

---

Telnet

Enabling Telnet allows inbound Telnet connections into a networking device.

**Configuration Example**

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

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

**Verification**

```
Router# show cinetd services
Vrf Name Family Service Proto Port ACL max_cnt curr_cnt wait Program Client Option
   default  v4     tftp    udp  69 unlimited 0    wait    tftpd    sysdb
 disk0:
   default  v4  telnet  tcp  23     10  0      nowait  telnetd sysdb
```
Associated Commands

**Syslog source-interface**

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

**Configuration Example**

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

```
Router#configure
Router(config)#logging source-interface Loopback2
Router(config)#logging source-interface Loopback3 vrf vrfa
Router(config)#commit
```

**Running Configuration**

```
Router#show running-config logging
/*Logging configuration after changing the source into loopback2 interface.*/
logging console debugging
logging monitor debugging
logging facility local4
logging 123.100.100.189 vrf default severity info port default
logging source-interface Loopback2
logging source-interface Loopback3 vrf vrfa
```

**Associated Commands**

- logging source-interface
- show running-configuration logging
Syslog source-interface
Implementing Access Lists and Prefix Lists

- Understanding Access Lists, on page 73
- User-Defined TCAM Keys for IPv4 and IPv6, on page 76
- Configuring IPv4 ACLs, on page 78
- Configuring IPv6 ACLs, on page 81
- Modifying ACLs, on page 86
- Configuring ACL-based Forwarding, on page 87
- ACLs on Bridge Virtual Interfaces, on page 89
- Configuring ACLs with Fragment Control, on page 92
- Configuring ACL Filtering by IP Packet Length, on page 97
- Understanding Object-Group ACLs, on page 100
- Configuring TTL Matching and Rewriting for IPv4 ACLs, on page 105
- Configuring Interface-Based Unique IPv4 ACLs, on page 106
- Configuring TTL Matching and Rewriting for IPv6 ACLs, on page 107
- Configuring Interface-Based Unique IPv6 ACLs, on page 108
- Understanding IP Access List Logging Messages, on page 110
- Understanding Prefix Lists, on page 110
- Configuring Prefix Lists, on page 111
- Sequencing Prefix List Entries and Revising the Prefix List, on page 112

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. There are 2 types of ACLs:

- Standard ACLs- Verifies only the source IP address of the packets. Traffic is controlled by the comparison of the address or prefix configured in the ACL, with the source address found in the packet.

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

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

**Purpose of IP Access Lists**

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

**How an IP Access List Works**

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

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

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

You can also filter packets on the basis of transport layer information, such as whether the packet is a TCP, UDP, 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.
- If a packet does not match an access list statement, the packet is then tested against the next statement in the list.
- If a packet and an access list statement match, the remaining statements in the list are skipped and the packet is permitted or denied as specified in the matched statement. The first entry that the packet matches
determines whether the software permits or denies the packet. That is, after the first match, no subsequent entries are considered.

- If the access list denies the address or protocol, the software discards the packet and returns an Internet Control Message Protocol (ICMP) Host Unreachable message. ICMP is configurable in the Cisco IOS XR software.

- If no conditions match, the software drops the packet because each access list ends with an unwritten or implicit deny statement. That is, if the packet has not been permitted or denied by the time it was tested against each statement, it is denied.

- The access list should contain at least one permit statement or else all packets are denied.

- Because the software stops testing conditions after the first match, the order of the conditions is critical. The same permit or deny statements specified in a different order could result in a packet being passed under one circumstance and denied in another circumstance.

- Only one access list per interface, per protocol, per direction is allowed.

- Inbound access lists process packets arriving at the router. Incoming packets are processed before being routed to an outbound interface. An inbound access list is efficient because it saves the overhead of routing lookups if the packet is to be discarded because it is denied by the filtering tests. If the packet is permitted by the tests, it is then processed for routing. For inbound lists, permit means continue to process the packet after receiving it on an inbound interface; deny means discard the packet.

- Outbound access lists process packets before they leave the router. Incoming packets are routed to the outbound interface and then processed through the outbound access list. For outbound lists, permit means send it to the output buffer; deny means discard the packet.

- An access list can not be removed if that access list is being applied by an access group in use. To remove an access list, remove the access group that is referencing the access list and then remove the access list.

- Before removing an interface, which is configured with an ACL that denies certain traffic, you must remove the ACL and commit your configuration. If this is not done, then some packets are leaked through the interface as soon as the `no interface <interface-name>` command is configured and committed.

- An access list must exist before you can use the `ipv4 access group` command.

**ACL Filtering by Wildcard Mask and Implicit Wildcard Mask**

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

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

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

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

You do not have to supply a wildcard mask with a source or destination address in an access list statement. If you use the `host` keyword, the software assumes a wildcard mask of 0.0.0.0.
Unlike subnet masks, which require contiguous bits indicating network and subnet to be ones, wildcard masks allow noncontiguous bits in the mask.

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

Including Comments in Access Lists

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

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

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

User-Defined TCAM Keys for IPv4 and IPv6

Access-lists on the use a TCAM (internal and external) to perform the lookup and action resolution on each packet. The TCAM is a valuable and constrained resource in hardware, which must be shared by multiple features. Therefore, the space (key width) available for these key definitions is also constrained. A key definition specifies which qualifier and action fields are available to the ACL feature when performing the lookup. Not all available qualifier and action fields can be included in each key definition.

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

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

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

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

The User-Defined TCAM Key (UDK) functionality provides the flexibility to define your own TCAM key for one of the three possible reasons (for ingress, traditional, IPv4/IPv6 ACL only):

To include qualifier fields which are not included in the default TCAM key

To change the ACL mode from shared to unique to support a greater number of unique ACLs, unique counters, etc.

To reduce the size of the TCAM key (number of banks consumed)

A UDK can be configured using the following command:

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

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

```
udf udf-name header [ inner | outer ] [ 12 | 13 | 14 ] offset byte-offset length no of bytes
```

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

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

Note: Up to 8 UDFs can be defined system wide. Currently, UDFs are globally defined.

IPv4 and IPv6 Key Formats for Traditional Ingress ACL

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

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

**Table 4: Qualifier Fields Supported in IPv4 and IPv6 Key Formats**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default TCAM Key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IPv4</td>
</tr>
<tr>
<td>Source Address</td>
<td>Enabled</td>
</tr>
<tr>
<td>Destination Address</td>
<td>Enabled</td>
</tr>
<tr>
<td>Source Port</td>
<td>Enabled</td>
</tr>
<tr>
<td>Destination Port</td>
<td>Enabled</td>
</tr>
<tr>
<td>Port Range</td>
<td>Enabled</td>
</tr>
<tr>
<td>Protocol/Next Header</td>
<td>Enabled</td>
</tr>
<tr>
<td>Fragment bit</td>
<td>Enabled</td>
</tr>
<tr>
<td>Packet length</td>
<td>Disabled</td>
</tr>
<tr>
<td>Precedence/DSCP</td>
<td>Disabled</td>
</tr>
<tr>
<td>TCP Flags</td>
<td>Enabled</td>
</tr>
<tr>
<td>TTL Match</td>
<td>Disabled</td>
</tr>
<tr>
<td>Interface based</td>
<td>Disabled</td>
</tr>
</tbody>
</table>
The following table shows the action fields supported in the IPv4 and IPv6 key formats.

### Table 5: Action Fields Supported in IPv4 and IPv6 Key Formats

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Action Field</th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permit</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Deny</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Log</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Capture</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Stats Counter</td>
<td>Deny stats is always Enabled (permit stats has its own hw-module command)</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>TTL Set</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

**Note**
- The Capture parameter is not supported on Jericho 2 ASIC line cards.
- For Jericho 2 ASIC line cards, both the Permit stats and Deny stats are always enabled. There is no need to use the hw-module command to enable Permit stats.

## Configuring IPv4 ACLs

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

### Notes and Restrictions for Configuring IPv4 Ingress ACLs

IPv4 ingress ACLs are characterized by the following behavior.

- Ingress IPv4 ACLs are supported on all interfaces except management interfaces.
ACL-based Forwarding (ABF) is supported only in the ingress direction.

- The total number of ACLs allowed by default per NPU is 31.
- The number of attached ACEs allowed per line card is 4000.
- ACL logging with input interface (using the `log-input` keyword) is not supported.
- The `show access-lists ipv4 acl_name stats` command is not supported for ACE with logs, to check statistics. Therefore, use the `show access-lists acl-name hardware [ingress | egress] detail location/loc` command to check statistics for ACE with logs.

**Notes and Restrictions for Configuring IPv4 Egress ACLs**

IPv4 egress ACLs are characterized by the following behavior.

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

  **Note**

  Egress ACLs are not directly supported on sub-interfaces. However, If you configure an egress ACL on a main interface that has sub-interfaces, the ACL action is also applied to the sub-interface traffic. This egress ACL behavior holds true even if the sub-interfaces are configured after the ACL is applied to the main interface.

- The total number of egress ACLs allowed per NPU is 255.
- ACL is not supported on Management interface on egress direction.
- The number of attached ACEs allowed per line card is 4000.
- ACL logging (using the `log` command) and ACL logging with input interface (using the `log-input` command) is not supported.

**Configuring an Ingress IPv4 ACL on a Gigabit Ethernet Interface**

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

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

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

<table>
<thead>
<tr>
<th>Interface</th>
<th>IP-Address</th>
<th>Status</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vrf-NameTenGigE0/11/0/0</td>
<td>10.1.1.1</td>
<td>Up</td>
<td>Up</td>
</tr>
</tbody>
</table>

/* Configure an IPv4 ingress ACL */
Router(config)# ipv4 access-list V4-ACL-INGRESS
Router(config-ipv4-acl)# 10 permit tcp 10.2.1.1 0.0.0.255 any
Router(config-ipv4-acl)# 20 deny udp any any
```
You have successfully configured an IPv4 ingress ACL on a Gigabit Ethernet interface.

**Configuring an Egress IPv4 ACL on a Gigabit Ethernet Interface**

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

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

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

<table>
<thead>
<tr>
<th>Interface</th>
<th>IP-Address</th>
<th>Status</th>
<th>Protocol</th>
<th>Vrf-Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TenGigE0/11/0/0</td>
<td>10.1.1.1</td>
<td>Up</td>
<td>Up</td>
<td>default</td>
</tr>
<tr>
<td>TenGigE0/11/0/1</td>
<td>20.1.1.1</td>
<td>Up</td>
<td>Up</td>
<td>default</td>
</tr>
</tbody>
</table>

/* Configure an IPv4 egress ACL */
Router(config)# ipv4 access-list V4-ACL-EGRESS
```
You have successfully configured an IPv4 egress ACL on a Gigabit Ethernet interface.

## Configuring IPv6 ACLs

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

### Notes and Restrictions for Configuring IPv6 Ingress ACLs

IPv6 ingress ACLs are characterized by the following behavior.

- Ingress IPv6 ACLs are supported on all interfaces:
- ACL-based Forwarding (ABF) is supported only in the ingress direction.
- The total number of ACLs allowed per NPU is 31.
- The number of attached ACEs allowed per line card is 2047.
- ACL logging with input interface (using the `log-input` keyword) is not supported.
• Packet Length (using the `pkt-length` keyword) is not supported.

• The `show access-lists ipv4 acl_name stats` command is not supported for ACE with logs, to check statistics. Therefore, use the `show access-lists acl-name hardware [ingress | egress] detail location` command to check statistics for ACE with logs.

**Notes and Restrictions for Configuring IPv6 Egress ACLs**

IPv6 egress ACLs are characterized by the following behavior:

• Configuring packet length is not supported on egress ACLs.

• TCP flags are not supported on egress ACLs.

• Egress ACLs are not supported on BVI interfaces and L2 interfaces.

• Configuring qos-group is not supported on egress ACLs.

• A throughput of 50% or less is supported on egress ACLs.

• Apart from the throughput limitation, router-generated traffic is not be affected by egress IPv6 ACLs.

• The total number of egress ACLs allowed per NPU is 255.

• The total number of attached ACEs allowed per line card is 2000.

• Configuring dynamic TCAM key is not supported on egress ACLs.

• Upto 160GB of total IPv6 egress ACL is supported per NPU.

**Configuring an Ingress IPv6 ACL on a Gigabit Ethernet Interface**

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

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

/* Verify if the interface is up */
Router(config)# do show ipv6 interface brief
Thu Jan 25 12:38:35.742 IST
TenGigE 0/11/0/0 [Up/Up]
   fe80::bd:b9ff:fea9:5606
   1001::1
...

/* Configure an IPv6 ingress ACL */
Router(config)# ipv6 access-list V6-INGRESS-ACL
Router(config-ipv6-acl)# 10 permit ipv6 any any
Router(config-ipv6-acl)# 20 deny udp any any
Router(config-ipv6-acl)# commit
Thu Jan 25 11:31:24.488 IST
Router(config-ipv6-acl)# exit

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

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

/* Verify if the ingress ACL has been successfully applied to the interface */
Router(config)# do show ipv6 interface
Thu Jan 25 11:32:48.028 IST
TenGigE 0/11/0/0 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::bd:b9ff:fea9:5606
  Global unicast address(es):
    1001::1, subnet is 1001::/64
  Joined group address(es):
    ff02::1:ff00:1 ff02::1:ffa9:5606 ff02::2
    ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachable is enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND cache entry limit is 100000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound common access list is not set, access list is V6-INGRESS-ACL
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0

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

Configuring an Egress IPv6 ACL on a Gigabit Ethernet Interface

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

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

/* Verify if the interface is up */
Router(config)# do show ipv6 interface brief
Thu Jan 25 12:38:35.742 IST
TenGigE 0/11/0/0 [Up/Up]
  fe80::bd:b9ff:fea9:5606
  1001::1
TenGigE 0/11/0/1 [Up/Up]
fe80::23:e9ff:fea8:a44e
2001::1

/* Configure an IPv6 egress ACL */
Router(config)# ipv6 access-list V6-EGRESS-ACL
Router(config-ipv6-acl)# 10 permit ipv6 any any
Router(config-ipv6-acl)# 20 deny udp any any
Router(config-ipv6-acl)# commit
Thu Jan 25 11:44:03.969 IST
Router(config-ipv6-acl)# exit

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

/* Apply the egress ACL to the GigE interface */
Router(config)# interface TenGigE 0/11/0/1
Router(config-if)# ipv6 access-group V6-EGRESS-ACL egress
Router(config-if)# commit
Thu Jan 25 11:45:12.682 IST
Router(config-if)# exit

/* Verify if the egress ACL has been successfully applied to the interface */
Router(config)# do show ipv6 interface
Thu Jan 25 11:46:43.234 IST
... TenGigE 0/11/0/1 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
  IPv6 is enabled, link-local address is fe80::23:e9ff:fea8:a44e
  Global unicast address(es):
    2001::1, subnet is 2001::/64
  Joined group address(es):
    ff02::1:ff00:1 ff02::1:ff02::1: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 100000000
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is V6-EGRESS-ACL
  Inbound common access list is not set, access list is not set
  Table Id is 0xe0800000
  Complete protocol adjacency: 0
  Complete glean adjacency: 0
  Incomplete protocol adjacency: 0
  Incomplete glean adjacency: 0
  Dropped protocol request: 0
  Dropped glean request: 0
...

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

**Configuring Ingress and Egress IPv6 ACLs on Bundle Interfaces**

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

/* Configure a bundle interface with an IPv6 address */
Router(config)# interface Bundle-Ether 1

---

IP Addresses and Services Configuration Guide for Cisco NCS 560 Series Routers, IOS XR Release 7.0.x
Implementing Access Lists and Prefix Lists

Configuring IPv6 ACLs

Router(config-if)# ipv6 address 3001::1/64
Router(config-if)# no shut
Router(config-if)# commit
Thu Jan 25 13:53:47.435 IST
Router(config-if)# exit

/* Configure an IPv6 egress ACL */
Router(config)# ipv6 access-list V6-EGRESS-ACL-bundle interface
Router(config-ipv6-acl)# 10 permit tcp any any range 3000 4000
Router(config-ipv6-acl)# 20 permit ipv6 any any
Router(config-ipv6-acl)# commit
Thu Jan 25 13:57:14.960 IST
Router(config-ipv6-acl)# exit

/* Configure an IPv6 ingress ACL to deny ingress traffic on the bundle interface */
Router(config)# ipv6 access-list V6-DENY-INGRESS-ACL
Router(config-ipv6-acl)# 10 deny ipv6 any any
Router(config-ipv6-acl)# commit
Thu Jan 25 13:59:23.198 IST
Router(config-ipv6-acl)# exit

/* Verify the egress and ingress ACL creation */
Router(config)# do show access-lists ipv6
Thu Jan 25 14:00:24.055 IST
ipv6 access-list V6-DENY-INGRESS-ACL
  10 deny ipv6 any any
ipv6 access-list V6-EGRESS-ACL-BI
  10 permit tcp any any range 3000 4000
  20 permit ipv6 any any
...

/* Apply the egress and ingress ACLs to the bundle interface */
Router(config)# interface Bundle-Ether 1
Router(config-if)# ipv6 access-group V6-EGRESS-ACL-BI egress
Router(config-if)# ipv6 access-group V6-DENY-INGRESS-ACL ingress
Router(config-if)# commit
Thu Jan 25 14:04:19.536 IST
Router(config-if)# exit

/* Verify if the ACLs have been successfully applied to the interface */
Router(config)# do show ipv6 interface
Thu Jan 25 11:46:43.234 IST
...
Thu Jan 25 14:04:51.322 IST
Bundle-Ether1 is Down, ipv6 protocol is Down, Vrfid is default (0xe6000000)
IPv6 is enabled, link-local address is fe80::1:10ff:fe87:8d04 [TENTATIVE]
Global unicast address(es):
  3001::1, subnet is 3001::/64 [TENTATIVE]
Joined group address(es): ff02::2 ff02::1
MTU is 1514 (1500 is available to IPv6)
ICMP redirects are disabled
ICMP unreachables are enabled
ND DAD is enabled, number of DAD attempts 1
ND reachable time is 0 milliseconds
ND cache entry limit is 1000000000
ND advertised retransmit interval is 0 milliseconds
ND router advertisements are sent every 160 to 240 seconds
ND router advertisements live for 1800 seconds
Hosts use stateless autoconfig for addresses.
Outgoing access list is V6-EGRESS-ACL-BI
Inbound common access list is not set, access list is V6-DENY-INGRESS-ACL
Table Id is 0xe0800000
Complete protocol adjacency: 0
You have successfully configured ingress and egress IPv6 ACLs on a bundle interface.

Modifying ACLs

This section describes a sample configuration for modification of ACLs.

/* Create an Access List*/
Router(config)#ipv4 access-list acl_1
*/Add entries (ACEs) to the ACL*/
Router(config-ipv4-acl)#10 permit ip host 10.3.3.3 host 172.16.5.34
Router(config-ipv4-acl)#20 permit icmp any any
Router(config-ipv4-acl)#30 permit tcp any host 10.3.3.3
Router(config-ipv4-acl)#end
*/Verify the entries of the ACL*/:
Router#show access-lists ipv4 acl_1
ipv4 access-list acl_1
10 permit ip host 10.3.3.3 host 172.16.5.34
20 permit icmp any any
30 permit tcp any host 10.3.3.3
*/Add new entries, one with a sequence number "15" and another without a sequence number to the ACL. Delete an entry with the sequence number "30":*/
Router(config)#ipv4 access-list acl_1
Router(config-ipv4-acl)# 15 permit 10.5.5.5 0.0.0.255
Router(config-ipv4-acl)# no 30
Router(config-ipv4-acl)# permit 10.4.4.4 0.0.0.255
Router(config-ipv4-acl)# commit
*/When an entry is added without a sequence number, it is automatically given a sequence number that puts it at the end of the access list. Because the default increment is 10, the entry will have a sequence number 10 higher than the last entry in the existing access list*/
*/Verify the entries of the ACL*/:
Router(config)#show access-lists ipv4 acl_1
ipv4 access-list acl_1
10 permit 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)*/
20 permit icmp any any
30 permit ipv4 10.4.4.0 0.0.0.255 any ---*newly added ACE (without the sequence number)*/
*/The entry with the sequence number 30, that is, "30 permit tcp any host 10.3.3.3" is deleted from the ACL*/
You have successfully modified ACLs in operation.
**Configuring ACL-based Forwarding**

Converged networks carry voice, video and data. Users may need to route certain traffic through specific paths instead of using the paths computed by routing protocols. This is achieved by specifying the next-hop address in ACL configurations, so that the configured next-hop address from ACL is used for forwarding packet towards its destination instead of routing packet-based destination address lookup. This feature of using next-hop in ACL configurations for forwarding is called ACL Based Forwarding (ABF).

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

---

**IPv4 ABF nexthops routed over GRE interfaces are supported.**

---

**Feature Highlights**

- ABF is only supported on ingress ACL.
- ABF supports nexthop modifications. You can modify a nexthop, remove a nexthop, or make changes between existing nexthops.

---

**Note**

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

- VRF-aware ABF is supported for IPv4 and IPv6 with up to three next hops.
- As ABF is ACL-based, packets that do not match an existing rule (ACE) in the ACL are subject to the default ACL rule (drop all). If the ACL is being used for ABF-redirect only (not for security), then include an explicit ACE rule at the end of the ACL (lowest user priority) to match and "permit" all traffic. This ensures that all traffic that does not match an ABF rule is permitted and forwarded as normal.
- ABF is supported on permit rules only.
- VRF-select (where only the VRF is configured for the nexthop) is not supported.
- ABF default route is not supported.
- Packets punted in the ingress direction from the NPU to the linecard CPU are not subjected to ABF treatment due to lack of ABF support in the slow path. These packets will be forwarded normally based on destination-address lookup by the software dataplane. Some examples of these types of packets are (but are not limited to) packets with IPv4 options, IPv6 extension headers, and packets destined for glean (unresolved/incomplete) adjacencies.
- Packets destined to the local IP interface ("for-us" packets) are subjected to redirect if they match the rule containing the ABF action. This can be avoided by either designing the rule to be specific enough to avoid matching the “for-us” packets or placing an explicit permit ACE rule (with higher priority) into the ACL before the matching ABF rule.
Configuration Example

To configure ACL-based forwarding, perform this task:
/* Enter IPv4 access list configuration mode and configure an ACL: */
router# configure
router(config)# ipv4 access-list abf-acl

/* Set the conditions for the ACL and configure ABF: */
/* The next hop for this entry is specified. */
router(config-ipv4-acl)# 10 permit ipv4 192.168.18.0 0.255.255.255 any nexthop1 ipv4 192.168.20.2
router(config-ipv4-acl)# 15 permit ipv4 192.168.21.0 0.0.0.255 any
router(config-ipv4-acl)# 20 permit ipv4 192.168.22.0 0.0.255.255 any nexthop1 ipv4 192.168.23.2

/* More than two nexthops */
router(config-ipv4-acl)# 25 permit tcp any range 2000 3000 any range 4000 5000 nexthop1 ipv4 192.168.23.1 nexthop2 ipv4 192.168.24.1 nexthop3 ipv4 192.168.25.1

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

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

router(config-ipv4-acl)# commit

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

Running Configuration

ipv4 access-list abf-acl
10 permit ipv4 192.168.18.0 0.255.255.255 any nexthop1 192.168.20.2
15 permit ipv4 192.168.21.0 0.0.0.255 any
20 permit ipv4 192.168.22.0 0.0.255.255 any nexthop1 192.168.23.2
25 permit tcp any range 2000 3000 any range 4000 5000 nexthop1 ipv4 192.168.23.1 nexthop2
ipv4 192.168.24.1 nexthop3 ipv4 192.168.25.1
30 permit tcp any eq www host 192.168.12.2 precedence immediate nexthop1 vrf vrf1_ipv4 ipv4 192.168.13.2 nexthop2 vrf vrf1_ipv4 ipv4 192.168.14.2
35 permit ipv4 any any
commit

Verification

Use the following command to verify the IP nexthop state in ABF to ensure that the expected nexthop is up:
Router# show access-lists ipv4 abf nexthops client pfilter_ea location 0/0/CPU0
Wed Jan 24 14:18:58.667 UTC

ACL name : abf-acl
ACE seq. NH-1 NH-2 NH-3
------------- --------------- --------------- ---------------
10 192.168.13.2 status UP
at status Not Present
exist No
vrf default
track
pd ctx Present
ACLs on Bridge Virtual Interfaces

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

Do not delete an ACL attached to a BVI interface when the BVI interface is not part of a bridge domain. Later, if you add the BVI interface to the bridge domain then the traffic is dropped.

Increased TCAM Consumption with Configuring ACLs on BVIs

The consumption of TCAM resources is impacted in the following manner when ACLs are configured on BVIs.

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

Restrictions for Configuring ACLs on BVIs

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

- Egress IPv6 ACLs are not supported on BVIs.
- When an egress IPv4 ACL is enabled on a BVI through the `hw-module` command, no other interface types are supported for the ACL (non-BVI interfaces are not supported for the ACL in this mode).

Prerequisites for Configuring IPv4 Egress ACLs on BVIs

By default, an IPv4 egress ACL on a BVI is disabled, and ACL filtering does not take place even when the ACL is attached to the BVI. Hence, we use the `hw-module` command, which enables the ACL when the line cards are reloaded.
IPv4 and IPv6 ingress ACLs do not require this configuration.

Use the following configuration to enable an IPv4 egress ACL on a BVI on the hardware and reload the line cards.

```bash
/* Enable an IPv4 egress ACL on BVI */
RP/0/RP0/CPU0:router(config)# hw-module profile acl egress layer3 interface-based
/* Enable permit statistics for the egress ACL (by default, only deny statistics are shown) */
RP/0/RP0/CPU0:router(config)# hw-module profile stats acl-permit
RP/0/RP0/CPU0:router(config)# commit
RP/0/RP0/CPU0:router(config)# end
RP/0/RP0/CPU0:router# reload location all
Wed Apr 5 23:05:46.193 UTC
Proceed with reload? [confirm]
```

**Configuration**

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

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

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

   ```bash
   RP/0/RP0/CPU0:router(config)# ipv4 access-list v4-acl-ingress
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit tcp any 10.1.1.0/24 dscp cs6
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 20 deny udp any any eq ssh
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 30 permit ipv4 any any
   RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
   RP/0/RP0/CPU0:router(config-ipv4-acl)# exit
   ```

2. Configure an IPv4 egress ACL.

   ```bash
   RP/0/RP0/CPU0:router(config)# ipv4 access-list v4-acl-egress
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 deny ipv4 any any fragments log
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 20 deny tcp any any ack
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 30 permit ipv4 any any
   RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
   RP/0/RP0/CPU0:router(config-ipv4-acl)# exit
   ```

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

   ```bash
   RP/0/RP0/CPU0:router(config)# interface GigabitEthernet 0/0/0/0
   RP/0/RP0/CPU0:router(config-if)# l2transport
   RP/0/RP0/CPU0:router(config-if-l2)# commit
   ```

4. Attach the ingress and egress ACLs to the BVI.

   ```bash
   RP/0/RP0/CPU0:router(config)# interface BVI1
   RP/0/RP0/CPU0:router(config-if)# ipv4 access-group v4-acl-ingress ingress
   RP/0/RP0/CPU0:router(config-if)# ipv4 access-group v4-acl-egress egress
   RP/0/RP0/CPU0:router(config-if)# commit
   RP/0/RP0/CPU0:router(config-if)# exit
   ```

5. Configure the bridge domain with the Gigabit Ethernet interface and BVI.
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group BG1
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain B1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/0
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)# routed interface BVI1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# commit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# exit
RP/0/RP0/CPU0:router(config-l2vpn-bg)# exit
RP/0/RP0/CPU0:router(config-l2vpn)# exit

6. Confirm that your configuration has been successfully committed.

RP/0/RP0/CPU0:router(config)# show run
...
ipv4 access-list v4-acl-egress
10 deny ipv4 any any fragments log
20 deny tcp any any ack
30 permit ipv4 any any
!
ipv4 access-list v4-acl-ingress
10 permit tcp any 10.1.1.0/24 dscp cs6
20 deny udp any any eq ssh
30 permit ipv4 any any
!
interface GigabitEthernet0/0/0/0
l2transport
!
interface BVI1
ipv4 address 209.165.200.224/27
ipv4 access-group v4-acl-ingress ingress
ipv4 access-group v4-acl-egress egress
!

l2vpn
bridge group BG1
bridge-domain B1
interface GigabitEthernet0/0/0/0
!
! ruted interface BVI1
!
!
!
end

7. Exit to the Executive Privileged mode and confirm that the ACLs are in operation.

RP/0/RP0/CPU0:router# show access-lists interface bvi1
Tue May 9 10:01:25.732 EDT
Input ACL (common): GigabitEthernet 0/0/0/0 (interface): v4-acl-ingress
Output ACL: v4-acl-egress

RP/0/RP0/CPU0:router# show access-lists summary
Tue May 9 10:02:01.167 EDT
ACL Summary:
Total ACLs configured: 2
Total ACEs configured: 6

RP/0/RP0/CPU0:router# show access-lists ipv4 v4-acl-egress hardware egress location 0/0/CPU0
The output clearly shows the configured ACLs, the total number of ACEs (three per ACL), and also the ACE matches in hardware.

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

### Configuring ACLs with Fragment Control

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

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

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

The behavior of access-list entries regarding the presence or absence of the `fragments` keyword can be summarized as follows:
<table>
<thead>
<tr>
<th>If the Access-List Entry has...</th>
<th>Then...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...no <strong>fragments</strong> keyword and all of the access-list entry information matches</td>
<td>For an access-list entry containing only Layer 3 information:</td>
</tr>
<tr>
<td></td>
<td>• The entry is applied to non-fragmented packets, initial fragments, and non-initial fragments.</td>
</tr>
<tr>
<td></td>
<td>For an access-list entry containing Layer 3 and Layer 4 information:</td>
</tr>
<tr>
<td></td>
<td>• The entry is applied to non-fragmented packets and initial fragments.</td>
</tr>
<tr>
<td></td>
<td>• If the entry matches and is a <strong>permit</strong> statement, the packet or fragment is permitted.</td>
</tr>
<tr>
<td></td>
<td>• If the entry matches and is a <strong>deny</strong> statement, the packet or fragment is denied.</td>
</tr>
<tr>
<td></td>
<td>• The entry is also applied to non-initial fragments in the following manner. Because non-initial fragments contain only Layer 3 information, only the Layer 3 portion of an access-list entry can be applied. If the Layer 3 portion of the access-list entry matches, and</td>
</tr>
<tr>
<td></td>
<td>• If the entry is a <strong>permit</strong> statement, the non-initial fragment is permitted.</td>
</tr>
<tr>
<td></td>
<td>• If the entry is a <strong>deny</strong> statement, the next access-list entry is processed.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> The deny statements are handled differently for non-initial fragments versus non-fragmented or initial fragments.</td>
</tr>
<tr>
<td>...the <strong>fragments</strong> keyword and all of the access-list entry information matches</td>
<td>The access-list entry is applied only to non-initial fragments.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> The <strong>fragments</strong> keyword cannot be configured for an access-list entry that contains any Layer 4 information.</td>
</tr>
</tbody>
</table>

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

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

**Note** The **fragments** keyword cannot solve all cases involving access lists and IP fragments.
Within the scope of ACL processing, Layer 3 information refers to fields located within the IPv4 header; for example, source, destination, protocol. Layer 4 information refers to other data contained beyond the IPv4 header; for example, source and destination ports for TCP or UDP, flags for TCP, type and code for ICMP.

**Configuring an IPv4 ACL to Match on Fragment Type**

Most DoS (Denial of Service) attacks work by flooding the network with fragmented packets. By filtering the incoming fragments of the packet in a network, an extra layer of protection can be added against such attacks.

You can configure an IPv4 ACL to match on the fragment type, and perform an appropriate action. You can use the following sample configuration with the different fragment options:

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

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

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

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

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

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

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

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

In this configuration, the ACL checks the fragment offset value ('0' for the first fragment). If the fragment is the first fragment of the packet, the packet is forwarded. If the fragment is the last fragment of the packet, it is dropped at the interface.

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

/* Configure an ACE to match on the first fragment. */
Router(config-ipv4-acl)# 10 permit tcp any any fragment-type first-fragment default

/* Configure an ACE to match on the last fragment. */
Router(config-ipv4-acl)# 50 permit icmp any any fragment-type last-fragment default

Router(config-ipv4-acl)# commit
```
If the fragment offset value equals 0, the fragment is forwarded to the 192.168.1.2 next hop */
Router(config-ipv4-acl)# 10 permit tcp any any fragment-type first-fragment nexthop1 ipv4 192.168.1.2
/* Configure an ACE to match on the last fragment, and drop the fragment at the interface. */
Router(config-ipv4-acl)# 20 deny tcp any any fragment-type last-fragment
Router(config-ipv4-acl)# commit
Thu Jan 11 12:01:33.297 IST
/* Validate the configuration */
Router(config-ipv4-acl)# do show access-lists
Thu Jan 11 12:05:23.646 IST
ipv4 access-list ACLFIRSTFRAG
  10 permit tcp any any fragment-type first-fragment nexthop1 ipv4 192.168.1.20
  20 deny tcp any any fragment-type last-fragment

You have successfully configured an IPv4 ACL to match on the fragment type.

Matching by Fragment Offset in ACLs

You can configure an access control list (ACL) rule to filter packets by the fragment-offset value. Depending on whether a packet matches the criteria in a permit or deny statement, the packet is either processed or dropped respectively at the interface. Fragment-offset filtering is supported only on ingress direction with compression mode of an ACL.

For more information about this feature, see the Implementing Access Lists and Prefix Lists chapter in the .For complete command reference, see the Access List Commands chapter in

Configuring ACL Matching by Fragment Offset

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

Note

For fragment-offset filtering, you must attach the particular ACL to an interface with compression level 3. Else, the configuration is rejected.

Configuration

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

/* Configure ACL */
Router# configure
Router(config)# ipv4 access-list fragment-offset-acl
Router(config-ipv4-acl)# 10 permit ipv4 any any fragment-offset range 300 400
Router# commit

/* Attach the ACL to the interface */
Router# configure
Router(config)# interface Bundle-Ether70
Router(config-if)# ipv4 access-group fragment-offset-acl ingress compress level 3
Router# commit

Running Configuration

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

interface Bundle-Ether70
ipv4 address 192.0.2.1 255.255.255.0
ipv6 address 2001:DB8::1/48
ipv4 access-group fragment-offset-acl ingress compress level 3
!

Verify Fragment-offset Match in ACL

Router# show access-lists ipv4 fragment-offset-acl usage pfilter loc 0/0/CPU0
Wed Apr 12 19:49:54.457 UTC
Interface : Bundle-Ether70
   Input ACL : Common-ACL : N/A ACL : fragment-offset-acl (comp-lvl 3)
   Output ACL : N/A

Router# show access-lists ipv4 fragment-offset-acl hardware ing int Bundle-Ether70 loc 0/0/CPU0
Wed Apr 12 19:51:07.837 UTC
ipv4 access-list fragment-offset-acl
10 permit ipv4 any any fragment-offset range 300 400

Associated Commands

- ipv4 access-list
- ipv6 access-list
- deny (IPv4)
- deny (IPv6)
- fragment-offset
- permit (IPv4)
- permit (IPv6)
Configuring ACL Filtering by IP Packet Length

You can configure an access control list to filter packets by the packet length at an ingress interface. Depending on whether a packet matches the packet-length condition in a permit or deny statement, the packet is either processed or dropped respectively at the interface.

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

Restrictions

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

- Packet length filtering is supported only on ingress direction, for both simple (non-compression) and hybrid (compression) ACLs.
- IPv6 packet length filtering is supported only for hybrid ACLs; not for simple ACLs.
- Only quantized (value divisible by 16) packet length filtering is supported for simple ACLs on IPv4.
- Packet length filtering is not supported in the default TCAM key, but instead requires a User-Defined TCAM Key (UDK) using the `hw-module profile tcam format` command as described in the configuration section.

Configuring Simple IPv4 ACLs to Filter by Packet Length

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

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

   ```
   Router# config
   Router(config)# hw-module profile tcam format access-list ipv4 dst-addr dst-port proto packet-length frag-bit port-range
   ```

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

   In this particular example, we configure a set of statements to process only those packets that match the specified packet length condition. All other packets are dropped when this ACL is applied to an ingress interface.

   ```
   Router# config
   Router(config)# ipv4 access-list pktlen-v4
   Router(config-ipv4-acl)# 10 permit tcp any any packet-length eq 1664
   Router(config-ipv4-acl)# 20 permit udp any any packet-length range 1600 2000
   Router(config-ipv4-acl)# 30 deny ipv4 any any
   ```

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

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

4. Apply the ACL to the required Ethernet interface.
Router(config)\# interface Te0/0/0/0
Router(config-if)# ipv4 access-group pktlen-v4 ingress

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

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

6. Verify your configuration.

Router# show access-lists pktlen-v4

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

7. Verify the ACL matches in hardware.

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

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

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

**Configuring Scaled IPv4 ACLs to Filter by Packet Length**

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

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

```text
Router# config
Router(config)# hw-module profile tcam format access-list ipv4 dst-addr dst-port proto packet-length frag-bit port-range
```

2. Enter the global configuration mode and create an object group for configuring a scaled ACL.

```text
Router(config)# object-group network ipv4 netobject1
Router(config-object-group-ipv4)# 50.0.0.0/24
Router(config-object-group-ipv4)# commit
```

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

In this particular example, we configure a statement to process only those packets that match the specified packet length condition. All other packets are dropped when this ACL is applied to an ingress interface.

```text
Router# configure
Router(config)# ipv4 access-list scaled_acl1
Router(config-ipv4-acl)# 10 permit ipv4 net-group netobject1 any packet-length eq 1000
```

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

```text
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# end
```
5. Apply the ACL to the required Gigabit Ethernet interface.

   Router(config)# interface Te0/0/0/3
   Router(config-if)# ipv4 access-group scaled_acl1 ingress

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

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

7. Verify your configuration.

   Router# show access-lists scaled_acl1
   ipv4 access-list scaled_acl1
   10 permit ipv4 net-group netobject1 any packet-length eq 1000

8. Verify the ACL matches in hardware.

   Router# show access-lists scaled_acl1 hardware ingress location 0/0/CPU0
   ipv4 access-list scaled_acl1
   10 permit ipv4 net-group netobject1 any packet-length eq 1000 (1500 hw matches)

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

### Configuring Scaled IPv6 ACLs to Filter by Packet Length

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

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

   Router# config
   Router(config)# hw-module profile tcam format access-list ipv4 dst-addr dst-port proto packet-length frag-bit port-range

2. Enter the global configuration mode and create an object group for configuring a scaled ACL.

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

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

   In this particular example, we configure a statement to process only those packets that match the specified packet length condition. All other packets are dropped when this ACL is applied to an ingress interface.

   Router(config)# ipv6 access-list scaled_acl2
   Router(config-ipv6-acl)# 10 permit ipv6 net-group netobject2 any packet-length eq 1000
   Router(config-ipv6-acl)# commit

4. Commit the ACL and exit the IPv6 ACL configuration mode.

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

5. Apply the ACL to the required Gigabit Ethernet interface.
Understanding Object-Group ACLs

You can use object-group ACLs to classify users, devices, or protocols into groups so you can have a group-level access control policy. Instead of specifying individual IP addresses, protocols, and port numbers in multiple ACEs, you can specify just the object group in a single ACL.

This feature is very beneficial in large scale networks which currently contain hundreds of ACLs. By using the object-group ACL feature, the number of ACEs per ACL are significantly reduced. Object-group ACLs are also more readable, and easier to manage than conventional ACLs. Using object-group ACLs instead of conventional ACLs optimizes the storage needed in TCAM.

Types of Object-Group ACLs

You can create two types of object-group ACLs on Cisco IOS XR:

- **Network object-group ACLs**: Consist of groups of host IP Addresses and network IP addresses.
- **Port object-group ACLs**: Consist of groups of ports and supporting Layer 3/Layer 4 protocols.

Compressing ACLs

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

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

There are only two compression levels in the access-group configuration for an ACL on an ingress interface:

- **Compress level 0**: No compression is done on the ACE fields.
In this mode, the object-group ACL behaves like a traditional ACL. Internal TCAM resources are utilized and there will be a huge impact on system resources and time taken for processing the ACL.

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

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

## Configuring an Object-Group ACL

### Before You Begin

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

- You can configure ACLs that contain both conventional and object-group ACEs.
- You can modify the objects in an object group dynamically without redefining the object group or the ACE that references the object group.
- You can configure an object-group ACL multiple times with a source group, or a destination group, or both source and destination groups.

### Restrictions

Configuring object-group ACLs involves the following restrictions:

- Object-group ACLs can only be configured to an interface. They cannot be used or referenced by applications like SSH, SNMP, NTP.
- To delete an object-group, you must first delete it from all ACLs.
- You cannot configure object-group ACLs along with QoS policies.
- Object-group ACLs are not supported in any policy based configuration.
- Nested object-groups are not supported from Release 6.2.1.

## Configuring a Network Object-Group ACL

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

### Configuration

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

```bash
/* From the global configuration mode, create a network object group. */
Router(config)# object-group network ipv4 netobj1
Router(config-object-group-ipv4)# description my-network-object
Router(config-object-group-ipv4)# host 10.1.1.1
Router(config-object-group-ipv4)# 10.2.1.0 255.255.255.0
Router(config-object-group-ipv4)# range 10.3.1.10 10.3.1.50

/* Create an access list referencing the object group. */
Router(config)# ipv4 access-list network-object-acl permit ipv4 net-group netobj1 any
```
/* Apply the access list containing the object group to the desired interface and commit your configuration. */
Router(config)# interface TenGigE0/0/0/10/3
Router(config-if)# ipv4 address 1.1.1.1/24
Router(config-if)# no shut
Router(config-if)# ipv4 access-group network-object-acl ingress compress level 3
Router(config-if)# commit
Tue Mar 28 10:23:34.106 IST

RP/0/0/CP00:Mar 28 10:37:48.570 : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : Interface
TenGigE0/0/0/10/3 , changed state to Down
RP/0/0/CP00:Mar 28 10:37:48.608 : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : Interface
TenGigE0/0/0/10/3 , changed state to Up

Router(config-if)# exit

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

/* From the global configuration mode, create a network object group. */
Router(config)# object-group network ipv6 netobj1
Router(config-object-group-ipv6)# description my-network-object
Router(config-object-group-ipv6)# host 2001:DB8:1::1
Router(config-object-group-ipv6)# 2001:DB8::1 2001:DB8:0:ABCD::1
Router(config-object-group-ipv6)# range 2001:DB8::2 2001:DB8::5

/* Create an access list referencing the object group. */
Router(config)# ipv6 access-list network-object-acl permit ipv6 net-group netobj1 any

/* Apply the access list containing the object group to the desired interface and commit your configuration. */
Router(config)# interface TenGigE0/0/0/10/3
Router(config-if)# ipv6 address 2001:DB8::1/32
Router(config-if)# no shut
Router(config-if)# ipv6 access-group network-object-acl ingress compress level 3
Router(config-if)# commit
Tue Mar 28 10:23:34.106 IST

RP/0/0/CP00:Mar 28 10:37:48.570 : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : Interface
TenGigE0/0/0/10/3 , changed state to Down
RP/0/0/CP00:Mar 28 10:37:48.608 : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : Interface
TenGigE0/0/0/10/3 , changed state to Up

Router(config-if)# exit

Running Configuration

Confirm your configuration.

Router(config)# show run
Tue Mar 28 10:37:55.737 IST

Building configuration...
!! IOS XR Configuration 0.0.0
... 

object-group network ipv4 netobj1
10.2.1.0/24
You have successfully configured a network object-group ACL.

**Configuring a Port Object-Group ACL**

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

**Configuration**

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

```sh
/* From the global configuration mode, create a port object group, and commit your configuration. */
RP/0/RP0/CPU0:router(config)# object-group port portobj1
RP/0/RP0/CPU0:router(config-object-group-ipv4)# description my-port-object
RP/0/RP0/CPU0:router(config-object-group-ipv4)# eq bgp
RP/0/RP0/CPU0:router(config-object-group-ipv4)# range 100 200
RP/0/RP0/CPU0:router(config-object-group-ipv4)# commit
RP/0/RP0/CPU0:router(config-object-group-ipv4)# exit

/* Create an access list referencing the object group. */
RP/0/RP0/CPU0:router(config)# ipv4 access-list port-object-acl permit ipv4 net-group portobj1

/* Apply the access list containing the object group to the desired interface and commit your configuration. */
RP/0/RP0/CPU0:router(config)# interface Te0/0/0/3
RP/0/RP0/CPU0:router(config-if)# ipv4 address 2.2.2.2/24
RP/0/RP0/CPU0:router(config-if)# ipv4 access-group port-obj-acl ingress compress level 3
RP/0/RP0/CPU0:router(config-if)# no shut
RP/0/RP0/CPU0:router(config-if)# commit

Tue Mar 28 10:23:34.106 IST
RP/0/RP0/CPU0:Mar 28 10:23:34.106 IST : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : Interface TenGigE0/0/0/10/3 , changed state to Down
RP/0/RP0/CPU0:Mar 28 10:23:34.106 IST : ifmgr[397]: %PKT_INFRA-LINK-3-UPDOWN : Interface TenGigE0/0/0/10/3 , changed state to Up
RP/0/RP0/CPU0:router(config-if)# exit

**Running Configuration**

Confirm your configuration.

RP/0/RP0/CPU0:router(config)# show run

Tue Mar 28 10:37:55.737 IST
Building configuration...
!! IOS XR Configuration 0.0.0
...
object-group port portobj1
eq bgp
range 100 200
!

ipv4 access-list port-object-acl
10 permit tcp net-group portobj1
!
interface Te/0/0/3
ipv4 access-group port-obj-acl ingress compress level 3
!
end
!

You have successfully configured a port object-group ACL.

**Verifying Object-Group ACL Compression**

You can use the commands described in this section to verify the configured object-group ACLs in operation and the compression of the ACEs in the ACL.

**Note**
The outputs provided in this section are a standalone sample and are not related to the configurations provided in the preceding sections.

**Verification**

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

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

RP/0/RP0/CPU0:router# show access-lists ipv4 network-object-acl hardware ingress location 0/0/CPU0
ipv4 access-list network-object-acl
40 permit ospf net-group n_192.168.0.0.16 any (20898463272 matches)
70 permit tcp any net-group CORP_ALL_V4 established
100 permit udp net-group INTERNAL port-group KERBEROS_UDP net-group CORP_ALL_V4
130 permit udp net-group INTERNAL port-group DNS_UDP net-group CORP_ALL_V4
160 permit udp net-group INTERNAL port-group NTP net-group CORP_ALL_V4
190 permit udp net-group INTERNAL port-group LDAP_UDP net-group CORP_ALL_V4
...
1500 permit udp net-group VLAN60_SECURITY net-group h_192.168.77.242 port-group UDP_50000-50100
1530 deny ipv4 net-group VLAN60_SECURITY any log (20891956640 matches)
...

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

RP/0/RP0/CPU0:router# show access-lists ipv4 network-object-acl hardware ingress verify location 0/0/CPU0
Verifying TCAM entries for network-object-acl
Please wait...
You have successfully verified the compression of ACEs within an ACL.

**Configuring TTL Matching and Rewriting for IPv4 ACLs**

You can configure ACLs to match on the TTL value specified in the IPv4 header. You can specify the TTL match condition to be based on a single value, or multiple values. You can also rewrite the TTL value in the IPv4 header by using the `set ttl` command.

**Limitations for using TTL matching and rewriting for IPv4 ACLs**

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

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

**Configuration**

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

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

/* Configure an IPv4 ACL with the TTL parameters */
Router(config)# ipv4 access-list acl-v4
Router(config-ipv4-acl)# 10 deny tcp any any ttl eq 100
Router(config-ipv4-acl)# 20 permit tcp any any ttl range 1 50 set ttl 200
Router(config-ipv4-acl)# 30 permit tcp any any ttl neq 100 set ttl 255
Router(config-ipv4-acl)# commit
Thu Nov 2 12:22:58.948 IST
/* Attach the IPv4 ACL to the GigE interface */
Router(config)# interface Te0/0/0/0
Router(config-if)# ipv4 address 15.1.1.1 255.255.255.0
Router(config-if)# ipv4 access-group acl-v4 ingress
Router(config-if)# commit

Running Configuration

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

Router(config)# show run
Thu Nov 2 14:01:53.376 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Thu Nov 2 12:22:59 2017 by annseque
!
hw-module profile tcam format access-list ipv4 dst-addr dst-port proto port-range enable-set-ttl ttl-match
ipv4 access-list acl-v4
  10 deny tcp any any ttl eq 100
  20 permit tcp any any ttl range 1 50 set ttl 200
  30 permit tcp any any ttl neq 100 set ttl 255
!
interface Te0/0/0/0
ipv4 address 15.1.1.1 255.255.255.0
ipv4 access-group acl-v4 ingress
!

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

### Configuring Interface-Based Unique IPv4 ACLs

ACLs that are shared across interfaces and use the same TCAM space are known as shared ACLs. However, you can configure only 31 unique, shared ACLs. To configure more unique ACLs, ACL sharing must be disabled by using the `interface-based` command. By making the ACLs unique for an interface, you can configure more than 31 ACLs.

#### Configuration

Use the following configuration to create unique, interface-based IPv4 ACLs.

```plaintext
/* Enable interface-based, unique IPv4 ACLs */
Router(config)# hw-module profile tcam format access-list ipv4 src-addr src-port dst-addr dst-port interface-based

/* Configure an IPv4 ACL with the TTL parameters */
```
Router(config)# ipv4 access-list acl-v4
Router(config-ipv4-acl)# 10 deny tcp any any ttl eq 100
Router(config-ipv4-acl)# 20 permit tcp any any ttl range 1 50 set ttl 200
Router(config-ipv4-acl)# 30 permit tcp any any ttl neq 100 set ttl 255
Router(config-ipv4-acl)# commit
Thu Nov 2 12:22:58.948 IST

/* Attach the IPv4 ACL to the GigE interface */
Router(config)# interface Te0/0/0/0
Router(config-if)# ipv4 address 15.1.1.1 255.255.255.0
Router(config-if)# ipv4 access-group acl-v4 ingress
Router(config-if)# commit

Running Configuration
Validate your configuration by using the show run command.

Router(config)# show run
Thu Nov 2 14:01:53.376 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Thu Nov 2 12:22:59 2017 by annseque
!!
hw-module profile tcam format access-list ipv4 src-addr src-port dst-addr dst-port
interface-based
!
ipv4 access-list acl-v4
  10 deny tcp any any ttl eq 100
  20 permit tcp any any ttl range 1 50 set ttl 200
  30 permit tcp any any ttl neq 100 set ttl 255
!
interface Te0/0/0/0
ipv4 address 15.1.1.1 255.255.255.0
ipv4 access-group acl-v4 ingress
!

You have successfully configured unique, interface-based IPv4 ACLs.

Configuring TTL Matching and Rewriting for IPv6 ACLs

You can configure ACLs to match on the TTL value specified in the IPv6 header. You can specify the TTL match condition to be based on a single value, or multiple values. You can also rewrite the TTL value in the IPv6 header by using the set ttl command.

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

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

• TTL matching is supported only for ingress ACLs.

• ACL logging is not supported for ingress ACLs after a User-Defined TCAM Key (UDK) is configured with the enable-set-ttl option.
• If a TTL rewrite is applied to the outer IPv6 header of an IP-in-IP header, then when the outer IPv6 header is decapsulated, (by GRE decapsulation) the TTL rewrite is also applied to the inner IPv6 header.

• TTL matching is not supported in the default TCAM key, but instead requires a User-Defined TCAM Key (UDK) using the `hw-module profile tcam format` command as described in the Configuration section.

**Configuration**

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

/* Enable TTL matching and rewriting in the global configuration mode by using the hw-module command */
Router(config)# hw-module profile tcam format access-list ipv6 dst-addr dst-port src-port next-hdr enable-set-ttl ttl-match

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

/* Attach the IPv6 ACL to the GigE interface */
Router(config)# interface Te0/0/0/0
Router(config-if)# ipv6 address 2001:2:1::1/64
Router(config-if)# ipv6 access-group acl-v6 ingress
Router(config-if)# commit

**Running Configuration**

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

Router(config)# show run
Thu Nov 2 14:01:53.376 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Thu Nov 2 12:22:59 2017 by annseque
!hw-module profile tcam format access-list ipv6 dst-addr dst-port src-port next-hdr enable-set-ttl ttl-match
!
ipv6 access-list acl-v6
  10 deny tcp any any ttl eq 50
  20 permit tcp any any ttl lt 50 set ttl 255
  30 permit tcp any any ttl gt 50 set ttl 200
!
interface Te0/0/0/0
ipv6 address 2001:2:1::1/64
  ipv6 access-group acl-v6 ingress
!

You have successfully configured TTL matching and rewriting for IPv6 ACLs.

**Configuring Interface-Based Unique IPv6 ACLs**

ACLs that are shared across interfaces and use the same TCAM space are known as shared ACLs. However, you can configure only 31 unique, shared ACLs. To configure more unique ACLs, ACL sharing must be
disabled by using the **interface-based** command. By making the ACLs unique for an interface, you can configure more than 31 ACLs.

**Configuration**

Use the following configuration to create unique, interface-based IPv6 ACLs.

**Note**

- A reboot of the line cards is required after entering the `hw-module profile` command to activate the command.
- TCAM is allocated for each ACL on an interface, and is not shared across ACLs. Hence, for instance, if an ACL utilizes 10 TCAM entries, and is applied to 100 interfaces, the total number of TCAM entries will be 1000.

```plaintext
/* Enable interface-based, unique IPv6 ACLs */
Router(config)# hw-module profile tcam format access-list ipv6 src-addr src-port dst-addr dst-port next-hdr interface-based

/* Configure an IPv6 ACL with the TTL parameters */
Router(config)# ipv6 access-list acl-v6
Router(config-ipv6-acl)# 10 deny tcp any any ttl eq 100
Router(config-ipv6-acl)# 20 permit tcp any any ttl range 1 50 set ttl 200
Router(config-ipv6-acl)# 30 permit tcp any any ttl neq 100 set ttl 255
Router(config-ipv6-acl)# commit
Thu Nov 2 12:22:58.948 IST

/* Attach the IPv6 ACL to the GigE interface */
Router(config)# interface Te0/0/0/0
Router(config-if)# ipv6 address 2001:2:1::1/64
Router(config-if)# ipv6 access-group acl-v6 ingress
Router(config-if)# commit

Running Configuration

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

```plaintext
Router(config)# show run
Thu Nov 2 14:01:53.376 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Thu Nov 2 12:22:59 2017 by annseque
!
hw-module profile tcam format access-list ipv6 src-addr src-port dst-addr dst-port next-hdr interface-based
!
ipv6 access-list acl-v6
  10 deny tcp any any ttl eq 100
  20 permit tcp any any ttl range 1 50 set ttl 200
  30 permit tcp any any ttl neq 100 set ttl 255
!
interface Te0/0/0/0
  ipv6 address 2001:2:1::1/64
  ipv6 access-group acl-v6 ingress
!
```

You have successfully configured unique, interface-based IPv6 ACLs.
Understanding IP Access List Logging Messages

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

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

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

**Caution**

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

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

**Note**

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

Understanding Prefix Lists

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

**BGP Filtering Using Prefix Lists**

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

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

Greater flexibility.

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

**How the System Filters Traffic by Prefix List**

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

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

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

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

### Configuring Prefix Lists

#### Configuration Example

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

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

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

```
Router(config-ipv4_pfx)#commit
```

#### Running Configuration

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

Verification

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

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

Associated Commands

Sequencing Prefix List Entries and Revising the Prefix List

Configuration Example

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

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 0.0.255.255
  !
/* 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
  35 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
  35 permit 172.16.0.0/16

Associated Commands
Implementing Cisco Express Forwarding

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

Components

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

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

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

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

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

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

CEF Benefits

• Improved performance—CEF is less CPU-intensive than fast-switching route caching. More CPU processing power can be dedicated to Layer 3 services such as quality of service (QoS) and encryption.

• Scalability—CEF offers full switching capacity at each line card.

• Resilience—CEF offers an unprecedented level of switching consistency and stability in large dynamic networks. In dynamic networks, fast-switched cache entries are frequently invalidated due to routing changes. These changes can cause traffic to be process switched using the routing table, rather than fast switched using the route cache. Because the Forwarding Information Base (FIB) lookup table contains all known routes that exist in the routing table, it eliminates route cache maintenance and the fast-switch or process-switch forwarding scenario. CEF can switch traffic more efficiently than typical demand caching schemes.

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

• IPv4 CEF database—Stores IPv4 Unicast routes for forwarding IPv4 unicast packets

• IPv6 CEF database—Stores IPv6 Unicast routes for forwarding IPv6 unicast packets

• MPLS LFD database—Stores MPLS Label table for forwarding MPLS packets

Verifying CEF

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

• show cef {ipv4 address | ipv6 address} hardware egress

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

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

• show cef {ipv4 | ipv6} summary

Displays a summary of the IPv4 or IPv6 CEF table.

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

Router ID is 216.1.1.1

IP CEF with switching (Table Version 0) for node0_RP0_CPU0

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

• show cef { ipv4 address | ipv6 address } detail

Displays the details of the IPv4 or IPv6 CEF table.

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

Router# show adjacency detail

<table>
<thead>
<tr>
<th>Interface</th>
<th>Address</th>
<th>Version</th>
<th>Refcount</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hu0/9/0/0</td>
<td>(interface)</td>
<td>13</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Per-Flow Load Balancing

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

Per-flow load balancing performs these functions:

- Incoming data traffic is evenly distributed over multiple equal-cost connections.
- Incoming data traffic is evenly distributed over multiple equal-cost connections member links within a bundle interface.
- Layer 2 bundle and Layer 3 (network layer) load balancing decisions are taken on IPv4, IPv6, and MPLS flows. If it is an IPv4 or an IPv6 payload, then a 7-tuple hashing is done. If it is an MPLS payload with three or less labels, then the hardware parses the payload underneath and identifies whether the payload packet has an IPv4 or an IPv6 header. If it is an IPv4 or IPv6 header, then a 4-tuple hashing is performed based on the IP source, IP destination, router ID, and label stack; otherwise, an MPLS label based hashing is performed. In case of MPLS label-based hashing, the top 4 labels are used in hash computation. However, for J2 line cards, all the labels are used for MPLS label-based hashing.
- A 7-tuple hash algorithm provides more granular load balancing and used for load balancing over multiple equal-cost Layer 3 (network layer) paths. The Layer 3 (network layer) path is on a physical interface or on a bundle interface. In addition, load balancing over member links can occur within a Layer 2 bundle interface.
- The 7-tuple load-balance hash calculation contains:
  - Source IP address
  - Destination IP address
  - IP Protocol type
  - Router ID
  - Source port
  - Destination port
  - Input interface

Note: J2 line cards support 6-tuple load-balance hash calculation because input interface is not considered as a parameter for load-balance hash calculations.
Per-Destination Load Balancing

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

Configuring Static Route

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

Configuration Example

Create a static route between Router A and B over a HundredGigE interface. The destination IP address is 203.0.1.2/32 and the next hop address is 1.0.0.2.

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

Running Configuration

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

![Diagram showing router A and B connected by a HundredGigE interface]
```

Verification

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

```
Router#show cef 203.0.1.2/32 hardware egress detail location 0/RP0/CP0
203.0.1.2/32, version 0, internal 0x1020001 0x0 (ptr 0x8d7db7f0) [1], 0x0 (0x8daeedf0), 0x0 (0x0)
Updated Nov 20 13:33:23.557
local adjacency 203.0.1.2
Prefix Len 32, traffic index 0, Adjacency-prefix, precedence n/a, priority 15
  via 203.0.1.2/32, HundredGigE0/9/0/0, 3 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8cfc81a0 0x0]
```
BGP Attributes Download

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

- The `show cef bgp-attribute` command displays the installed BGP attributes in CEF.

- The `show cef bgp-attribute attribute-id` command and the `show cef bgp-attribute local-attribute-id` command are used to view the specific BGP attributes by attribute ID and local attribute ID.

Verification

Router# show cef bgp-attribute
Router ID is 216.1.1.1

IP CEF with switching (Table Version 0) for node0_RP0_CPU0

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

0 PD backwalks on LDIs with backup path

VRF: default

Table ID: 0xe0000000. Total number of entries: 0
OOR state: GREEN. Number of OOR attributes: 0
Proactive Address Resolution Protocol and Neighbor Discovery

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

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

Associated Commands

- show cef bgp-attribute
Implementing LPTS

- LPTS Overview, on page 121
- LPTS Policers, on page 121
- Defining Dynamic LPTS Flow Type, on page 126

LPTS Overview

Local Packet Transport Services (LPTS) maintains tables describing all packet flows destined for the secure domain router (SDR), making sure that packets are delivered to their intended destinations.

LPTS uses two components to accomplish this task: the port arbitrator and flow managers. The port arbitrator and flow managers are processes that maintain the tables that describe packet flows for a logical router, known as the Internal Forwarding Information Base (IFIB). The IFIB is used to route received packets to the correct Route Processor for processing.

LPTS interfaces internally with all applications that receive packets from outside the router. LPTS functions without any need for customer configuration. However, the policer values can be customized if required. The LPTS show commands are provided that allow customers to monitor the activity and performance of LPTS flow managers and the port arbitrator.

LPTS Policers

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

```
show lpts pifib hardware police
```

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

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

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

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

**Configuration Example**

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

- ospf unicast default rate 3000
- bgp default rate 4000

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

**Running Configuration**

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

**Verification**

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

**Configuration Example**

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

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

Running Configuration

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

Verification

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

Router#show lpts pifib hardware police location 0/RP0/CPU0
------------------------------------------------------------
Node 0/RP0/CPU0:
------------------------------------------------------------
FlowType Policer Type Cur. Rate Burst npu
---------------------- ------- ------- --------- --------- ---------
OSPF-uc-default 32106 np 3000 1000 0
BGP-default 32118 np 4000 1250 0

Verification

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

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

<table>
<thead>
<tr>
<th>Trap Type</th>
<th>NFU</th>
<th>Trap</th>
<th>TrapStats</th>
<th>Policer</th>
<th>Packet</th>
<th>Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ID</td>
<td>ID</td>
<td>ID</td>
<td>Accepted</td>
<td>Dropped</td>
<td></td>
</tr>
<tr>
<td>RxTrapMimSaMove (CFM_DOWN_MEP_DMM)</td>
<td>0</td>
<td>6</td>
<td>0x6</td>
<td>32037</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapMimSaUnknown (RCY_CFM_DOWN_MEP_DMM)</td>
<td>0</td>
<td>7</td>
<td>0x7</td>
<td>32037</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapAuthSaLookupFail (IPMC default)</td>
<td>0</td>
<td>8</td>
<td>0x8</td>
<td>32033</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapSaMulticast</td>
<td>0</td>
<td>11</td>
<td>0xb</td>
<td>32018</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapArpMyIp</td>
<td>0</td>
<td>13</td>
<td>0xd</td>
<td>32001</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapArp</td>
<td>0</td>
<td>14</td>
<td>0xe</td>
<td>32001</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapDhcpv4Server</td>
<td>0</td>
<td>18</td>
<td>0x12</td>
<td>32022</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapDhcpv4Client</td>
<td>0</td>
<td>19</td>
<td>0x13</td>
<td>32022</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapDhcpv6Server</td>
<td>0</td>
<td>20</td>
<td>0x14</td>
<td>32022</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapDhcpv6Client</td>
<td>0</td>
<td>21</td>
<td>0x15</td>
<td>32022</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapL2Cache_LACP</td>
<td>0</td>
<td>23</td>
<td>0x17</td>
<td>32003</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RxTrapL2Cache_LLDP1</td>
<td>0</td>
<td>24</td>
<td>0x18</td>
<td>32004</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Event Description</td>
<td>Count</td>
<td>Type</td>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
<td>Value 4</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>-------</td>
<td>-------</td>
<td>---------</td>
<td>-----------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>RxTrapL2Cache_LLDP2</td>
<td>25</td>
<td>0x19</td>
<td>32004</td>
<td>1205548</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapL2Cache_LLDP3</td>
<td>26</td>
<td>0x1a</td>
<td>32004</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapL2Cache_ELMI</td>
<td>27</td>
<td>0x1b</td>
<td>32005</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapL2Cache_BPDU</td>
<td>28</td>
<td>0x1c</td>
<td>32027</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapL2Cache_BUNDLE_BPDU</td>
<td>29</td>
<td>0x1d</td>
<td>32027</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapL2Cache_CDP</td>
<td>30</td>
<td>0x1e</td>
<td>32002</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapHeaderSizeErr</td>
<td>31</td>
<td>0x20</td>
<td>32018</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapIpCompMcInvalidIp</td>
<td>32</td>
<td>0x21</td>
<td>32018</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapMyMacAndIpDisabled</td>
<td>33</td>
<td>0x22</td>
<td>32018</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapMyMacAndMplsDisable</td>
<td>34</td>
<td>0x23</td>
<td>32018</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapArpReply</td>
<td>35</td>
<td>0x24</td>
<td>32001</td>
<td>2693</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapFibDrop</td>
<td>36</td>
<td>0x25</td>
<td>32018</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapMTU</td>
<td>37</td>
<td>0x26</td>
<td>32020</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapMiscDrop</td>
<td>38</td>
<td>0x27</td>
<td>32020</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapL2AclDeny</td>
<td>39</td>
<td>0x28</td>
<td>32034</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Rx_UNKNOWN_PACKET</td>
<td>40</td>
<td>0x29</td>
<td>32018</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapL3AclDeny</td>
<td>41</td>
<td>0x2a</td>
<td>32034</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapOamY1731MplsTp(OAM_SWOFF_DN_CCM)</td>
<td>42</td>
<td>0x2b</td>
<td>32029</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapOamY1731Pwe(OAM_SWOFF_DN_CCM)</td>
<td>43</td>
<td>0x2c</td>
<td>32030</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapOamLevel</td>
<td>44</td>
<td>0x2d</td>
<td>32023</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapRedirectToCpuOamPacket</td>
<td>45</td>
<td>0x2e</td>
<td>32025</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapOamPassive</td>
<td>46</td>
<td>0x2f</td>
<td>32024</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrap1588</td>
<td>47</td>
<td>0x30</td>
<td>32038</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapExternalLookupError</td>
<td>48</td>
<td>0x31</td>
<td>32023</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapArplookupFail</td>
<td>49</td>
<td>0x32</td>
<td>32001</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapUcLooseRpfFail</td>
<td>50</td>
<td>0x33</td>
<td>32035</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapMplsControlWordTrap</td>
<td>51</td>
<td>0x34</td>
<td>32015</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapMplsControlWordDrop</td>
<td>52</td>
<td>0x35</td>
<td>32015</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapMplsUnknownLabel</td>
<td>53</td>
<td>0x36</td>
<td>32018</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapIpv4VersionError</td>
<td>54</td>
<td>0x37</td>
<td>32018</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapIpv4ChecksumError</td>
<td>55</td>
<td>0x38</td>
<td>32018</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RxTrapIpv4HeaderLengthError</td>
<td>56</td>
<td>0x39</td>
<td>32018</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Defining Dynamic LPTS Flow Type

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

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

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

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

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

Note
The sum of maximum LPTS entries configured for all flow types must not exceed 8000 entries per line card.

Configuration Example

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

```
Router#configure
Router(config)#lpts pifib hardware dynamic-flows location 0/1/CPU0
Router(config-pifib-flows-per-node)#flow bgp-known max 1800
Router(config-pifib-flows-per-node)#flow ISIS-known max 500
Router(config-pifib-flows-per-node)#flow RSVP-known max 0
Router(config-pifib-flows-per-node)#commit
```
Running Configuration

Router# show run lpts pifib hardware dynamic-flows location 0/1/CPU0
flow bgp known max 1800
flow isis-known 500
flow RSVP-known 0

Verification

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

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

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

<table>
<thead>
<tr>
<th>FLOW-TYPE</th>
<th>C</th>
<th>Def_Max</th>
<th>Conf_Max</th>
<th>HWCnt/ActLimit</th>
<th>SWCnt</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fragment</td>
<td>F</td>
<td>2</td>
<td></td>
<td>2/2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>OSPF-mc-known</td>
<td>T</td>
<td>600</td>
<td></td>
<td>2/600</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>OSPF-mc-default</td>
<td>F</td>
<td>4</td>
<td></td>
<td>4/4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>OSPF-uc-known</td>
<td>T</td>
<td>300</td>
<td></td>
<td>1/300</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>OSPF-uc-default</td>
<td>F</td>
<td>2</td>
<td></td>
<td>2/2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ISIS-known</td>
<td>T</td>
<td>300</td>
<td>500</td>
<td>500/300</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ISIS-default</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>BGP-known</td>
<td>T</td>
<td>900</td>
<td>1800</td>
<td>1800/900</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BGP-cfg-peer</td>
<td>T</td>
<td>900</td>
<td></td>
<td>0/900</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BGP-default</td>
<td>F</td>
<td>4</td>
<td></td>
<td>4/4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>PIM-mcast-default</td>
<td>F</td>
<td>40</td>
<td></td>
<td>0/40</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PIM-mcast-known</td>
<td>T</td>
<td>300</td>
<td></td>
<td>0/300</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PIM-ucast</td>
<td>F</td>
<td>40</td>
<td></td>
<td>2/40</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ICMP</td>
<td>T</td>
<td>1200</td>
<td></td>
<td>0/1200</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ICMP-local</td>
<td>F</td>
<td>4</td>
<td></td>
<td>4/4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>ICMP-control</td>
<td>F</td>
<td>5</td>
<td></td>
<td>5/5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>ICMP-default</td>
<td>F</td>
<td>9</td>
<td></td>
<td>9/9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>ICMP-app-default</td>
<td>F</td>
<td>2</td>
<td></td>
<td>2/2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LDP-TCP-known</td>
<td>T</td>
<td>300</td>
<td></td>
<td>0/300</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>LDP-TCP-cfg-peer</td>
<td>T</td>
<td>300</td>
<td></td>
<td>0/300</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>LDP-TCP-default</td>
<td>F</td>
<td>40</td>
<td></td>
<td>0/40</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>LDP-UDP</td>
<td>T</td>
<td>300</td>
<td></td>
<td>0/300</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>All-routers</td>
<td>T</td>
<td>300</td>
<td></td>
<td>0/300</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RSVP-default</td>
<td>F</td>
<td>4</td>
<td></td>
<td>1/4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RSVP-known</td>
<td>T</td>
<td>300</td>
<td></td>
<td>0/300</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SNMP</td>
<td>T</td>
<td>300</td>
<td></td>
<td>0/300</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SSH-known</td>
<td>T</td>
<td>150</td>
<td></td>
<td>0/150</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SSH-default</td>
<td>F</td>
<td>40</td>
<td></td>
<td>0/40</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TELNET-known</td>
<td>T</td>
<td>150</td>
<td></td>
<td>0/150</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TELNET-default</td>
<td>F</td>
<td>4</td>
<td></td>
<td>0/4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>UDP-default</td>
<td>F</td>
<td>2</td>
<td></td>
<td>2/2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>TCP-default</td>
<td>F</td>
<td>2</td>
<td></td>
<td>2/2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Raw-default</td>
<td>F</td>
<td>2</td>
<td></td>
<td>2/2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>GRE</td>
<td>F</td>
<td>4</td>
<td></td>
<td>0/4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VRRP</td>
<td>T</td>
<td>150</td>
<td></td>
<td>150/150</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
In the above show command output, the last column P specifies the pending software flow entries for the flow type.
Implementing VRRP

• Configuring VRRP, on page 129
• Enabling Multiple Group Optimization (MGO) for VRRP, on page 141
• Configuring SNMP Server Notifications for VRRP Events, on page 142

Configuring VRRP

The Virtual Router Redundancy Protocol (VRRP) feature allows for transparent failover at the first-hop IP router, enabling a group of routers to form a single virtual router. For more information on VRRP and related concepts, see Understanding VRRP, on page 129.

Restrictions for Configuring VRRP

• If you configure the command hw-module vrrpscale enable on the router, upto 255 VRRP groups (IPv4 and IPv6 combined) are supported on Cisco NCS 5500 Series Routers and Cisco NCS 540 Series routers. By default, 16 VRRP groups (IPv4 and IPv6 combined) are supported on Cisco NCS 5500 Series Routers and Cisco NCS 540 Series routers.
• If you do not configure the command hw-module vrrpscale enable on the router and if you configure BFD along with BVI and if all the BVIIs are sharing the same Chassis (default) MAC, the VRRP scale is reduced to 13. You cannot use any custom BVI MAC in this mode until VRRP scale is reduced to 11.
• If you configure BFD along with BVI and all the BVIIs are configured with one custom MAC or mix of one custom MAC and Chassis (default) MAC, the VRRP scale is reduced to 11.
• ICMP redirects are not supported.

Understanding VRRP

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

Note

VRRP is supported over VRF.
VRRP Overview

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

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

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

An alternative to dynamic 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**.

For example, Figure 12: Basic VRRP Topology, on page 131 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).
Because the virtual router uses the IP address of the physical interface of Router A, Router A assumes the role of the master virtual router and is also known as the IP address owner. As the master virtual router, Router A controls the IP address of the virtual router and is responsible for forwarding packets sent to this 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 master virtual router fails, the router configured with the higher priority becomes the master virtual router and provides uninterrupted service for the LAN hosts. When Router A recovers, it becomes the master virtual router again.

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 100 virtual routers on a router interface. You can configure up to 256 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 a master 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 master 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 master virtual router.

If no VRRP router owns the IP address, the priority of a VRRP router, combined with the reemt settings, determines if a VRRP router functions as a master or a backup virtual router. By default, the highest priority VRRP router functions as master, and all the others function as backups. Priority also determines the order of ascendency to becoming a master virtual router if the master 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 master 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 master 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 master virtual router.

By default, a preemptive scheme is enabled whereby a higher-priority backup virtual router that becomes available takes over from the current master 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 master upon the failure of the original higher priority master, remains the master even if the original master virtual router recovers and becomes available again.

**VRRP Advertisements**

The master virtual router sends VRRP advertisements to other VRRP routers in the same group. The advertisements communicate the priority and state of the master 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 master 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 master 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 Master 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.

```
/* 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
GigabitEthernet0/0/0/1, changed state to Down
GigabitEthernet0/0/0/1, changed state to Up
```

**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 master virtual Router.
Router(config-if)# exit
Router(config)# do show ip int brief
Fri Dec  8 13:50:05.505 IST

<table>
<thead>
<tr>
<th>Interface</th>
<th>IP-Address</th>
<th>Status</th>
<th>Protocol</th>
<th>Vrf-Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GigabitEthernet0/0/0/0</td>
<td>unassigned</td>
<td>Shutdown</td>
<td>Down</td>
<td>default</td>
</tr>
<tr>
<td>GigabitEthernet0/0/0/0/1</td>
<td>10.10.10.1</td>
<td>Up</td>
<td>Up</td>
<td>default</td>
</tr>
<tr>
<td>GigabitEthernet0/0/0/2</td>
<td>unassigned</td>
<td>Shutdown</td>
<td>Down</td>
<td>default</td>
</tr>
<tr>
<td>GigabitEthernet0/0/0/3</td>
<td>unassigned</td>
<td>Shutdown</td>
<td>Down</td>
<td>default</td>
</tr>
<tr>
<td>GigabitEthernet0/0/0/4</td>
<td>unassigned</td>
<td>Shutdown</td>
<td>Down</td>
<td>default</td>
</tr>
</tbody>
</table>

/* 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 master virtual Router. */
Router(config-vrrp-virtual-router)# preempt delay 15

/* CUSTOMIZATION: Configure the interval between successive advertisements by the master 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 master 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 master
virtual Router. */
Router(config-vrrp-virtual-Router)# preempt delay 15

/* CUSTOMIZATION: Configure the interval between successive advertisements by the master
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
You have successfully validated VRRP for IPv6 networks.

**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.
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 */
Router# configure
Router(config)# interface HundredGigE0/0/0.1 l2transport
Router(config-subif)# exit
Router(config)# interface HundredGigE0/0/1.1 l2transport
Router(config-subif)# commit
Router(config-subif)# exit

/* Enter the Layer 2 VPN configuration mode and Configure a bridge group */
Router(config)# 12vpn
Router(config-12vpn)# bridge group 5
Router(config-12vpn-bg-bd)# bridge-domain 5
Router(config-12vpn-bg-bd)# interface HundredGigE 0/0/0/1
Router(config-12vpn-bg-bd-ac)# exit
Router(config-12vpn-bg-bd)# interface HundredGigE 0/0/1/1
Router(config-12vpn-bg-bd-ac)# exit
Router(config-12vpn-bg-bd)# routed interface BVI 10
Router(config-12vpn-bg-bd-bvi)# commit
Router(config-12vpn-bg-bd-bvi)# exit

/* Configure a BVI in the global configuration mode*/
Router(config-12vpn-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)# 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::2/64
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 12vpn bridge-domain detail

Legend: pp = Partially Programmed.
*Bridge group: 10, bridge-domain: 10, id: 0, 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: 64000, 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 cvpIsConfigIndex: 1
Filter MAC addresses:
P2MP PW: disabled
Create time: 25/07/2018 19:04:41 (3w5d ago)
No status change since creation
ACS: 2 (2 up), VFIs: 0, PWs: 0 (0 up), PEBs: 0 (0 up), VNI: 0 (0 up)
List of ACS:
  AC: BVI10, state is up
    Type Routed-Interface
      MTU 1514; XC ID 0x80000002; interworking none
      BVI MAC address:
        008a.9651.e0e0
      Split Horizon Group: Access
  AC: Bundle-Ether10.1, state is up
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [10, 10]
      MTU 1500; XC ID 0xa0000001; interworking none
      MAC learning: enabled
Flooding:
  Broadcast & Multicast: enabled
  Unknown unicast: enabled
MAC aging time: 300 s, Type: inactivity
MAC limit: 64000, 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: bridge-domain policer
Static MAC addresses:
Statistics:
  packets: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 4429828
  bytes: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 344854904
  MAC move: 0
  Storm control drop counters:
Use the following command to show the VRRP details:

`Router# show vrrp ipv4 detail`

**BVI10 - IPv4 vrID 10**

*State is Master*

- 63 state changes, last state change 5d20h
- State change history:
  - Aug 15 20:30:47.986 UTC Backup -> Init Interface Down update
  - Aug 15 20:30:50.642 UTC Init -> Backup Delay timer expired
  - Aug 15 20:30:51.304 UTC Backup -> Init Interface Down update
  - Aug 15 20:31:44.957 UTC Init -> Backup Delay timer expired
  - Aug 15 20:31:48.562 UTC Backup -> Master Master down timer expired
- Last resign sent: Aug 15 20:30:20.700 UTC
- Last resign received: Jul 25 19:04:21.466 UTC
- 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*

- 63 state changes, last state change 5d20h
- State change history:
  - Aug 15 20:30:48.032 UTC Backup -> Init Interface Down update
  - Aug 15 20:30:50.517 UTC Init -> Backup Delay timer expired
  - Aug 15 20:30:51.348 UTC Backup -> Init Interface Down update
  - Aug 15 20:31:44.996 UTC Init -> Backup Delay timer expired
  - Aug 15 20:31:48.605 UTC Backup -> Master Master down timer expired
- Last resign sent: Aug 15 20:30:20.702 UTC
- Last resign received: Never
- Virtual IP address is fe80::200:5eff:fe00:20b
  - Secondary Virtual IP address is 2001:DB8:A::2/64
- 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 + (155 x 1/256))
- Minimum delay 1 sec, reload delay 5 sec
- Current priority 100
  - Configured priority 100, may preempt
    - minimum delay 0 secs
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 slaves of this primary session, and inherit their states from it.

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-virtual-router)#name s1
/* Specifies the VRRP session name. */

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

Slave Follow
```
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
/* Configures a slave follow. Instructs the slave 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 slave group. */

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

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

Primary and Secondary Virtual IPv4 Addresses for the Slave Group
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 master, and forwards packets sent to these IP addresses. The election process provides dynamic fail over (standby) in the forwarding responsibility should the master 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 master state or if the standby router is made master.
Configuration Example

Enables SNMP server notifications (traps) for VRRP.

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

Use the `show snmp traps details` command to view details of SNMP server notifications.
CHAPTER 10

Information About Configuring TCP, UDP Transports

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

- Graceful Restart, on page 145
- TCP Overview, on page 145
- UDP Overview, on page 146

Graceful Restart

You can use nonstop forwarding (NSF) for BGP to forward data packets along known routes in the Forward Information Base (FIB) while the BGP routing protocol information is being restored following a failover. With NSF, BGP peers do not experience routing flaps. During a failover, the data traffic is forwarded through intelligent modules while the standby supervisor becomes active.

If a Cisco router experiences a cold reboot, the network does not forward traffic to the router and removes the router from the network topology. In this scenario, BGP experiences a nongraceful restart and removes all routes. When Cisco operating system applies the startup configuration, BGP reestablishes peering sessions and relearns the routes.

A Cisco router that has dual supervisors can experience a stateful supervisor switchover. During the switchover, BGP uses nonstop forwarding to forward traffic based on the information in the FIB, and the system is not removed from the network topology. A router whose neighbor is restarting is referred to as a “helper.” After the switchover, a graceful restart operation begins. When it is in progress, both routers reestablish their neighbor relationship and exchange their BGP routes. The helper continues to forward prefixes pointing to the restarting peer, and the restarting router continues to forward traffic to peers although those neighbor relationships are restarting. When the restarting router has all route updates from all BGP peers that are graceful restart capable, the graceful restart is complete, and BGP informs the neighbors that it is operational again.

TCP Overview

TCP is a connection-oriented protocol that specifies the format of data and acknowledgments that two computer systems exchange to transfer data. TCP also specifies the procedures the computers use to ensure that the data arrives correctly. TCP allows multiple applications on a system to communicate concurrently, because it handles all demultiplexing of the incoming traffic among the application programs.
UDP Overview

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

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

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